A MODULAR DESIGN APPROACH FOR IMPROVING THE LEARNING PROCESS OF UNDERGRADUATE STUDENTS IN THE EMBEDDED SYSTEM DESIGN LABORATORY

By

Danilo Alfonso Rojas Méndez

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

December, 2016

Approved by:

Manuel Jiménez, Ph.D. Chair, Graduate Committee	Date
Aidsa I. Santiago-Román, Ph.D. Co-Chair, Graduate Committee	Date
Leyda León, Ph.D. Member, Graduate Committee	Date
Wilford Schmidt, Ph.D. Member, Graduate Committee	Date
Cecilio Ortiz, Ph.D. Graduate Studies Representative	Date
José Colom, Ph.D. Department Chairperson	Date

Abstract of Thesis Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

A MODULAR DESIGN APPROACH FOR IMPROVING THE LEARNING PROCESS OF UNDERGRADUATE STUDENTS IN THE EMBEDDED SYSTEM DESIGN LABORATORY

By

Danilo Alfonso Rojas Méndez

December 2016

Chair: Dr. Manuel Jiménez
Department: Electrical and Computer Engineering Department

Teaching embedded systems design concepts and enhancing student skills in this area is an important task for universities in order to provide an up to date education. Several methodologies have been developed and implemented to teach the fundamental concepts and at the same time enhance students' practical skills. The purpose of this thesis was to develop a teaching methodology based on a modular design approach, aided by an Outcome-Based educational framework. To achieve this objective, the content, pedagogical methods, and assessment activities were aligned to ensure a proper student learning. The modular approach was applied on the pedagogical methods through the design of progressive laboratory experiments and educational modules. Using this methodology, effective laboratory experiments, that promoted better student learning in the area of embedded systems design, were developed. As a result, the overall laboratory student performance was improved and therefore the proposed methodology was validated.

Resumen de tesis presentado a la Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
requerimientos para el grado de Maestría en Ciencias

UNA ESTRATEGIA DE DISEÑO MODULAR PARA MEJORAR EL PROCESO DE APRENDIZAJE DE LOS ESTUDIANTES SUBGRADUADOS EN EL LABORATORIO DE DISEÑO DE SISTEMAS EMBEDIDOS

Por

Danilo Alfonso Rojas Méndez

Diciembre 2016

Consejero: Dr. Manuel Jiménez
Departamento: Ingeniería Eléctrica y Computadoras

La enseñanza de conceptos relacionados al diseño de sistemas embebdidos y mejorar las habilidades practicas de los estudiantes en esta área es un tarea importante para las universidades con el fin de proporcionar una educacion actualizada. Varias metodologias se han desarrollado e implementado para enseñar los conceptos fundamentales y al mismo tiempo mejorar las habilidades prácticas de los estudiante en esta área. El propósito de esta tesis fue desarrollar una metodología de enseñanza basada en un diseño modular, ayudada por un marco educativo basado en resultados. Para lograr este objetivo, el contenido, los métodos pedagógicos y las actividades de evaluación se alinearon para asegurar un aprendizaje adecuado del estudiante. El enfoque modular se aplicó a los métodos pedagógicos a través del diseño de experimentos de laboratorio progresivos y módulos educativos. Utilizando esta metodología, se desarrollaron experimentos de laboratorio eficaces que promovieron un mejor aprendizaje de los estudiantes en el área del diseño de sistemas embebidos. Como resultado, el rendimiento general de los estudiantes de laboratorio fue mejorado y por lo tanto la metodología propuesta fue validada.

To my family, specially to my mother Carmen, my father Alfonso, and brother Hector, who have always given me their love, affection and support to keep going.

Acknowledgements

I would like to express my thanks to my advisors Professor Dr. Manuel Jiménez and Dr. Aidsa Santiago, thank you for encouraging my research and for allowing me to grow as a student, researcher, and professional. Also, thanks to my committee members, professor Dr. Leyda León and professor Dr. Wilford Schmidt for serving as my committee and for taking part in the review of my work. Thanks to MSc. Juan Patarroyo for his work because it served a base for my research. I would especially like to thank Sandy, the graduate academic counselor and friend, who guided and helped me through this experience in a new university and country. Last but not least, I want to thanks all my Puerto Rican and Colombian friends, who have encourage and supported me to move forward during good and bad moments.

Contents

List of Tables

List of Figures

List of Abbreviations

ADC	Analog-to-Digital Converter
BE	Basic Exercise
CGUDPCE	Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering
CT	Complementary Task
DAC	Digital-to-Analog Converter
ECE	Electrical and Computer Engineering
ESD	Embedded System Design
ESDC	Embedded System Design Course
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
I^2C	Inter-Integrated Circuit
ICOM	Ingenieria en Computadoras
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
MCU	Microcontroller Unit
OBE	Outcome-Based Education
PWM	Pulse-Width Modulation
UPRM	University of Puerto Rico at Mayagüez

Chapter 1
INTRODUCTION

An embedded system is defined by Jiménez et. al., as *"A device that contains tightly coupled hardware and software components to perform a single function, forms part of a larger system, is not intended to be independently programmable by the user, and is expected to work with minimal or no human interaction"* [2]. These type of devices are extensively used in a wide range of applications (Automotive, military/aerospace systems, home automation, and industrial control among others) however its usage is continuously growing due to the integration technologies that have been developed; denoting the relevance of embedded systems in our daily lives [3].

Due to the relevance of embedded systems, universities offer courses related to this kind of devices as a part of their electrical and computer engineering curriculum. Such courses cover fundamental aspects of embedded systems design, including basic topics such as microcontroller unit (MCU) architectures, peripherals interfacing, and more advanced topics such as real-time operating systems, fault tolerant design, among others [4–8].

Different methodologies have been developed and implemented to teach the fundamental concepts and at the same time enhance students' practical skills in this area. Some of these methods use problems [9–11], projects [12–14], video games [15, 16] or virtual labs [17, 18], to attract and motivate students in their learning process. Once there are several methodologies that have been developed, well-designed courses on this area need to cover not only aspects related to embedded systems architecture but also in the design process of these devices.

This work proposes a strategy based on modular design techniques and an Outcome-Based Education (OBE) approach to teach embedded design concepts in an applied laboratory. An OBE framework using a modular design approach attempts to align the content, pedagogical, and assessment methods in the laboratory experience. The course content was revised taking into consideration industrial and social expectations, and guidelines for a computer engineering program. Moreover, the pedagogical methods were implemented using modular design concepts for designing progressive laboratory experiences and educational modules. The assessment methods, for this work, were implemented through the use of pre- and post-tests to validate student learning process. Finally, the implemented methodology was validated against the old methodology through a student performance comparison. The results showed improvements in the students learning gains, test grades, and better overall student performance in the development of the laboratory practices.

The rest of the document is organized as follows: Chapter 2 presents a background of the different elements that are part of an embedded system course. Also, it introduces the Embedded System Design course (ESDC) currently taught at the University of Puerto Rico at Mayagüez (UPRM), and concepts related to modular design. Chapter 3 presents a literature review of works related to a course design, embedded system teaching methodologies, and initial efforts to apply a modular design approach in a course. Chapter 4 presents a discussion of the problem statement and hypothesis for this research. Chapter 5 shows the general and specific objectives defined for this work. Chapter 6 details the methodology followed for structuring the laboratory based on modular design aided by an Outcome-based education framework. In Chapter 7, obtained results for the students' groups under study are analyzed and detailed. The last chapters show the conclusions, contributions, and future work for this research.

Chapter 2
BACKGROUND

In this chapter, reviews of the ESD Course at UPRM and terms related to modular design are presented. The characteristics of the course and the laboratory role are explained. Also, the module, modular, and modularity concepts are defined to show how they are related to the development of embedded systems.

2.1 Elements in an Embedded System Course

According to The Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering by the IEEE/ACM (CGUDPCE) the embedded system area is recommended to be part of all computer engineering curricula due their expertise and application area [19]. Table 2.1 shows the body of knowledge recommended by the CGUDPCE for a computer engineering program in which embedded system consist of a total of 20 hours period.

For the embedded system area (CE-ESY), the CGUDPCE recommend a series of knowledge units that should be taught in order to ensure proficient knowledge in this area. The topics are divided into a minimum of a core material and elective units where the core material have specific hours and the elective units could be used as a complementary material for the class. At the same time, each knowledge unit is divided into a set of topics. Table 2.2 describe each knowledge unit to be part of an embedded system area with a summary of their most relevant topics.

The course must provide student with a laboratory experience that allows contact with real devices, system, and process. This experience is very important for

Table 2.1 : CE2004 Discipline Areas containing core material

Label	Major Knowledge Area	Hours
CE-ALG	Algorithms	30
CE-CAO	Computer Architecture and Organization	63
CE-CSE	Computer Systems Engineering	18
CE-CSG	Circuits and Signals	43
CE-DBS	Database Systems	5
CE-DIG	Digital Logic	57
CE-DSP	Digital Signal processing	17
CE-ELE	Electronics	40
CE-ESY	Embedded Systems	20
CE-HCI	Human-Computer Interaction	8
CE-NWK	Computer Networks	21
CE-OPS	Operating Systems	20
CE-PRF	Programming Fundamentals	39
CE-SPR	Social and Professional Issues	16
CE-SWE	Software Engineering	13
CE-VLS	VLSI Design and Fabrication	10
CE-DSC	Discrete Structures	33
CE-PRS	Probability and Statistics	33

the student' growth because this is the way in which they can design, implement, test, and document hardware and software in a practical manner. The CGUDPCE recommend laboratory practices should help students to improve their practical skills and expertise while learning the importance of work teams and technical staff [19].

2.2 Embedded System Design Course (ICOM4217)

The Electrical and Computer Engineering Department in the University of Puerto Rico at Mayagüez (UPRM) offers a 5-year program in Computer Engineering (CE). This program, accredited by ABET (Accreditation Board for Engineering and Technology) since 1990, is based on student learning outcomes [20] where the outcomes defined by ABET have to be achieved by the students at the end of their careers. The program has 167 credits, 15 of which are devoted to technical electives courses which allow students to delve in one of three areas of emphasis: Hardware and Embedded Systems, Computing Systems, and Communications and Signal Processing [21].

The Embedded System Design Course (ESDC) ICOM4217 is a major technical elective in the CE curriculum and a prerequisite for the Capstone Design course. Also,

Table 2.2 : CE-ESY knowledge units

Cod	Unit Name	Hours	Unit Topics
CE-ESY0	History and overview	1	Reason for studying embedded systems - contrast between an embedded system and other computer system
CE-ESY1	Embedded microcontrollers	6	Structure of a basic computer system - Basic I/O devices - Polled vs Interrupt - Interrupts structures
CE-ESY2	Embedded programs	3	Program translation process - Representation of programs - Fundamental concepts of assembly language
CE-ESY3	Real-Time operating systems	3	Scheduling policies - Priority inversion - Message passing vs shared memory
CE-ESY4	Low-power computing	2	Source of energy consumption - power consumption with multiple processors - system level power management
CE-ESY5	Reliable system design	2	Transient vs permanent failures - Fault tolerant techniques - famous failures of embedded computers
CE-ESY6	Design Methodologies	3	Multi-person design projects - Design reviews - Change management
CE-ESY7	Tool support	Elective	Compiler and programming environments - power analysis - logic analyzers - software and management tools
CE-ESY8	Embedded multiprocessors	Elective	Importance of multiprocessors - Hardware/software partitioning
CE-ESY9	Networked embedded system	Elective	Why networked embedded system - The OSI reference model - Basic principles of internet protocol
CE-ESY10	Interfacing and mixed-signal system	Elective	Digital to analog conversion - Analog to digital conversion - Digital processing

it is a required course for those who decide to emphasize in Hardware & Embedded Systems. While in the Embedded System Introduction (prerequisite for ESDC) students learns about microprocessor architecture, organization, and operation, in the ESDC students learn how to interface and manage I/O ports, interrupts, timers,

pulse width modulation (PWM), motors, analog-to-digital (ADC) and digital-to-analog (DAC) converters, and memories to develop applications based on embedded systems. The topics are covered through class lectures (3 hours per week) and their implementation and practical knowledge are covered through weekly laboratory experiences within the ESDC laboratory (2 hours). The laboratory objectives are covered in two steps: (1) Guided Exercise section that guide students with step-by-step on the fundamental experiment tasks and (2) a Homework section that provides a complementary task to deepen in the laboratory topic teaches in the week. The laboratory also works as a complementary learning tool that helps students develop fundamental modules that facilitate the construction of their course projects.

The ESD course project consists in the development of a functional embedded system prototype where the prototype is designed with the objective to satisfy a client necessity (research group, organization, or external person to the class). Each project must be composed of the following 4 elements: (1) communications, (2) user interface, (3) control scheme, and a (4) microprocessor unit. The project begins with the idea that students bring to the class and therefore with the professor and teaching assistant (T.A.) meetings the idea is transformed into a correct proposal with defined requirements. Then, during the class period, students work in the development of their prototypes while several progress reports are required to follow students progress.

2.3 Definitions: Module, Modularity, and Modularization

During the design process of a system, equipment, or device there are several methods that define the structure, constructions, and functionality of a product. One of the methods corresponds to the modular design. A modular design approach allows for developing a product from a set of different smaller modules where each module represents a functionality component of the entire system. Each module in a modular design can be easily replaced by another module with similar characteristics

without affecting the final functionality [22]. This characteristic is exemplified in Figure 2.1 where a Power Supply module can be replaced by another while keeping the system functionality intact. To help the comprehension of the modular design process, we adopt the following definitions [23–25]:

Modularization: is the action of separating or organizing the parts of a product or process in specific functions or tasks.

Module: A module is a component formed by one or more elements, which has one or several specific functions and can work independently or with other elements or modules to construct a system.

Modularity: is related to a product feature, where each module achieves one or few main functions, it provides some advantages such as flexibility and reduction of the complexity. Additionally, modularity allows for replacing a module without affecting the system functionality.

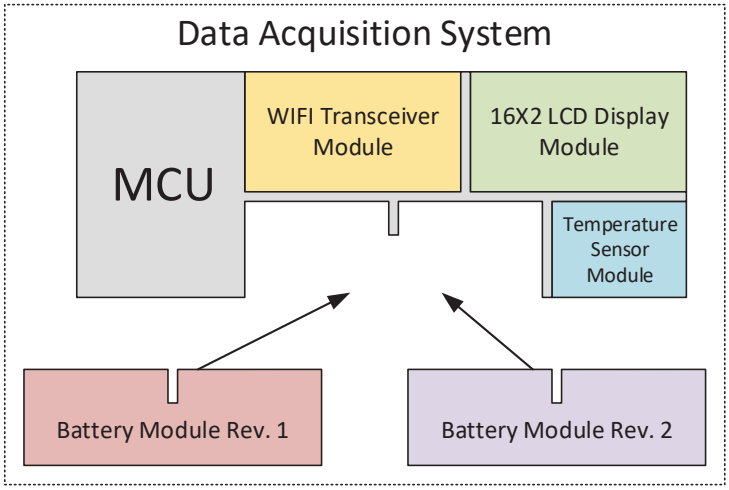


Figure 2.1 : Example of modular design

Chapter 3
PREVIOUS WORK

In this Chapter, a summary of relevant publications found in course design, embedded system teaching approaches, and modular design in embedded systems are presented and discussed. Important approaches in how to design or redesign a course were analyzed to identify their main concepts. Also, strengths and weaknesses of different embedded systems teaching methodologies, and similar courses in other universities were studied to learn about their class and laboratory structure.

3.1 Curriculum and Course Design

Duffy *et al.* proposed a learning methodology based on a Learning-Centered Approach (LCA) focus in the course syllabus [26]. The authors explained that a syllabus is one of the most important parts when a professor creates a course because it defines all the content that has to be taught and learned. The syllabus has to include information about topics, define course limits, all the material needed by students, and the assessment methods to be applied. Although a syllabus has all the information needed for a course, the authors proposed the creation of a syllabus based only in what the professor considers is necessary material for students. Also, the methodology does not necessarily take into account the current necessities for the industry at the moment of the curriculum creation because it is not required.

Fink explained that the most common methodology to structure a class is the content-centered approach also named List Of Topics (LOT) [27]. In this approach, the professor creates a list of topics that he considers are the more suitable to be

learned by students. Topics must have relation to the class and are chosen based only on the professor's criteria. Then, the professor establishes the time for each topic and decides the form in which each will be evaluated. A drawback of this methodology is that the professor is only focused in completing the topics previously established without considering the students' learning process. Also, this methodology did not implement a feedback process to know the students' perception about the course.

Fink also proposed a methodology called Integrated Course Design (ICD) [27,28]. This method consists of four main interrelated elements: (1) situational factors related with the number of students in the course and their prior knowledge, (2) learning goals that define the knowledge and abilities the student must learn and retain after they take the course, (3) feedback and assessment, used to validate if students achieve learning goals and (4) teaching/learning activities, to ensure students' learning process. Elements had to be developed in a specific order and each must be taken into account for the design of the next element. As this methodology defines the assessment methods before the learning activities, the feedback is only about the topics learned by students excluding the teaching methods selected for the class. For this reason, the professor does not receive any feedback about the effectiveness of the methods selected and can not determine whether the methods are appropriate for the class or not.

Pellegrino proposed a Curriculum-Instruction-Assessment Triad (CIAT) in which the three elements in the triad have to be aligned to ensure a proper education process [29]. Curriculum refers to the knowledge and skills to be learned by students. Instructions refer to methods how the topics shall be taught; and assessment refers to the measurement of the outcomes achieved by students. The author explains that the three elements in the triad must have the same objectives and reinforce each other. Otherwise, the assessment could be carried out in an incorrect manner and the instructions could be inadequate and ineffective.

Based on the work by Pellegrino, Streveler *et al.* proposed an approach called Outcome-Based Education (OBE) in which the content, assessment, and pedagogy must be aligned to achieve the requirements (outcomes) desired by the students [30]. The authors explained step-by-step how these elements must be aligned to ensure a correct and complete student's learning process. The process begins establishing the desired profile for students and continues with the determination of the methodology to teach students according to the course characteristics. Finally, the process ends with the methods to assess students' progress and acquired level of knowledge. This work follows the "Backward Design Approach" proposed by Wiggins and McTighe [31], in which the process to create a course begins in the *end* defining the learning goals and continues *backward* to create the lesson and methods to achieve those goals. The authors also used the "How People Learn" framework proposed by Bransford *et al.* [32] that explain how individuals build their knowledge in an education institution and during their life.

In this section, we can observe that several efforts have been performed to design a course or curriculum. However, some methodologies present significant limitations that have been identified. Table 3.1 shows relevant aspects of the process to design a course that will be addressed in this research. To ensure a well-structured course, the three fundamental components of a course (Topics, pedagogy, and the assessment methods) must be linked. We can see that some of the reviewed approaches did not meet this requirement. Also, the majority of the approaches did not measure students' progress in terms of outcomes to be achieved or did not take into account the current industrial and social necessities to create the desired student profile. But, one approach did take into account all the aspects mentioned before. The approach was proposed by Streveler *et al.* and consist in an outcomes-based education [30]. For this reason, this approach will be used as a guide to structure the laboratory for the ESDC.

Table 3.1 : Summary of reviewed approaches for course design

Approach	Outcomes	Learning goals	Desired characteristics	Content and assessments	Pedagogy	Link C&A&P
LCA [26]	No	Yes	Yes	Yes	No	No
LOT [28]	No	Yes	No	Yes	No	No
ICD [27]	No	Yes	No	Yes	Yes	Yes
CIAT [29]	Yes	No	No	Yes	Yes	Yes
OBE [30]	Yes	Yes	Yes	Yes	Yes	Yes

Note: C&A&P means Content&Assessment&Pedagogy

3.2 Engineering Education

3.2.1 Embedded Systems Education

During the past years, various methodologies have been developed to enhance the learning process in embedded system. These methodologies used a great variety of resources to attract students and to motivate them to learn about embedded systems inspired from textbooks to daily life issues.

Torroja *et al.* proposed a Scale Model methodology in which, students developed the code and the necessary hardware to control a scale garage and a car wash model using the Motorola 68HC11 [33]. The fundamentals topic were taught through lecture sections before the development of the scale models. These models contained sensors and actuators that allowed for replicating the complete functionality of real systems and their associated problems, like noise, bad connections, and sensor signal compatibilities. The authors explained that this methodology generated strong motivation among students because they worked in scale real systems and had to solve real system problems. Although students rated the method as very positive (based on a survey developed at the end of the course) it had limitations because only one copy of each scale model existed and the price for its maintenance was high. It was also difficult to use the scale models in large groups of students (more than 20).

Doug *et al.* proposed a multidisciplinary cooperative Problem-Based learning approach also using the Motorola 68HC11 [9]. This methodology exposed students

to a problem situation that they had not encountered before to conduct the learning activities and, therefore, improve their technical skills. The class had a lecture session and a session on how to use CAD-Tools that ran in parallel with the problem session causing timing constraints. Also, the course's evaluation survey was conducted before the completion of the course, providing an incomplete assessment about the course perception by the students as the last part of the class was not evaluated. Finally, all projects ran with the same processor, and therefore students did not learn about other architectures and the process to select a processor according to the specifications of the problem to be solved.

Bruce *et al.* used a problem-based approach with progressive design experiences to teach embedded systems [10]. In this methodology, students had to develop small electronic systems in each laboratory experiment. Each subsystem built in the previous laboratory was required to be built in the new subsystem. The proposed problem was designed to incorporate industrial requirements defined by the authors and the prototype was constructed bit by bit with each design experiment. All the projects and practices were based on an 8-bit microcontroller. This approach limited the skills developed by the students as they did not work in the process to conceptualize, define objectives, and scopes for their designs.

Ordinez and Alimenti proposed a constructivist and problem-based approach to conduct their course [11]. In this approach, students develop all their knowledge based on a problem to be solved that had to be attractive and included a wide range of topics. The main drawback of this approach was that the problem did not cover a basic knowledge in embedded systems for all the cases because the problem in each semester was not the same. For this reason, the skills mastered by students in each semester were not the same, generating groups of students each term with inconsistent levels of preparation.

Nooshabadi and Garside proposed a teaching method based on an International Collaborative project with universities in Australia and the United Kingdom [34]. This approach used Laboratory Experiences that replicated industrial design practices. The authors expressed that the best way to teach embedded systems was using constant development practices. These practices also included the use of devices such as mobile phones, web phones, televisions, digital cameras, and personal digital assistants. All the practices were conducted using a DSLMU (Digital System Laboratory at Manchester University) hardware development board designed by the authors. In this approach, students worked with a predefined hardware platform and software (IDE and Compilers), documentation, and scripts provided by the professors. This strategy limited the students' skill in areas or topics such as MCU and compiler selection, and design process.

Gonzalez *et al.* used Video-Game devices to enhance the learning process, exploiting the connection students developed with these type of devices [15]. The approach used a Nintendo DS (NDS) as teaching platform, as it is built with a multiprocessor architecture. The authors use development tools such as GNU compilers, VisualBoy advance simulator (VBA-M), C libraries and insight debugger to carry out the laboratory experiences. All laboratories consisted of two sections: one introductory and another with exercises. The main drawback of this approach was that the authors focused only on the software of the embedded systems neglecting the knowledge about hardware design and implementation.

Münz *et al.* proposed the use of Educational Games as a learning tool to teach the theoretical material to students [16]. This strategy only focused on solving practical problems. To achieve this objective, the authors developed a game to teach the principles of automatic control. The game consisted on a submarine that had to follow a given trajectory controlled by the student or by automatic controllers. Students had to develop the controller for the system using the previous knowledge

learned in class. This approach only taught control in a theoretical way and did not provide the knowledge to implement a physical controller. Therefore, students did not learn concepts like signal conditioning, MCU implementations, timers, and interrupts, among others.

Kodoma *et al.* proposed a Remote virtual laboratory approach [17]. The authors developed a remote control system for an embedded board to facilitate the self-learning process at home. The board consisted of an MPU and CPLD as processors with common electronic elements. The system could be accessed at any time, but each session was only five minutes long, making it difficult for the development and debugging of a large task. In addition, students could do multiple reservations but it resulted in a tedious process. Furthermore, it was difficult to check high-speed process and clicking input sequential signals on the web page.

Büchner and Jaschke developed a virtual-machine-based environment laboratory to foster the preparation outside of the laboratory [18]. Their approach consists of the virtualization of different applications related to the development of embedded systems on a USB flash drive. The installed software allowed for programming different devices like FPGAs, ATXmega, android devices, and provided access to a reference manual for hardware elements and simulators. Once students finished their programs, they could go to the laboratory to test their solutions in a real hardware platform. To get students motivated, all applications and projects were based on home automation. Although, this model facilitated working outside the laboratory, not all the devices had simulation tools limiting the debugging and testing process of the code. Also, they did not develop or improve their abilities to interface hardware components because the hardware used was on a pre-built platform.

Lee *et al.* implemented a project-based laboratory with industry support to improve the abilities of the students in specific peripherals and topics [12]. The project consisted of developing a line-following robot using a quadratic interpolation

technique to predict the line position. The hardware platform and C-compilers were supported by the local branch Microchip Inc., and consisted of a vehicle composed mainly by a dsPIC4011 microcontroller. The class was conducted through two lecture hours and one laboratory hour per week. In each class, students had to answer questions related to their work and turn-in reports about their work. Since the hardware structure was provided by a company, students only focused their work on programming the robot and calibrating the sensors, while keeping the hardware design skills in a second plane.

Kumar *et al.* used an FPGA-based platform with a Project-Based methodology to teach students in two different courses [13]. The project for the first course was related to real-time embedded systems and consisted of developing a five versus five soccer system on multiple FPGA. For the second course, the project was devoted to the hardware aspects of embedded systems. The main task consisted of decrypting a block encrypted image, accelerated through a custom co-processor. In both projects students had to learn topics related to real-time performance, timing behavior, HDL, and design methodologies for embedded systems from course lectures. This approach only focused on the programming aspect of the FPGA, using the resources attached to the board and did not teach how to interface electronic components to the board.

Couvertier *et al.* implemented a approach in which students propose a project to be solved [14]. All projects and hardware platform are freely selected by students but the majority of them used the MSP-FET430P120 as hardware platform. In this methodology, students learned the basic principles of MSP and microcontroller in a course where learning was conducted in parallel with the project. Some laboratory practices are also provided to enhance the students skills in the embedded system area. Although, this model provided some laboratory practices, the course did not have an appropriate laboratory section with predefined hours and a well established laboratory manual to conduct the laboratory practices.

In this section, we could observe that some efforts have been performed to enhance the learning process in embedded systems. However, significant limitations have been identified. Table 3.2 shows the relevant aspects of embedded systems teaching approaches addressed in this research.

To ensure a proper education in the design of embedded systems, it is necessary that students pass trough all the stages of a product development, where the MCU to be used is selected according to the application characteristics. This conditioning is not fulfilled by the majority of the approaches because they use predefined hardware platforms. Also, some of the approaches are applied to the class in the form of a final project but are not applied to structure the laboratory content. Although authors describe how they designed and implemented their approaches, they do not describe or develop formal assessment methods that allow for measuring the level progress by students and their acquired skills. Finally, neither of the authors developed assessment methods to verify the effectiveness of their implemented approaches. All the aspects mentioned before reveal important challenges that we will need to address.

Table 3.2 : Summary of reviewed teaching embedded systems approaches

Approach	**Author**	**Applied to**		**Predefined hardware platform**	**Final project**	**Formal Assessment**
		Laboratory section	**Lecture section**			
Scale Model [33]	Torroja	Yes	Yes	Yes	Yes	No
Problem-Based [9]	Doug	No	Yes	Yes	Yes	No
Problem-Based [10]	Bruce	Yes	No	Yes	Yes	No
Problem-Based [11]	Ordinez	Yes	No	Yes	Yes	No
Laboratory [34]	Nooshabadi	Yes	No	Yes	No	No
Games [15]	Gonzales	No	Yes	Yes	Yes	No
Games [16]	Münz	Yes	Yes	N/A	No	No
Virtual-Labs [17]	Kodoma	Yes	No	Yes	No	No
Virtual-Labs [18]	Büchner	Yes	No	Yes	No	No
Project-Based [12]	Lee	No	Yes	Yes	Yes	No
Project-Based [13]	Kumar	No	Yes	Yes	Yes	No
Project-Based [14]	Couvertier	Yes	Yes	No	Yes	No

Note: N/A means Not applied

3.2.2 Representatives Embedded System Teaching Approaches

Each university and program offer their courses according to their characteristics and desired approach. Furthermore, it is important to know about their embedded systems courses. Two top-ten universities in computer engineering and two UPRM-equivalent universities in terms of student population and academic program were studied.

The Massachusetts Institute of Technology (MIT) has The Microcomputer Project Laboratory (6.115) [35] as a course to introduce students in the analysis and design of embedded systems. The course is a laboratory required in the Electrical Engineering and Computer Science department. The laboratory emphasizes in the construction of electronics systems. It also introduces the usage of basic electronic tools, and teaches fundamental principles about microcontroller peripherals such as A/D converters, communication schemes, motors, and power electronic converters, among others. The laboratory is conducted through weekly experiments with a final project selected by the students. The project proposal is presented during the week 10 of classes and the final product 5 weeks later. The main microcontroller used is the Intel 8051. This laboratory has as prerequisites the courses: 6.002 Circuits and electronics, 6.003 Signals and Systems, and 6.004 Computation Structures.

The University of California at Berkeley has the course Introduction to Embedded Systems (EECS149) [36] where they introduce students to the design and analysis of computational systems that interact with physical processes. The course has an emphasis on finite state machines, basic control systems, physical world interfaces, mapping, and distributed embedded systems. The course is conducted through a weekly lecture session (3 hours), a weekly laboratory (3 hours), and a team project with a final paper and poster presentation. Each laboratory has a pre-lab work, a laboratory demonstration, and a laboratory report. A project proposal is presented during the week 8 of classes and the final presentation is due 5 weeks later. Students

are suggested to use the FRDM-K64F or the NUCLEO-F411RE platform as embedded boards to develop their projects. This course has as prerequisites the courses: (EECS 16B) Designing information devices and systems I, (CS 61C) Great ideas in computer architecture, and (EECS 70) Discrete Math & Probability.

The University of Texas at El Paso (UTEP) offers and Embedded Systems (EE3376) [37] course that covers the fundamental aspects of designing embedded systems. The course is part of the computer engineer program and teaches how to use the main peripherals that can be found in a microcontroller system. The course content includes interrupts, peripherals interfacing, timers, Analog-to-Digital Converter (ADC), and program structures, among others. The course combines class lectures and laboratory sessions (EE3176) in which a final project has to be developed as a part of the laboratory. Each laboratory has one week to be developed and uses the MSP430 launchpad as a hardware platform. Exams and quizzes are carried-out in the course and laboratory to assess the level of knowledge acquired by the students. The final project is defined by the professor and has to be developed in the last 3 weeks of the semester. This course has a prerequisite the course: (EE2372) Software Design I.

The University of New Mexico offers a Microprocessors course (ECE344L) [38] to teach students the topics related with the architecture and use of embedded systems. The course is part of the computer engineering program and is focused mainly on the MIPS architecture, although all covered topics can be applied to other microprocessor systems as well. The course covers fundamental topics like overview of computer systems, information representation, instruction sets, memories, interrupts, and I/O methods. The course is conducted in the form of class lectures (3 hours) per week and laboratory session with a class/lab final project. The project needs to be developed from scratch and has to integrate hardware and software components. Also, two midterms and a final exam are given to assess the level of knowledge gained by the

students. This course has as prerequisites the courses: (ECE 206L) Instrumentation, (ECE 238L) Computer Logic Design, and (ECE 321L) Electronics I.

3.2.3 Modular Design Applied to Embedded Systems

Kamal [22] and Zurawski [39] agreed that the modular design is one of the techniques currently used to design and construct embedded systems because with this technique, an entire system is constructed from small parts (modules), where each module represents a feature. Also, they argument that the embedded systems constructed in the form of modular designs allow for the reuse of elements (hardware or software components from other projects), risk reduction (design is based on tested components), migration (change between new or older hardware/software versions), and cost reduction through the use of standardize components.

Valvano [40] explained that modular design can be applied to develop software components for an embedded system. This concept is called modular programming and it refers a program structure that allows for using software modules from multiple locations, and divide a highly complex task into smaller less complicated tasks. It also includes the possibility of using software modules developed for one machine in another with different I/O ports. With this programming style, the system can be easily tested because each module can be independently debugged.

Meng *et al.* used a modular approach for constructing a generic architecture based on embedded systems for miniature mobile robots [41]. The authors constructed an architecture called SMARboot with different modules in which each module could be replaced by another with the objective of improving the capabilities and function to be executed. The modules included a high-performance microprocessor, reconfigurable hardware, wireless communication, sensors, and actuators. The main processor was composed by an ARM7TDMI microprocessor and an FPGA module. The communication was carried out through synchronous/asynchronous serial and

CANbus protocols. Also, the software was designed to be modular on a real-time operating system.

Li used a modular approach to teaching embedded systems [42]. He only focused the approach on designing a modular microcontroller training kit to improve students' learning process. The kit was composed by an ATmega127 microcontroller with different modules (USB interface, I/O, expansion ports, and user interface) and could be used by students in their projects. Özgü developed a modular embedded system to teach students in mechatronic engineering [43]. The system was composed by four modules: USB, AIDO (Analog and digital I/O), motor, and LCD boards. All communication processes were carried out through SPI protocol controlled by PIC microcontrollers. Both authors were concentrated on incorporating the modular approach to design an electronic platform but not on how to design a course or teaching methodology based on modular design.

Hu *et al.* developed a courseware based on modular design to teach embedded system design [44]. The authors focused their work on developing and organizing different teaching modules, where each module was based on a specific topic and included all the related material (lectures, laboratory experiments, and assignments). The courseware constructed in this form allowed for the professor to adopt all the course material or integrate the modules that he considered necessary in his current course. The courseware also used a predefined portable hardware platform based on the C8051F005DK microcontroller to conduct the experiments. Although the authors used a modular approach to design the courseware, they did not incorporate modular design concepts in the course's topics and laboratory experiments.

In this section, we analyzed several efforts that have been performed to enhance students learning processes and skills usign modular design. The authors introduced the modular approach in different ways, some used it to design and construct modular hardware platforms and others to design courseware material. Although they

incorporated the approach in their courses, neither of the authors incorporated the modular design in their class topics and neither of them used the modular approach as a base to design the course laboratory experiments' content. This reveals an important challenge that we will need to address.

Chapter 4
PROBLEM STATEMENT AND HYPOTHESIS

4.1 Problem Statement

The average American interacts with al least 100 embedded systems per day [3] giving us an idea of their relevance. Furthermore, it is important that universities provide students with proficient knowledge in embedded systems design to meet industrial and societal expectations [45–47].

The problem addressed in this research was that of developing a methodology to improve students' design skills in the embedded system area while also taking advantage of contemporary modular technologies. Methodologies used in similar courses in other universities implement different tools to motivate students and help them in their learning process. Furthermore, most of these methodologies use prebuilt platforms or modules that do not allow students to follow through the design process.

Our research questions was: What is the effectiveness of using a class/laboratory structure based on a backward design with a learning methodology based on modular design? How can these be used to ensure strengthening and improving the skills acquired by students in an embedded system design laboratory?

4.2 Hypothesis

This work was developed under the hypothesis that *an OBE-based laboratory with a teaching methodology based on a modular design will improve student learning by incorporating contemporary design skills aided by progressive lab experiments and*

module design strategies. These should provide a focus on modularity and how to build embedded systems from a basic idea (necessity), passing through design considerations (module and system design) to the final conception (functional prototype).

Chapter 5
OBJECTIVES

This section describes the objectives that were formulated for this work. First, the general objective is presented and then the specific objectives which are composed of educational and technical objectives.

5.1 General Objective

The main goal of this research was to design and implement a backward design teaching methodology to teach students in the Embedded System Design Laboratory using a modular design approach.

5.2 Educational Objectives

1. To establish and determine a laboratory content based on a predetermined student profile.
2. To design and implement a laboratory teaching methodology in embedded systems.
3. To develop assessment methods to verify student's learning progress and proposed teaching methodology.
4. To develop a laboratory manual to improve students' design skills in an embedded system design laboratory.

5.3 Technical Objectives

1. To recommend and evaluate suitable electronic modules that can support the educational activities recommended as part of the teaching methodology derived from this work.

2. To design and document student handouts, tutorials, and reference guides about the necessary laboratory equipment and educational electronic modules to be made as part of the proposed teaching activities.

Chapter 6
LABORATORY REDESIGN

This Chapter presents the research methodology used in this thesis. It was based on a modular design approach and guided by an educational framework referred to as the "outcome based-education". In this work the laboratory content, the methodology, and the assessment methods were designed and integrated to impact the student learning experience.

The first part of this chapter introduces the developed laboratory content based on the desired student profile and the topics required in the course. Next, we explain the methodology applied based on a modular approach and the assessment methods developed to validate the proposed methodology. In the last section, complementary materials used to aid in the laboratory learning process are described.

6.1 Laboratory Content Design

Streveler *et al.* [30] developed a pedagogical approach called Outcome Based-Education which search for aligning course content, pedagogical methods, and assessment methods in an education process. In this approach it is strongly recommended to begin with the desired student profile, because it works as a base for the establishment of the laboratory content and learning objectives. Also, the content design process, for an applied laboratory, presented by Patarroyo *et al.* [48] was use as a reference.

6.1.1 Desired Student Profile

To define the course content, Wiggins and McTighe [31] explained that it was crucial to know what we want about our students and how we want our students to be. To accomplish these objectives it is important to define the desired profile for our students in embedded systems. For our student's desired profile, we took as a base The Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering (CE) by IEEE/ACM in which the characteristics of a computer engineer graduate are specified [19]. This curriculum divided the characteristics for a CE graduate in four principal aspects: (1) distinctions, (2) professionalism, (3) ability to design, and (4) breadth of knowledge.

Taking the division mentioned before and the course characteristics that are focused on the design of embedded systems, our students are expected to have [19]:

Distinction: possess the ability to design embedded systems that include hardware and software components that allow solving engineering problems.

Professionalism: students must be responsible and must take into account the effects of their designs on the surrounding community. Also, they have to be conscious of the ethical and social responsibilities related to property rights, security, and privacy.

Ability to design: students must have abilities for the development of new devices and products based on embedded systems. They must have the capability of identifying a necessity or problem and produce a reliable solution based on desired specifications.

Breadth of knowledge: students are expected to know about embedded systems organization and architecture, algorithms, and programming skills. Also, they are also expected to have:

- A system level perspective that allows them to appreciate the concept of an embedded system and their hardware and software components. They should understand the embedded system applications in the society and industry.
- Design experiences that allow them to develop an entire system or prototype as an embedded systems application. The design could address a real problem or yield a functional product.
- Ability to use different computer-based and microcontroller-based tools for embedded systems analysis and design.
- Communication skills to express their work in a written, oral, or graphical way. Also, students must develop critical analysis skills to provide feedback in that format and team-work abilities to contribute in an active way to the design process of a system.

Also, according to Henzinger and Sifakis [45] industry requires that a person with knowledge on embedded systems possess abilities to work in problems related to the design of software and hardware components. They must also have, the capability to produce novel ideas that would result in a final embedded system prototype/product. Jing *et al.* [49] expressed that the knowledge required by the industry in an embedded developer can de divided into three categories: the first referred to technology elements such as compilers, programming, measurement and control, user interface, and multimedia. The second is related to the development and management skills to be used to develop systems and processes. Those abilities include programming, system design, and testing skills. In the management skills area, industry requires abilities such as time, cost, and risk management. Finally, the third category is referred as personal abilities such as leadership and teamwork because these skills have an influence in the success of a project.

6.1.2 Laboratory Content

In this work, the laboratory content was developed to improve the design abilities of students through a set of laboratory experiments. Also, the content was established to help students accomplish outcomes for the Embedded System Design course. Table 6.1 presents the class outcomes selected to be accomplished in the laboratory aligned with accreditation criteria.

Table 6.1 : ICOM 4217 course outcomes [1]

Activity	***Program Outcomes***
1. Students conduct laboratory work to implement a working prototype of their project.	b
2. Students perform a project of an original idea proposed by their group. Project implementation is a course requirement.	b
4. Each group must show originality in their work, the procedure of partitioning a complex problem into parts, and combining peer work into the final solution.	d
6. The project idea, along with a plausible procedure, is submitted by the working group as a proposal.	e
13. Students must be able to program a microcontroller using a development environment that includes debuggers, editing tools, and compilers, among others. The microcontrollers used in class are considered state of the art.	k

Taking as a reference the procedure presented by Streveler *et al.*, [30] in the OBE, the laboratory content was re-designed along these three guidelines: (1) desired outcomes, (2) curricular priorities, and (3) learning objectives. The first two steps worked as a baseline for Step 3.

Desired Outcomes

According to Streveler *et al.*, [30] the first step in the OBE model is to determine the desired laboratory outcomes by answering the three following questions: (1) What do we want students to know? (2) What do we want students to be able to do? and (3) Who do we want students to be?. The answer to these questions gave us the values

and the attitudes shared by students and a guideline for our learning objectives, teaching methodology, and assessments.

The laboratory desired outcomes shown in Table 6.2 were identified and defined according to the course topics and characteristics and general objective. The outcomes were concentrated in enhancing the practical skills of students for embedded systems' designing, assembling, testing, and constructing. The outcomes were written based on the cognitive domain of Bloom's taxonomy which involvs the knowledge and the development of intellectual abilities and skills [50].

Table 6.2 : Student Outcome Analysis

What do we want students to know? (A) Students should:

- A1: Identify the uses and advantages of embedded systems.
- A2: Identify the main components of an embedded system.
- A3: Know the steps in the hardware design and building process for an embedded system prototype.
- A4: Know the software structures to create functional programs to govern the operation of electronics modules or embedded systems prototypes.
- A5: Recognize the social and industrial necessities or problems in which an embedded system can be implemented.

What do we want students to be able to do? (B) Students should be able to:

- B1: Configure the main components of an embedded system.
- B2: Use Assembly and C language to develop the software for a project based on embedded systems.
- B3: Design electronics modules that will help in the construction of an embedded system prototype.
- B4: Interface different electronics modules to a main controller.
- B5: Design a product, device, or prototype based on embedded systems.

What do we want students to be? (C) Students should be:

- C1: Self-learners.
- C2: Curious persons about how an embedded systems works and its applications.
- C3: Active members in a workgroup.

Curricular Priorities

Once student outcomes were determined, the next step consisted of defining the curricular priorities. These outcomes gave us the basic idea in terms of student necessities that have to be covered during the laboratory. According to Streveler *et. al.*, the curricular priorities have to be organized in three levels of content: Enduring Understanding, Important Elements to Know&Do, and Worth Being Familiar With [30].

The curricular priorities shown in Table 6.3 were established taking into account the knowledge, skills, and the abilities that students are expected to learn by the end of the laboratory.

Table 6.3 : Curricular Priorities of the Embedded System Design Course

Enduring Understanding

- **Identify** the main component and peripherals of an embedded system.
- **Known** the steps for designing prototypes based on embedded systems.
- **Implement** an embedded system for solving a necessity or project.

Important Elements to Know&Do

- How to **structure** and present project proposals in which an embedded systems prototype should be used.
- How to **design** an embedded system prototype based on microcontrollers and microprocessors.
- How to **identify** the electronics modules needed for constructing an embedded system.
- How to **interface** different elements that are part of an embedded system prototype.
- How to employ datasheets to **design** electronics systems modules.

Worth Being Familiar With

- Design process of an electronic prototype.
- Embedded systems development tools.
- Hardware design, verification, and integration for an embedded system.
- Develop flowcharts and Block diagrams for an embedded system based prototype.
- Microcontroller and microprocessor programming.

Learning Objectives

The final step consisted of determining the learning objectives that would guide students in their learning process. These objectives were established taking into account the student outcomes analysis and the curricular priorities previously defined. The learning objectives were established based on the cognitive domain of Bloom's Taxonomy which divides the cognitive knowledge into six levels or categories (knowledge, comprehension, application, analysis, evaluation, and synthesis) that can be thought as degrees of difficulties [50]. Therefore, before transitioning to a new, higher level, the preceding levels have to be completed. Based on our analysis, at the end of the laboratory, students are expected to complete the following learning objectives:

1. **Identifying** the main components that form an embedded system.
2. **Identifying** the steps in the design process of an embedded system prototype.
3. **Explaining** the basic functionality of the main peripheral modules incorporated in a microcontroller chip.
4. **Interfacing** external electronic elements and modules to a microcontroller to manage different types of sensors, actuators, graphical user interfaces, and data.
5. **Sketching** block diagrams and schematics that allows understanding the architecture and electronic structure of an embedded system or electronic module.
6. **Creating** software plans and flow diagrams that allow for planning the software of an embedded system application.
7. **Integrating** the hardware and software components that make-up an embedded system prototype.
8. **Designing** a functional embedded system-based prototype.
9. **Testing** the functionality of an embedded system prototype.

Table 6.4 shows the relation of each learning objective with each cognitive level in the Bloom's taxonomy.

Table 6.4 : Learning objectives related to levels of Bloom's taxonomy

Cognitive Level	L. Obj. 1	L. Obj. 2	L. Obj. 3	L. Obj. 4	L. Obj. 5	L. Obj. 6	L. Obj. 7	L. Obj. 8	L. Obj. 9
Knowledge	•	•							
Comprehension			•						
Application				•					
Analysis					•				
Synthesis						•	•	•	
Evaluation									•

6.2 Methodology Implementation

According to Streveler *et al.* the methodology refers to the methods or approaches to be used in the teaching process [30]. The authors explained that the methodology must take into account the characteristics of the course/laboratory and desired student profile. It should also select the appropriate teaching mechanisms that could positively impact the student learning process. A laboratory teaching methodology was implemented based on a modular design approach in conjunction with the use of progressive laboratory experiments and module design.

6.2.1 Laboratory Teaching Methodology

To improve the student's abilities and design skills of embedded systems, a modular design approach was selected to be implemented as a part of the laboratory teaching methodology. An experiment structure, to guide the students in the fulfilment of their activities, was designed to ensure a proper student learning independently of the type of learner as explained later.

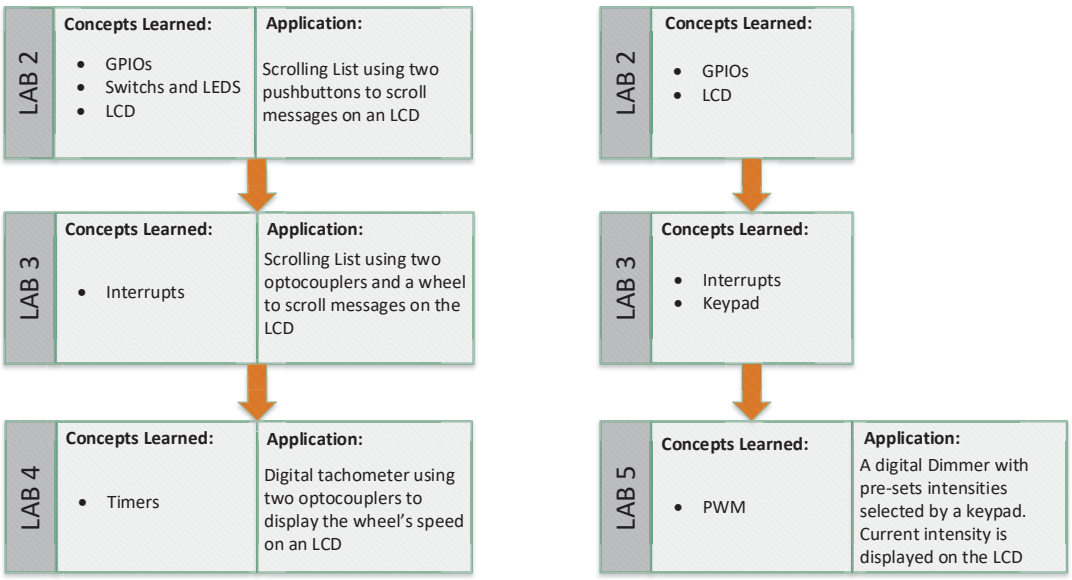
Modular Design Approach

A modular design technique, as mentioned in Section 2.3, allows for the development of a product from a set of different smaller modules where each module

represents a functional component of an entire system. This approach was implemented in the embedded system design laboratory through a set of guided laboratory experiences, incorporated in a structured laboratory manual, and a set of educational modules.

With the objective to improve student's design techniques, the modular approach was implemented in the laboratory experiments through the development of a set of small circuits and short programs that allowed students to acquire a base knowledge about their MCU's internal peripherals. Some experiences included progressive designs that required previously designed and tested circuits (in past experiment's sections) for the fulfillment of a new design. Figure 6.1 shows a progressive design example in which one system was built in three steps and the other system required the abilities learned and circuits tested in past experiments. This process provided students with the knowledge to design and construct their own embedded system prototype from incremental steps and small functional modules. Furthermore, a structured laboratory manual containing the experiments developed for the course was designed to provide the student with a step-by-step guide. This manual also provided fundamental theoretical background of each experiment and a description of each laboratory activity.

As part of the proposed approach implementation, a set of educational modules were developed to help students in their learning process (see Section 6.2.3 for the complete description of the modules developed). These modules were designed taking in consideration the different topics discussed in each experiment. Their design was included as complementary material to the laboratory manual and experiment. Each designed circuit module is composed of a set of components placed on a printed circuit board (PCB) where each PCB could work as a single experimental module or combined with other modules to provide the desired functionality. These modules were designed to work with either 3.3V or 5V microcontrollers, facilitating their usage with



(a) Digital Tachometer (b) Digital Dimmer

Figure 6.1 : Progressive Design exemplification

a wide range of MCUs and their incorporation in different students' class projects. Modules can then be combined with the target microcontroller unit (MCU) to develop embedded applications. These modules also provided students with an example of how electronic modules are structured including functional diagrams, schematics, board design, and software usage guidelines. Even though, the modules were designed based on the circuits provided in the manual, they could or could not be used in the development of each experiment and their usage depends on the instructor's criteria.

Laboratory Experiment Structure

According to Felder and Silverman [51] there are different types of learners (sensing, intuitive, visual, verbal, active, reflective, sequential, and global) that represent the different ways in which a student could learn or understand a topic. Based on these types of learners, the laboratory experiences were designed and implemented to provide a good understanding of each topic, independently of the type of learner.

Each experiment consists of four sections: (1) objectives, (2) lecture, (3) basic exercises, and (4) complementary tasks. Sections 2, 3, and 4 possess different types of elements that facilitate the student's understanding process, such as flow diagrams, block diagrams, and pseudo-codes, among others.

The objective section provides an explanation of what students should do and learn in the experiment while the lecture section provides a short review about the topics treated in the experiment.

The guided exercises section provides students with a step-by-step explanation on how they develop the proposed activity. Finally, the complementary tasks give students an implementation task with a set of predefined rubrics.

The implemented laboratory experiences covered each learning style through the activities described in Table 6.5 .

Table 6.5 : Type of learner vs laboratory experience sections

Learner Type	**Activities**
Sensing Intuitive	Lecture sections at the beginning of each experiment
Visual	Laboratory presentations and different diagrams (flowcharts, schematics)
Active Reflective	Development of each experiment (building circuits in breadboards and developing codes for MCUs)
Sequential	Basic exercises and progressive designs
Global	Full development of the whole experience

6.2.2 Laboratory Manual Design

The laboratory manual was designed with the objective of providing students with a tool that guides them in their learning process and laboratory experiences development. This manual was designed based on the experiment structure described previously and a set of topics selected from the course syllabus. Each topic was

described as a laboratory experiment with a set of basic exercises (BE) and a complementary task (CT).

Topics Covered by the Manual

Once the experiment structure, learning objectives, and methodology were identified, the topics to be taught in the laboratory were selected. Based on the current class topics, the laboratory topics were chosen with the objective to help in the accomplishment of student's learning objectives as defined in Section 6.1.2. Also, the topics selected were those that provided enough material for a hands-on exercise that could be developed independently from the MCU platform to be used by students. Another important aspect taken into consideration was the course topics schedule because it was important to provide a period in which students could concentrate only on their class project. Table 6.6 shows the current course topics and those included in the laboratory manual.

An additional topic related to High-Voltage Safety was selected to be included in the manual due their importance for the laboratory work and students' safety. This topic was include in the manual as the first laboratory experiment.

Manual Experiment Descriptions

The topics included in the laboratory were organized in dedicated sections that provided: (1) a set of objectives to be met by students, (2) a bill of materials used for the experiment, (3) a lecture that provides a review of the current topic, and (4) exercises to be developed. Each experiment also includes schematics and block diagrams of every circuit to be used by students, and some of them included pseudocodes and flowcharts as a guide for their software implementation.

The experiments developed are described below while the developed manual is include in Appendix A:

Table 6.6 : Laboratory manual topics selected based on the current course topics

Class topics	Selected
1. Introduction to embedded system	
2. Embedded microcontroller architecture	
3. Life cycle of embedded	
4. Constraints in the design of embedded systems	
5. Basic interface and I/O fundamentals	X
6. Switches, keypads, and display	X
7. Interrupts	X
8. Timers and event counters	X
9. Pulse width modulation and event counter	X
10. Stepper motor interface	X
11. Serial communication	X
12. Analog-to-digital and digital-to-analog converters	X
13. Standard bus systems	
14. Synchronization schemes	
15. Memories	
16. DMA controllers	
17. Design technology in embedded systems	

Experiment 1. High-Voltage Safety: this experiment introduces students into the risk associated of working with high voltage. It also provides an understanding of how the electrical current could affect the human body, depending on their level and circulation path. The experiment ends with recommendations on how electrical risks could be reduced and describes a procedure to be followed in case of an emergency.

Experiment 2. IDE, GPIOs, and LCDs: this experiment introduces students to the process of creating and debugging a program for an MCU, while providing an understanding of how the General Purpose Input/Outputs (GPIOs) work. The experiment also provides the opportunity to learn how these GPIOs could be interfaced with other electronics components such as resistors, LEDs, and LCD displays. The list of activities are described below:

- Blinking LED

- Polling a switch
- LCD configuration
- Scrolling List (complementary task) ← (GPIOs, switches, and LCDs concepts)

Experiment 3. Interrupts, Switch Debouncing, and Keypad: this experiment provides students with an explanation of how interrupts in an MCU work and what are the functionalities of this MCU resource. The experiment also explains the bouncing phenomena associated with mechanical switches and the mechanisms used to mitigate it. Finally, it introduces students to the use of a keypad via interrupts. The list of activities include:

- Reading a key using interrupts
- Hardware debouncing
- Software debouncing
- Reading keypads through interrupts ← (Switches, LCDs, and interrupt concepts)
- Scrolling list with wheel (complementary task) ← (Laboratory 2 and interrupts concepts)

Experiment 4. Timers and LEDs: this experiment provides students with an explanation of how a timer works and how it can be used in typical embedded applications. The experiment also allows students to understand and apply display techniques to use multiplexed 7-segment displays. The list of activities are described below:

- Timer by polling
- Timer by interrupt
- 7-segment display
- Multiplexed display using a dual 7-segment ← (LEDs and timer concepts)
- Digital Tachometer (complementary task) ← (Laboratories 2, 3, and timer concepts)

Experiment 5. Low-Power Modes and PWM: this experiment explains how a low power mode could impact the power consumption and performance of an MCU. It also illustrates the typical architecture of a Pulsed-width Modulation (PWM) module and how it could be used in conjunction with another MCU resource to develop an embedded system application. The list of activities include:

- Low-power modes
- PWM signal generation
- Generating colors with an RGB LED ← (Laboratories 2, 3, and PWM concepts)
- Digital dimmer (complementary task) ← (Laboratories 2, 3, and PWM concepts)

Experiment 6. Motor Interfacing: this experiment provides access to three different types of direct current (DC) motors commonly used in embedded applications. The experiment provides a basic explanation of how motors work and standard techniques and circuits used to interface them. The list of activities are described below:

- DC motor driven with transistors
- DC motor controlled through driver Integrated-Circuit (IC)
- Servo-motor interfaces ← (Laboratory 5, and servo-motor concepts)
- Stepper motor interfaces
- Stepper motor characterization (complementary task) ← (Laboratories 2, 3, 4, and stepper motor concepts)

Experiment 7. Serial Communication: this experiment introduces students to the different types of serial communication protocols. It promotes the use of an asynchronous communication to establish a connection with a PC and synchronous communication to establish a connection with a real-time clock-calendar device. The list of activities include:

- Asynchronous serial communication (UART)
- Sending and receiving characters via UART
- Synchronous serial communication (I^2C) ← (Laboratories 2, 3, 4, and I^2C concepts)
- Digital alarm clock (complementary task) ← (Laboratories 2, 3, 4, and I^2C concepts)

Experiment 8. Data Converters (DAC & ADC): this experiment provides students with the opportunity of working with Digital-to-Analog converters (DACs) and Analog-to-Digital Converters (ADCs). The DAC is introduced through the use of external components and the ADC through the use of MCU's internal resources. This experiment also introduces students to the use of signal conditioners to interface analog devices. The list of activities are described below:

- Generating voltages using a DAC ← (Laboratory 4, and DAC concepts)
- Reading Voltages
- Analog-digital dimmer ← (Laboratories 2, 3, 5, and ADC concepts)
- Digital temperature meter (complementary task) ← (Laboratories 2, 4, and ADC concepts)

6.2.3 Educational Modules Design

The educational modules were designed to provide students with examples of how an electronic module for an embedded system could be implemented. These modules were constructed based on the circuits that are part of the manual laboratory experiments, where each module could be used with at least one experiment. With the modules, students can develop each experiment while learning how a functional module is designed, interfaced with an MCU, and how contemporary design techniques could be used in the development of embedded systems applications.

Due to the wide range of existing MCU, the elements for the module were selected to work with either 3.3V or 5.0V DC. Figure 6.2 shows pictures of the six modules

developed. The physical characteristics and components of each PCB can also be observed.

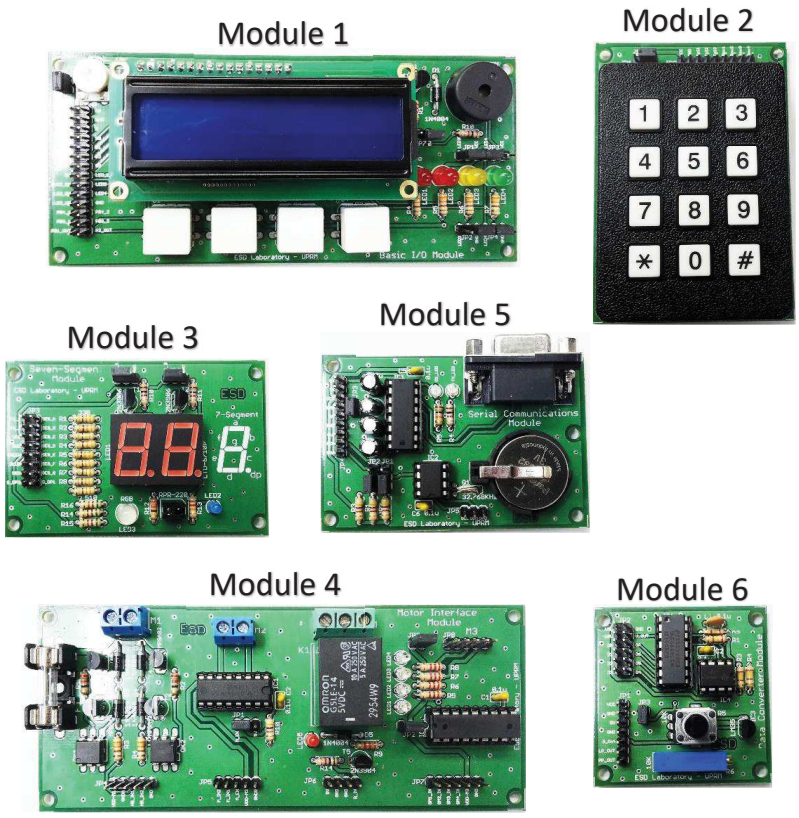


Figure 6.2 : Educational modules developed for the laboratory

The description and main characteristics of each module are provided below:

Basic I/O Module (Module 1)

The Basic I/O module is composed of four main blocks that provide access to common electronic peripherals that include pushbuttons, LEDs, LCD displays , and Buzzers. Figure 6.3 shows a block diagram of the Basic I/O module and its four blocks. This module is introduced in experiments 2, 3, 4, 5, 6, 7, and 8.

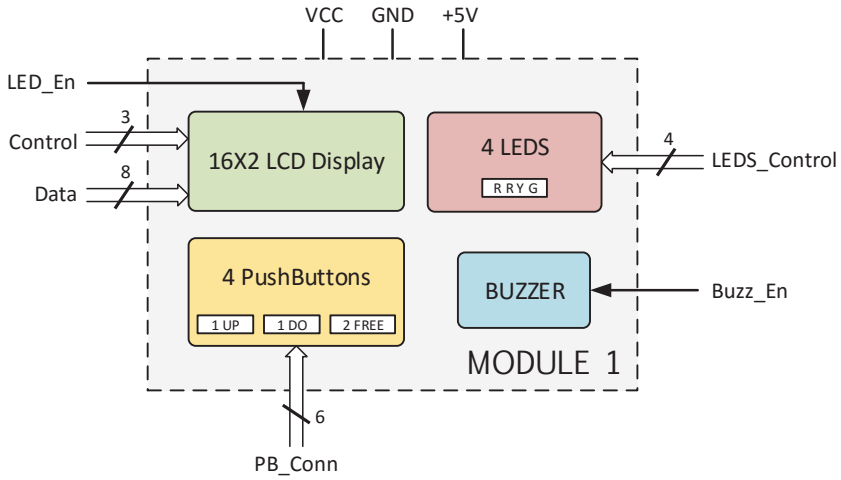


Figure 6.3 : Basic I/O module block diagram

Keypad Module (Module 2)

The keypad module provides access to one key matrix (keypad). Figure 6.4 shows a block diagram of the Keypad module and its 3x4 keypad. This module is used in experiments 3 and 5.

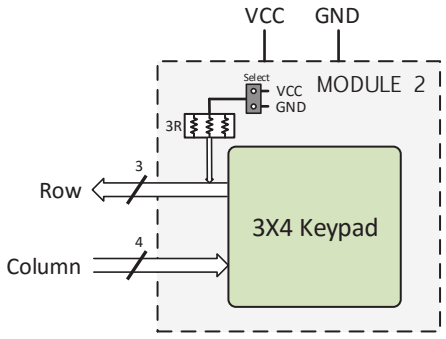


Figure 6.4 : Keypad module block diagram

Seven-Segment Module (Module 3)

The seven segment module is composed of three main blocks that provide access to optoelectronic peripherals such as 7-segment displays, RGB LEDs, and opto-switches. Figure 6.5 shows a block diagram of the seven segment module and its three blocks. This module is used in experiment 4.

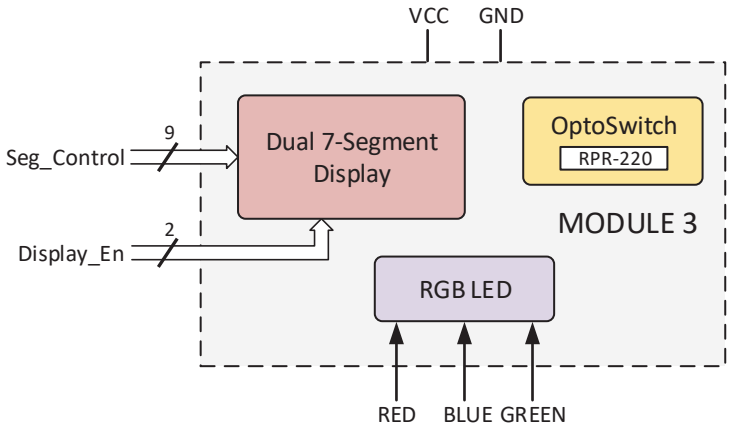


Figure 6.5 : Seven-Segment module block diagram

Motor Interface Module (Module 4)

The motor interface module is composed of four main blocks that provide access to common motor drivers such as H-bridges, relays, and stepper motors. Figure 6.6 shows a block diagram of the motor interface module and its four blocks. This module is used in experiment 6.

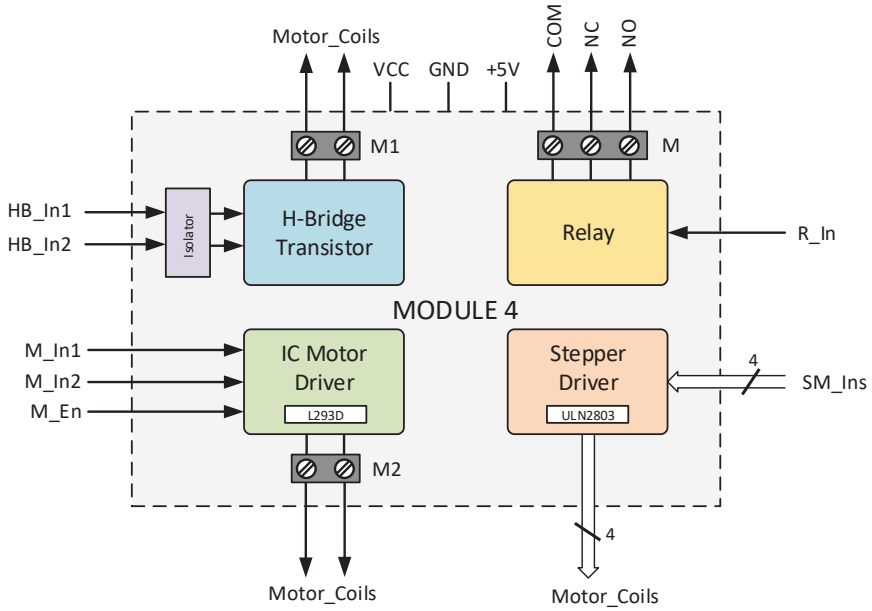


Figure 6.6 : Motor Interface module block diagram

Serial Communications Module (Module 5)

The serial communications module is composed of two main blocks that provide access to devices that use serial communication protocols such as RS232 and I^2C. Figure 6.7 shows a block diagram of the serial communication module and its two blocks. This module is used in experiment 7.

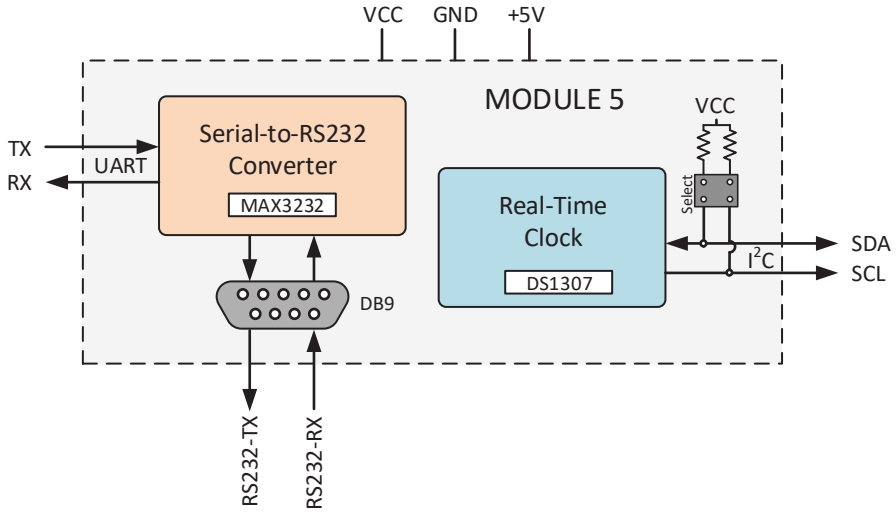


Figure 6.7 : Serial Communications module block diagram

Data Converters Module (Module 6)

The data converters module is composed of three main blocks that provide access to common analog devices and digital-to-analog converters. Figure 6.8 shows a block diagram of the data converter module and its three blocks. This module is used in experiment 8.

6.3 Assessment Methods

Following the Outcome Based-Education by Streveler *et al.* the assessment methods must allow to evaluate students' performance and the methodology implemented in class [30]. These evaluation methods were designed taking into consideration the learning objectives established for the class in order provide an alignment between

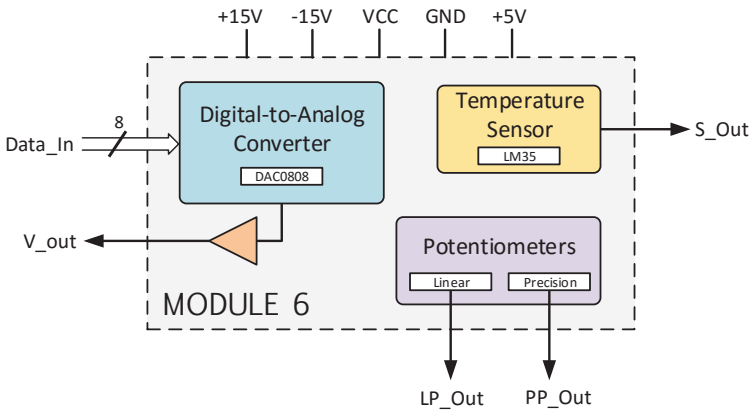


Figure 6.8 : Data Converters block diagram

the laboratory content and learning objectives. The assessment instruments developed included: a series of pre- and post- laboratory tests and a set of laboratory experiment activities.

6.3.1 Assessment Groups

With the objective of determining if the proposed methodology for the laboratory was successful, two groups of students were identified and compared. These two groups consisted of a representation of the laboratory methodology used in past years and the newly implemented methodology (Modular approach). Student results and data were divided into the following categories:

1. Students from Fall 2015 and Spring 2016 who took the laboratory without the implementation of the modular approach (control group)
2. Students from Fall 2016 who took the laboratory with the proposed modular design approach implemented (experimental group)

Due the implementation of the proposed modular design approach, some of the assessment activities implemented in the experimental group were slightly different in comparison with the control group. Also, some activities were valid only for the experimental group.

6.3.2 Tests Validations

A series of tests (pre- and post- laboratory tests) were designed in order to determine if the laboratory experiments developed positively impacted the student. Each test was composed of four questions, where the pre-test was given to students before the initiation of the experiment with the aim of identifying prior knowledge they had. Later on, after the experiment completition, students received a post-test, with the same questions as in the pre-test in order to assess the knowledge acquired during the experiment. See Appendix B for the actual tests developed for the control and experimental groups.

A validation test procedure was carried out for each test. The procedure took place before the use of their results to determine if the laboratory methodology applied was successful or not. This validation process consisted of an item analysis and a test reliability evaluation. The test reliability allowed us to determine if the tests implemented were appropriately designed while the item analysis is a procedure that allowed us to know the difficulty and discrimination of each item (question) [52, 53].

In the item analysis, the item difficulty index gave us an idea of how hard or easy a question was. This parameter was calculated based on Equation 6.1.

$$I_{Difficulty} = \frac{\#SWAC}{\#SWA} * 100\%, \tag{6.1}$$

where $\#SWAC$ is the number of students who answered the question correctly and the $\#SWA$ is the number of students who took the test. A value lower than 30% is considered as a high difficulty, a value between 30% and 80% is considered as a medium difficulty, and a value higher than 80% is considered as low difficulty [52].

The discrimination index, on the other hand, gave us an idea of how each question distinguishes the students who did well on the exam from the students who did poorly. The index was calculated using the Equation 6.2.

$$I_{Discrimination} = \frac{\#TSWAC}{\#TSWA} - \frac{\#BSWAC}{\#BSWA}, \tag{6.2}$$

where $\#TSWAC$ and $\#BSWAC$ are the students who correctly answered the question in the upper and lower student groups respectively, and $\#TSWA$ and $\#BSWA$ are the number of students who answered the question in the upper and lower student groups respectively. The number of students in the upper and lower groups was selected based on the criteria explained by Kelly [54] who mentioned that 27% from the extremes is an optimum value to define the upper and lower groups. In this index, a negative value indicates that the item or question does not allow to distinguish students who did well from students who did poorly in the test. Values near 1 are preferred.

For the test reliability, an internal-reliability factor was calculated to determine if the items that measure the same concept presented homogeneous responses. To calculate this factor for each test a Kuder-Richarson formula 20 (KR-20) was used because it is valid when items on an exam have different levels of difficulty and when they are discriminated in a dichotomous way [55, 56]. For our test responses, a factor of 70% was used to decide if the response for a question was correct or not because 70% is the current threshold value used by the ECE department to decided is a student passes or not a courses.

The KR-20 applied was:

$$\text{KR-20} = \left(\frac{K}{K-1}\right)\left(1 - \frac{\sum_{i=1}^{K} p_i q_i}{S^2}\right) \tag{6.3}$$

Where K is the number of items on the test, p is the probability of correctly answering a question (known as item difficulty), q is the probability of answering that question incorrectly, and S^2 is the overall students result variance. For this index,

values between 1 and 0.5 are preferred while values below 0.5 suggest that the items in the tests need to be revised or re-evaluated [57].

6.3.3 Assessment Laboratory Experiment

To assess the laboratory experiments, a set of pre- and post- laboratory tests were given to the student in each experiment. To determine a relation between the results of the pre- and post-tests, learning gain factors were computed for each test. These factors allowed to establish the students' performance at two different times or moments in their studies and also know if the laboratory experiences helped them in their learning process [58]. A individual gain factor ($G_i = post\_test\ score - pre\_test\ score$) was calculated for each student while for each test an absolute gain ($G_A = average(\frac{G_i}{Maximun\ score\ achievable})$) and relative factors ($G_R = average(\frac{G_i}{pre\_test\ score})$) were calculated. Also, a *test average normalized gain* ($\langle g \rangle$) and *average of single-student normalized gain* ($\langle g_i \rangle_{ave}$) were calculated to measure the tests' effectiveness. The angle brackets "$\langle\rangle$" indicate class average. The normalized gain was calculated using Equation 6.4 [59, 60]:

$$\langle g \rangle = \frac{\langle \%post\_test \rangle \ - \ \langle \%pre\_test \rangle}{100\% \ - \ \langle \%pre\_test \rangle}, \tag{6.4}$$

where $\langle \%pre\_test \rangle$ is the average pre-test grades and $\langle \%post\_test \rangle$ is the average post-test grades. For the single-student normalized gain calculation, the pre- and post-test grades correspond to each individual grade, without average. A predefined score of 30% was taken as the minimum value to define if the test was effective [60]. A positive value in the factors means that students presented a better performance in the post-test in comparison with the pre-test score or mean that the test had a positive impact on students' learning. The results for each pre- and post-test are presented in Section 7.2.

6.3.4 Methodology Assessment

To assess the methodology implemented, a comparison between the performances of the two students groups was carried out using an unpaired t-statistic (see Equation 6.6). This statistic allowed us to determine if there was a significance difference between the use of the old laboratory methodology and the proposed methodology. With the statistic, a t_0 was calculated using the mean of the students' performance. Finally, this value was compared against the t distribution that use a significance level (α) in order to infer if the students' performance was statistically different, by rejecting the null hypothesis.

$$H_0 : \mu = \mu_o, \tag{6.5}$$

H_0 is our null hypothesis that assumes the mean students' performance for both groups are equal.

The unpaired t-statistic formula used was:

$$t_0 = \frac{\bar{x_1} - \bar{x_2}}{SE(\bar{x_1} - \bar{x_2})}, \tag{6.6}$$

where $\bar{x_1}$ and $\bar{x_2}$ are the means of the two groups (1 for the experimental group and 2 for the control group), and the $SE(\bar{x_1} - \bar{x_2})$ is the standard error of the difference between the means. The standard error calculation may varies depending on whether the variances between the samples are similar or not. Equation 6.7 denotes the formula for approximately equal variances while Equation 6.8 denotes the formula for unequal variances.

$$SE(\bar{x_1} - \bar{x_2}) = \sqrt{\left(\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}\right) * \left(\frac{1}{n_1} + \frac{1}{n_2}\right)} \tag{6.7}$$

$$SE(\bar{x_1} - \bar{x_2}) = \sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}, \tag{6.8}$$

where S_i^2 and n_i are the variance and sample size of each group respectively.

6.4 Alignment

In the OBE approach, the course content, pedagogical methodology, and assessment activities must be aligned. Figure 6.9 shows the process carried for the alignment of the OBE in which the laboratory learning objectives were established in first place to later select the appropriate teaching methods according to the laboratory. Finally, assessment methods were developed to verify if the learning objectives were met and verify if the methods selected to teach students were satisfactory. Also, this alignment allows us knowing the level of accomplishment that each student has with each learning objective.

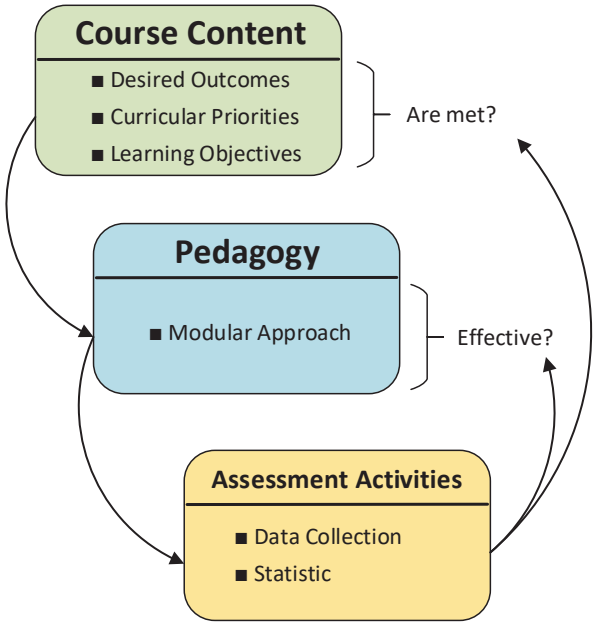


Figure 6.9 : OBE Alignment

Table 6.7 shows the relationship established between the laboratory manual activities and the learning objectives while Table 6.8 shows the relation between the test items and learning objectives. With this correlation, a complete profile for each student was created from the grades acquired during the semester. Each activity

marked for an objective contributes to the same weight in the accomplishment of the objective. Each objective was discriminated in levels based on the final result where a low level was assigned to grades under 70 pts, a medium level to grades between 70 and 85 pts, and high level of accomplishment for grades above of 85.

Table 6.7 : Laboratory activities vs learning objectives

Activities		L. Obj. 1	L. Obj. 2	L. Obj. 3	L. Obj. 4	L. Obj. 5	L. Obj. 6	L. Obj. 7	L. Obj. 8	L. Obj. 9
Exp. 2	BE-1	•			•			•		
	BE-2				•			•		
	BE-3				•			•		
	CT-4		•		•	•	•	•	•	•
Exp. 3	BE-1	•			•			•		
	BE-2				•			•		
	BE-3							•		
	BE-4				•			•		
	CT-5		•		•	•	•	•	•	•
Exp. 4	BE-1	•			•			•		
	BE-2							•		
	BE-3				•			•		
	BE-4							•		
	CT-5		•		•	•	•	•	•	•
Exp. 5	BE-1	•						•		
	BE-2	•						•		
	BE-3		•		•			•		
	CT-4		•		•	•	•	•	•	•
Exp. 6	BE-1	•			•			•		
	BE-2							•		
	BE-3	•			•			•	•	
	CT-4				•	•	•	•	•	•
Exp. 7	BE-1				•			•		
	BE-2				•			•		
	BE-3				•			•		
	BE-4				•			•		
	CT-5		•		•	•	•	•	•	•
Exp. 8	BE-1				•			•	•	
	BE-2	•			•			•		
	BE-3				•			•		
	CT-4		•		•	•	•	•	•	•

Table 6.8 : Tests items vs learning objectives

Items		L. Obj. 1	L. Obj. 2	L. Obj. 3	L. Obj. 4	L. Obj. 5	L. Obj. 6	L. Obj. 7	L. Obj. 8	L. Obj. 9
Test 2	IT-1	•	•							
	IT-2			•	•	•				
	IT-3					•	•			
	IT-4			•	•					
Test 3	IT-1	•		•						
	IT-2			•			•			
	IT-3			•	•					
	IT-4				•	•				
Test 4	IT-1	•		•						
	IT-2			•	•	•				
	IT-3					•	•			
	IT-4			•	•	•				
Test 5	IT-1	•		•						
	IT-2	•		•						
	IT-3				•	•				
	IT-4	•		•				•		
Test 6	IT-1	•		•		•				
	IT-2			•						
	IT-3				•					
	IT-4				•	•				
Test 7	IT-1	•		•						
	IT-2			•						
	IT-3			•	•					
	IT-4				•		•			
Test 8	IT-1	•		•						
	IT-2	•		•		•				
	IT-3			•	•					
	IT-4				•	•				

6.5 Complementary Laboratory Material

To help in the improvement of the students' design skills a set of tutorials were designed. Also, a manual reference for the educational modules was developed.

6.5.1 Tutorial Recommended Handouts Design

The tutorial topics were chosen based on a group of topics considered to be important for students that work with embedded systems. These tutorials were implemented through handouts that could be taught at any moment during the class. Some of the tutorials were developed in past semesters but they were modified to

introduce changes or adapting them to the current format used in the laboratory. Instructor decides the order and time for the tutorials. The topics also include the use of software and measurement equipments.

The handouts developed are listed below (see Appendix C):

- High Voltage Safety: This tutorial goes with the laboratory experiment 1 and it explains how to work with high voltage and its most important considerations.
- Soldering: This tutorial teaches the procedure to solder through hole and surface-mount device (SMD) components using a soldering iron and rework stations. It also explains the characteristic of a good soldering finish and the procedure to desolder components.
- EagleCAD: With this tutorial students learn how to use the essential features of the PCB design software EagleCAD. The students learn how to create schematic and board layout for a design. Also, they learn how to design library-entries for new components.
- Logic Analyzer: The tutorial introduces student in the utilization of a logic analyzer for debugging their circuits. It explains how to use the essential features of the laboratory oscilloscope and their main operation modes.

6.5.2 Electronics Modules Reference Manual Design

A modules manual was designed to provide students with the documentation necessary for the assembly and use of the educational modules. The manual includes a description, block diagram, schematic, and PCB layout of each module. It also contains a bill of materials with part numbers and suppliers for each component used in the development of the modules. The manual is divided into six chapters, where each chapter corresponds to a module. See Appendix D for the manual developed.

Chapter 7
RESULTS AND ANALYSIS

This chapter presents the results obtained from this work and their analysis. The first Section discusses the validation of each test through the use of an item analysis procedure. The second Section presents the comparison, in terms of learning gain factors, between the two student groups under study. The comparison was based on the pre- and post-test grades. Finally, a performance comparison between the groups using a t-statistic is presented. This last analysis took into consideration the tests and laboratory exercises grades.

7.1 Test Analysis and Validation

Before using the test results to determine whether the proposed methodology positively impacted student's performance or not, an item analysis for each test was performed. This analysis helped ascertain the item difficulty and discrimination for each test. Also, an inter-reliability factor was calculated to determine if different items, in the same test, presented homogeneous responses. These factors were used to determine how well the test was developed. For these calculations, only post-test grades were taken into consideration. The pre-tests were not used as the students had not yet been exposed to the material.

7.1.1 Control Group Test Analysis

The control group consisting of 16 students, was administered a total of six different tests. Table 7.1 shows the item analysis results for the control group, while Table 7.2 lists the topics for those tests.

Table 7.1 : Control group tests item analysis

	Item Difficulty				Item Discrimination				Reliability
	Q1	**Q2**	**Q3**	**Q4**	**Q1**	**Q2**	**Q3**	**Q4**	**(KR-20)**
Test 1	75%	75%	38%	44%	0.60	0.80	0.80	1.00	0.752
Test 2	38%	50%	88%	69%	0.60	0.80	0.40	0.80	0.596
Test 3	75%	6%	38%	25%	0.60	0.20	0.80	0.40	0.375
Test 4	38%	31%	38%	75%	0.80	0.20	1.00	0.40	0.388
Test 5	50%	25%	56%	13%	0.60	0.20	1.00	0.40	0.338
Test 6	50%	19%	19%	81%	0.80	0.60	0.60	0.60	0.674

Table 7.2 : Control group test topics

Test	**Topic**
Test 1	High-Voltage Safety
Test 2	IDE, ASM/C Programming & IO
Test 3	Interrupt & Switch Debouncing
Test 4	Timers and Applications
Test 5	Low-Power Modes, LED Display Techniques & keypads
Test 6	Introduction to Serial Communications

From Table 7.1 , in the case of Test 1, it was observed that the items (questions in a test) were between 38% and 75% in terms of difficulty. Furthermore, these items are considered as medium difficulty questions. It is worth says that values below 30% are for items considered as hard and values over 80% are for items considered as easy. In this test, the average difficulty expected (near to 50%) was achieved. In the case of item discrimination, all the items presented values near to 1 (above 0.5) allowing to distinguish students who did well from those who did poorly on the test. A KR-20 value above 0.5 was obtained for this test which indicates that the test was well designed. For Test 2, the items difficulty were in acceptable values, however, the discrimination factor for Q3 was low. The KR-20 value for this test was above 0.5 (can be considered well designed). For Test 3, only Q2 resulted too difficult (6%)

and the others were in the expected values. Q2 and Q4 obtained low values in the discrimination factor (0.20 and 0.4 respectively). These two questions (Q2 and Q4) could be revised to improve the test performance and reliability. Similar behavior was obtained for Tests 4 and 5. The item difficulties produced acceptable values but the discrimination index for Q2 and Q4 (in both tests) were low. In the case of Test 6, the item difficulties were appropriate and the discrimination index was above 0.5. The KR-20 value for this test was as expected.

Although three tests obtained values above 0.5 in the reliability factor, the average tests reliability was of 0.52. This general low value was attributed to the results obtained in Tests 3, 4, and 5 which obtained values below 0.4 in the KR-20.

7.1.2 Experimental Group Test Analysis

The experimental group consisting of 25 students, was administered a total of eight different tests. The difference between the number of tests applied to the control and experimental groups was determined by the number of laboratory topics included in the new modular teaching approach, which covered two new topics in the laboratory schedule. For this reason, some items (questions) had to be reorganized. All tests were revised by the class professor to ensure correlation between the items and the topic of each test. Tables 7.3 and 7.4 show the analysis and test topics for the experimental group.

Although small changes to the tests were introduced, the results changed significantly. In the case of Test 1, the average difficulty was of 84% with lower discrimination index on two items (Q1 and Q4). Unfortunately, results showed that this test obtained a reduced KR-20 factor of 0.135 and it can not be considered well designed. Similar results were obtained for Test 2. In the case of Tests 3 and 4, although the difficulty and correlation were low in both indexes, the KR-20 factors were in the expected values and therefore these tests can be considered well designed. For Tests 5, 6, and 7, the items had acceptable difficulty values. For these tests, only one question

Table 7.3 : Experimental group tests item analysis

	Item Difficulty				Item Discrimination				Reliability
	Q1	**Q2**	**Q3**	**Q4**	**Q1**	**Q2**	**Q3**	**Q4**	
Test 1	96%	80%	76%	84%	0.17	0.50	0.83	0.17	0.135
Test 2	92%	56%	92%	96%	0.17	0.83	0.33	0.17	0.329
Test 3	84%	84%	24%	84%	0.67	0.67	1.00	0.33	0.628
Test 4	60%	88%	84%	88%	0.83	0.50	0.33	0.50	0.613
Test 5	72%	76%	76%	68%	0.67	0.83	0.50	0.33	0.281
Test 6	76%	68%	76%	32%	0.67	0.17	0.83	0.67	0.182
Test 7	76%	92%	52%	68%	0.67	0.33	0.83	0.67	0.394
Test 8	76%	56%	60%	28%	0.50	1.00	1.00	0.67	0.600

Table 7.4 : Experimental group test topics

Test	**Topic**
Test 1	High-Voltage Safety
Test 2	IDE, GPIOs, and LCD
Test 3	Interrupts, Switch Debouncing, and Keypad
Test 4	Timers and LEDs
Test 5	Low-Power Modes and PWM
Test 6	Motors Interfacing
Test 7	Serial Communications
Test 8	Data Converters (DAC & ADC)

obtained a value below 0.5 in the discrimination index but the tests had lower KR-20 values, thus can not be considered well designed. In the case of Test 8, the average item difficulty was as expected (near to 50%), the discrimination and reliability were over 0.5. Furthermore, this test can be considered well designed.

The low values obtained for the different factors, in both student groups, could be associated with the sample size. As previously indicated, the sample size for the control group was of only sixteen students while for the experimental group had 25. Table 7.5 presents the KR-20 factors obtained for Tests 1 and 2 when students

for both groups were used in the same calculation (sample size of 41 students). In the aforementioned Table, we can observe that the two tests can be considered well designed by getting KR-20 factors above 0.5.

Table 7.5 : Control and Experimental group tests item analysis

	Item Difficulty				**Item Discrimination**				**Reliability**
	Q1	**Q2**	**Q3**	**Q4**	**Q1**	**Q2**	**Q3**	**Q4**	
Test 1	88%	78%	61%	68%	0.27	0.63	1	0.72	0.61
Test 2	70%	53%	90%	85%	0.82	0.82	0.27	0.54	0.52

Finally, we can conclude that a total of three tests in each group, Tests 1, 2, and 6 for the control group and Tests 3, 4, and 8 for the experimental group, obtained reliability factors over 0.5. Furthermore these tests can be considered well developed. Although the other tests in both groups did not reach the minimum value necessary to be considered well designed, the data obtained from these tests can still be used to analyze the students' learning gains and making the performance comparison needed. This was concluded when an analysis for Tests 1 and 2 using both student groups together was held. The results allowed attributing the low values obtained to the sample size because the reliability factors for both tests increased from 0.135 and 0.329 to 0.61 and 0.52 respectively (sample size of 41).

7.2 Learning Gain Analysis

Once the tests were validated, the pre- and post-test scores were used to analyze if the tests had a positive impact on the students' learning. For each test, learning factors were computed to determine the effectiveness of the test using a score threshold as a reference. According to Hake [60], a predefined value of 30% could be taken as the minimum value to define if a test was effective or not. Also, learning gain factors for each student were calculated to individually identify the performance of each student.

7.2.1 Control Group Gain Analysis

Table 7.6 summarizes the results for test learning factors. Figure 7.2 illustrates student results. The detailed grades for each student for each test are presented in Appendix E - Section E.1

Table 7.6 : Control group learning gain factors

	μ Pre-Test	μ Post-Test	μ G_i	G_A	G_R	$\langle g \rangle$	$\langle g_i \rangle_{ave}$
Test 1	27.7%	61.6%	34.0%	34%	123%	47%	49.28%
Test 2	56.9%	70.0%	13.1%	13%	23%	30%	37.04%
Test 3	44.3%	53.5%	9.2%	9%	21%	17%	20.68%
Test 4	47.4%	56.3%	8.8%	9%	19%	17%	24.95%
Test 5	41.0%	44.1%	3.1%	3%	8%	5%	14.93%
Test 6	25.0%	52.7%	27.7%	28%	111%	37%	36.64%

From Table 7.6 , in the case of Test 1, it was observed that the absolute learning gain (G_A) was of 34%, which means that students presented a better performance in the post-test in comparison with the pre-test. We can also reach this conclusion by noting that the average normalized grade for the post-test was higher than for the pre-test. The relative gain factor (G_R) for this test was 123%, which means that students significantly increased their grades from the pre-test to the post-test. Finally, the *average of single-student normalized gain* ($\langle g_i \rangle$) and *test average normalized gain factor* ($\langle g \rangle$) were above 45% meaning that this laboratory experience promoted student learning. For Test 2, the G_A was of 13% meaning that students did better on the post-test than the pre-test. Although the values for the average individual gain (G_i) and G_R were below 30% due to the proximity between pre- and post-test grades, the $\langle g \rangle$ and $\langle g_i \rangle$ obtained values above 30% meaning that this test can also be considered effective. For Tests 3, 4, and 5, the gain factors were below 30%. Although the pre- and post-test grades for these tests were near the 50% of the maximum score achievable, these tests can not be considered effective. In the case of Test 6, the G_A was of 28% which indicates that students did well in the post-test. The G_R was of

111% meaning that students significantly increased their grades. The $\langle g \rangle$ and $\langle g_i \rangle$ obtained values above 30%, for this reason the test can also be considered effective.

Although three of the six tests are considered effective in terms of the learning gain factors, only Test 2 reached the minimum score in the post-test grade for success in a class or examination, as established by the ECE department. The other post-tests obtained values near to 50%.

Figure 7.2 represents the average learning gain for each student. From this figure, we can observe that nine students obtained a learning gain factor above 30% and only three students, from those nine students, obtained gains above 40%. This means that the remaining students did not have a significant learning gain from the current laboratory experiments.

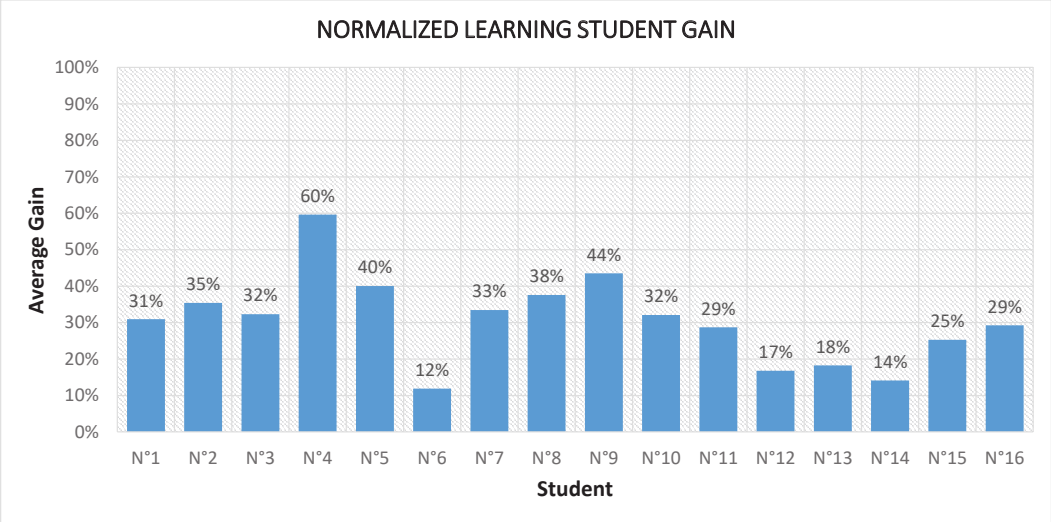


Figure 7.1 : Control group: Student learning gains

7.2.2 Experimental Group Gain Analysis

In the case of the experimental group, Table 7.7 summarizes the results for the test learning factors. Figure 7.2 also illustrates the student's results. The detailed grade distribution for each student for each test are presented in Appendix E - Section E.2

Table 7.7 : Control group learning gain factors

	μ Pre-Test	μ Post-Test	μ G_i	G_A	G_R	$\langle g \rangle$	$\langle g_i \rangle_{ave}$
Test 1	22.1%	88.6%	66.5%	66%	301%	85%	83.60%
Test 2	50.2%	82.5%	32.3%	32%	64%	65%	61.19%
Test 3	38.6%	76.3%	37.7%	38%	98%	61%	59.52%
Test 4	51.6%	83.3%	31.7%	32%	62%	66%	59.99%
Test 5	40.4%	79.7%	39.3%	39%	97%	66%	64.41%
Test 6	22.0%	75.8%	53.8%	54%	244%	69%	67.61%
Test 7	21.9%	73.0%	51.1%	51%	234%	65%	63.89%
Test 8	27.3%	64.1%	36.8%	37%	135%	51%	51.65%

From Table 7.7 , in the case of Test 1, it was observed that the absolute learning gain (G_A) was of 66%, which means that students presented a better performance in the post-test in comparison with the pre-test. We can also reach this conclusion by nothing that the post-test average grade was higher than the pre-test average grade. The relative gain factor (G_R) for this test presented a value of 301%, which means that students significantly increased their grades from the pre-test to the post-test. Finally, the *average single-student normalized gain* ($\langle g_i \rangle$) and *test average normalized gain* were over 30%, which means that this lab experiment promoted student learning. For tests 2-8, similar results were obtained. These tests achieved absolute, relative, and test average gains over 30% which means that these tests can also be considered effective.

In the case of the experimental group, only Test 8 scores (post-test) were below the threshold defined by the ECE department (70%). This means that students had a good performance solving the given problems in almost all tests.

Figure 7.2 represents the average gain learning for each student. From this figure, we can observe that all students obtained a learning gain above 30% and also, six students had gain values above 70%. This means that the laboratories, with the proposed methodology, positively impacted the student learning process.

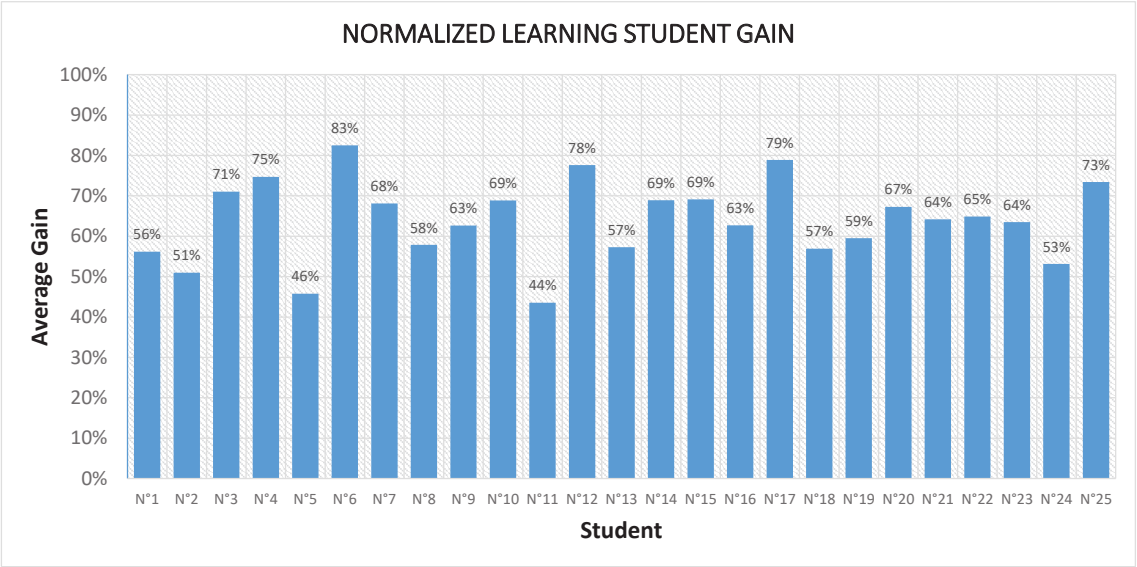


Figure 7.2 : Experimental group: Student learning gains

Finally, we can conclude that the laboratory tests and experiences developed with the proposed methodology promote a better student learning. This can be demonstrated by obtaining eight tests that can be considered effective, tests with grades over 64%, and students with a gain average above 63%.

7.3 Performance Comparison

In order to validate the proposed methodology, a set of performance comparisons between the two students groups were carried out. These comparison were developed using the students grades to the tests and laboratory exercises.

7.3.1 Using Tests Grades

For the tests grades comparison, an unpaired t-statistic formula was used because the sample size for both groups were different. The data used for this comparison was the *average single-student normalized gain* of each student. Table 7.8 shows the students gains for both the control and experimental group.

The first step was determining if the variance of both groups were similar to define which $SE(\bar{x_1} - \bar{x_2})$ formula needs to be used. Table 7.9 shows the results

Table 7.8 : Control and experimental group single-student gains

Control Group		**Experimental Group**	
Student	$\langle g_i \rangle_{ave}$	**Student**	$\langle g_i \rangle_{ave}$
N°1	30.93	N°1	56.16
N°2	35.39	N°2	50.94
N°3	32.32	N°3	71.06
N°4	59.65	N°4	74.71
N°5	40.04	N°5	45.76
N°6	11.90	N°6	82.53
N°7	33.46	N°7	68.10
N°8	37.59	N°8	57.86
N°9	43.53	N°9	62.67
N°10	32.08	N°10	68.87
N°11	28.71	N°11	43.54
N°12	16.81	N°12	77.59
N°13	18.28	N°13	57.27
N°14	14.15	N°14	68.92
N°15	25.27	N°15	69.13
N°16	29.26	N°16	62.71
		N°17	78.87
		N°18	56.89
		N°19	59.49
		N°20	67.29
		N°21	64.21
		N°22	64.90
		N°23	63.51
		N°24	53.15
		N°25	73.43

obtained in which we can conclude that the variances were completely different and therefore the formula for unequal variances was selected. The control group had a value of 145.432 which is grater than 98.4932 obtained by the experimental group.

Table 7.9 : Means, Sample size, standard deviation, and variance for control and experimental group

Group	**Sample size** (n)	**Mean** ($\bar{x}_i$)	**Std** (S_i)	**Var** (S_i^2)
Experimental	25	63.98	9.924373	98.49317
Control	16	30.59	12.05954	145.4325

The second step was to calculate $SE(\bar{x_1} - \bar{x_2})$ which had a value of 3.61.

$$SE(\bar{x_1} - \bar{x_2}) = \sqrt{\frac{98.49317}{25} + \frac{145.4325}{16}} = 3.61 \tag{7.1}$$

Finally, the t-statistic was calculated.

$$t_0 = \frac{\bar{x_1} - \bar{x_2}}{SE(\bar{x_1} - \bar{x_2})} = \frac{63.98 - 30.59}{3.61} = 9.25 \tag{7.2}$$

For rejecting or accepting the null hypothesis, the t_0 value calculated was compared against the critical t distribution value. This critical value was obtained using the degree of freedom for our analysis (28) and a significance value of $\alpha = 0.05$. The critical value obtained, from the t statistic table, was of 1.70 which is lower than the t_0 value calculated, therefore the null hypothesis was rejected. Furthermore, there is strong evidence that the experimental group performance was statistically different from the control group. Also, we can conclude that the experimental group had a better performance, therefore the proposed methodology promotes a better student learning.

7.3.2 Using Laboratory Exercises Grades

For the laboratory exercises grades comparison, a Mann-Whitney statistic was used instead of the t-statistic previously used. This change occurred because at the moment of analyzing the data, using the average laboratory exercise grades, it was found that these values were not normally distributed. For this statistic, a null hypothesis (H_0) that assumes "the medians of the students' grades for both groups are equal", was raised. Table 7.10 shows the student's laboratory grades for both the control and experimental group.

The first step was organizing the grades (including both groups) from the lower to the highest value, assigning a rank number to each value. If two or more grades

Table 7.10 : Control and experimental group laboratory grades

Control Group		Experimental Group	
Student	**Grade**	**Student**	**Grade**
N°1	84.13	N°1	96.70
N°2	76.15	N°2	87.51
N°3	68.02	N°3	96.86
N°4	84.13	N°4	88.00
N°5	77.22	N°5	70.09
N°6	77.22	N°6	88.00
N°7	76.75	N°7	68.34
N°8	84.13	N°8	68.34
N°9	68.02	N°9	87.51
N°10	68.58	N°10	96.70
N°11	82.76	N°11	87.51
N°12	82.76	N°12	70.09
N°13	68.58	N°13	88.00
N°14	73.83	N°14	97.14
N°15	73.83	N°15	96.96
N°16	82.76	N°16	89.09
		N°17	97.14
		N°18	96.75
		N°19	96.86
		N°20	96.75
		N°21	75.66
		N°22	96.75
		N°23	96.96
		N°24	89.09
		N°25	44.34

had the same value, a shared rank was determined for those numbers. Table 7.11 shows the organized grades and the ranks assigned.

The second step was to calculate the sum of the ranks for each group (R_i), calculated the U_i values using the Equation 7.3 and Equation 7.4, and choosing the lowest U_i value for the "Z" approximation (Equation 7.5). The Z statistic approximation was necessary because one of the groups had a sample size larger than 20.

$$U_1 = n_1 \cdot n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{7.3}$$

$$U_2 = n_1 \cdot n_2 + \frac{n_2(n_2+1)}{2} - R_2 \tag{7.4}$$

$$Z = \frac{U_{min} - ((n_1 \cdot n_2)/2)}{\sqrt{\frac{n_1 \cdot n_2(n_1+n_2+1)}{12}}} \tag{7.5}$$

where n_1 and n_2 represents the sample size for each group.

Having into account the sample sizes ($n_1 = 16$ for the control group and $n_2 = 25$ for the experimental group), the sum of ranks ($R_1 = 214$ and $R_2 = 647$), and the U_i values calculated ($U_1 = 322$ and $U_2 = 78$); the Z value was of:

$$Z = \frac{78 - ((16 \cdot 25)/2)}{\sqrt{\frac{16 \cdot 25(16+25+1)}{12}}} = \frac{-122}{37.4165} = -3.2605 \tag{7.6}$$

For rejecting or accepting the null hypothesis, the $|Z|$ value calculated was compared against the critical Z distribution value. This critical value was obtained using a significance value of $\alpha = 0.05$. The critical value obtained, from the Z statistic table, was 1.96 which is lower than the $|Z|$ value calculated, therefore the null hypothesis was rejected. Furthermore, there is strong evidence that the experimental group performance was statistically different from the control group performance. In addition, with this result we can reaffirm the conclusion raised previously "The experimental group had a better performance, therefore the proposed methodology promotes a better student learning".

Table 7.11 : Control and experimental group ranked grades

Groups		
Control	**Experimental**	**Rank**
	44.34	1
68.02		2.5
68.02		2.5
	68.34	4.5
	68.34	4.5
68.58		6.5
68.58		6.5
	70.09	8.5
	70.09	8.5
73.83		10.5
73.83		10.5
	75.66	12
76.15		13
76.75		14
77.22		15.5
77.22		15.5
82.76		18
82.76		18
82.76		18
84.13		21
84.13		21
84.13		21
	87.51	24
	87.51	24
	87.51	24
	88.00	27
	88.00	27
	88.00	27
	89.09	29.5
	89.09	29.5
	96.70	31.5
	96.70	31.5
	96.75	34
	96.75	34
	96.75	34
	96.86	36.5
	96.86	36.5
	96.96	38.5
	96.96	38.5
	97.14	40.5
	97.14	40.5

Chapter 8
CONCLUSIONS AND FUTURE WORK

A methodology to teach embedded system design concepts in a laboratory course was developed. It consisted of two main parts, a methodology for establishing the course structure (define the content and related activities) and an educational approach (based on modular design) for teaching the concepts related to the course.

Several methodologies for structuring a course or laboratory presented by other authors were initially discussed. General benefits and limitations of these methodologies were analyzed and compared. An OBE approach was selected because it is based on outcomes (same focus of the engineering department courses) and it search for aligning the laboratory content, pedagogical methods, and assessment activities. Also, approaches for teaching embedded systems design concepts were carefully analyzed. Strengths and weaknesses were identified. A modular approach was selected and implementations of this methodology were discussed. These implementations were taken as a base to propose a new teaching methodology.

The OBE framework was implemented in four steps. In the first step, the laboratory content was revised and designed taking into consideration the current *Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*, social and industrial expectations, and current departmental focus for the computer engineering program. Later, the pedagogical methods were implemented using a modular approach as bases. This implementation was carried out through a laboratory manual with a set of progressive experiments and a set of educational modules. The educational modules provided students with an example of how small circuits could

be grouped to create functional electronic modules and how they can be incorporated into an MCU to create functional embedded prototypes. Assessment methods that consisted in a series of pre- and post-tests were designed to validate the proposed methodology. These tests were given to the students in past semesters (control group) and students with the proposed methodology (experimental group). Finally, a correlation between the activities developed and learning objectives were established (student profile).

A validation process for the tests was carried out. This validation consisted on determining the item difficulty and discrimination index for each question, and reliability and learning gain factors for each test. In the case of the control group, it was observed that only three tests (Tests 1, 2, and 6) obtained results over the expected values in terms of reliability factor and only the same tests can be considered effective. Instead, for the experimental group, Tests 3, 4, and 8 obtained reliability factors above the expected value (0.5). Furthermore, these tests can be considered well developed. Although the remaining tests in both groups were considered not well developed, the low values were associate to the sample size. This was evident when the analysis for Tests 1 and 2 using both student groups together was held. Reliability factors for these tests improved from 0.135 and 0.329 to 0.61 and 0.52 respectively (values over 0.5). These results indicated that these tests can be considered well developed. For the experimental group, all tests were considered effective by obtaining gain scores above 51%. These results allowed to conclude that the laboratory experiences developed with the proposed methodology were correctly designed and promoted a successful learning experience for students.

Finally, a comparison between the both student's groups, in terms of test performance and laboratory exercises, was carried out. The results demonstrated that the performance obtained by the experimental group was statistically different from the performance obtained by the control group. In addition, the analysis showed that the

performance of the experimental group was better allowing to validate the proposed methodology which promotes a significant impact on the student learning.

8.1 Future Work

To expand and improve the current state of the presented work, the following directions were identified:

1. To collect data during two or more semesters using the proposed methodology. These data would be used to obtain more accurate statistical results.
2. To identify other suitable circuits that could be implemented in the form of modules. These modules could become part of the actual set of educational modules.

Chapter 9
CONTRIBUTIONS

The main contribution of this work was the implementation of an outcome-based education framework together with a modular design approach for teaching embedded systems design concepts in the ESD laboratory at the University of Puerto Rico Mayagüez campus. Others contributions of this work include:

1. A formal methodology based on modular design and an OBE framework for structuring an applied Laboratory.
2. A set of six different educational modules to be used by the students in the development of their laboratory practices.
3. A total of eighty-four electronic modules (fourteen copies of each module developed) to be used by the fourteen laboratory stations.
4. A laboratory manual composed of eight laboratory experiences to be used in next semesters to teach the embedded system design laboratory.
5. A set of tutorials, in the format of PowerPoint presentations, to enhance students abilities in the embedded systems design area and the usage of laboratory equipment.
6. A documentation package for the educational modules that include a set-up manual and blueprints. This documentation will allow others to replicate the educational modules.
7. A set of assessment tools that will help instructors to measure students performance and accomplishment of the learning objectives defined for the laboratory.
8. A better knowledge of how modular teaching techniques can aid in an Embedded System Design laboratory.

9. A poster presentation in the 2016 HENAAC Conference, Anaheim, CA (Presented). Third place award in the Engineering Category.
10. A publication pending in the ASEE Zone 2 Anual Conference (paper submitted)

Bibliography

[1] M. Jiménez, *Syllabus INEL4217: Course Outcomes.* Mayagüez, PR: UPRM Electrical and Computer Engineering Departement, 2015. [Online]. Available: http://ece.uprm.edu/~mjimenez/icom4217/

[2] M. Jimenez, R. Palomera, and I. Couvertier, *Introduction to Embedded Systems using Microcontrollers and the MSP430.* Springer, 2014.

[3] W. Wolf and J. Madsen, "Embedded systems education for the future," *Proceedings of the IEEE*, vol. 88, no. 1, pp. 23–30, Jan 2000.

[4] M. Jimenez, R. Palomera, and I. Couvertier, *Introduction to Embedded Systems using Microcontrollers and the MSP430.* Springer, 2014.

[5] C. Nagy, *Embedded systems design using the TI MSP430 series.* Elsevier, 2003.

[6] E. Palacios Municio, F. Remiro Domínguez, and L. J. López Pérez, *Microcontrolador PIC16F84: desarrollo de proyectos.* México, DF:. Alfaomega;, 2009.

[7] J. H. Davies, *MSP430 microcontroller basics.* Elsevier, 2008.

[8] E. White, *Making Embedded Systems: Design patterns for great software.* O'Reilly Media, Inc., 2011.

[9] D. L. Maskell and P. J. Grabau, "A multidisciplinary cooperative problem-based learning approach to embedded systems design," *Education, IEEE Transactions on*, vol. 41, no. 2, pp. 101–103, 1998.

[10] J. W. Bruce, J. C. Harden, and R. B. Reese, "Cooperative and progressive design experience for embedded systems," *Education, IEEE Transactions on*, vol. 47, no. 1, pp. 83–92, 2004.

[11] L. Ordinez and O. Alimenti, "A constructivist approach for teaching embedded systems," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 11, no. 1, pp. 572–578, Feb 2013.

[12] C.-S. Lee, J.-H. Su, K.-E. Lin, J.-H. Chang, and G.-H. Lin, "A project-based laboratory for learning embedded system design with industry support," *Education, IEEE Transactions on*, vol. 53, no. 2, pp. 173–181, May 2010.

[13] A. Kumar, S. Fernando, and R. Panicker, "Project-based learning in embedded systems education using an fpga platform," *Education, IEEE Transactions on*, vol. 56, no. 4, pp. 407–415, Nov 2013.

[14] I. Couvertier, M. Jiménez, R. Palomera, and M. Toledo, "Integrating concepts and practice in teaching embedded systems design," in *Proceedings of the International Conference on Engineering Education (ICEE 2004), Gainesville, FL*, 2004.

[15] J. Gonzalez, P. Pomares, M. Damas, P. Garcia-Sanchez, M. Rodriguez-Alvarez, and J. M. Palomares, "The use of video-gaming devices as a motivation for learning embedded systems programming," *Education, IEEE Transactions on*, vol. 56, no. 2, pp. 199–207, 2013.

[16] U. Münz, P. Schumm, A. Wiesebrock, and F. Allgöwer, "Motivation and learning progress through educational games," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 6, pp. 3141–3144, 2007.

[17] T. Kodama, Y. Suzuki, and S. Chiba, "Development of a remote practice system for embedded system education," in *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on.* IEEE, 2010, pp. 53–58.

[18] S. Buchner and S. Jaschke, "Preparation for embedded systems laboratories the virtual workspace approach," in *Global Engineering Education Conference (EDUCON), 2013 IEEE*, March 2013, pp. 171–175.

[19] D. Soldan, J. Hughes, J. Impagliazzo, A. McGettrick, V. P. Nelson, P. K. Srimani, and M. D. Theys, "Curriculum guidelines for undergraduate degree programs in computer engineering," *Retrieved December*, 2004.

[20] L. R. Lattuca, P. T. Terenzini, and J. F. Volkwein, *Engineering Change: A Study of the Impact of EC2000: Executive Summary.* ABET, Incorporated, 2006.

[21] Electrical and C. E. Department, *Manual Informativo INEL ICOM INSO CIIC 2015-2016.* Mayagüez, PR: UPRM Electrical and Computer Engineering Departement, 2015. [Online]. Available: http://136.145.34.25/wp-content/uploads/Manual-Informativo-INEL-ICOM-INSO-INCO-2015-2016.pdf

[22] R. Kamal, *Embedded systems 2E.* Tata McGraw-Hill Education, 2008.

[23] M. S. Salerno and A. V. C. Dias, "Product design modularity, modular production, modular organization: the evolution of modular concepts," *Automotive Industries*, 1999.

[24] T. Lehtonen, *Designing modular product architecture in the new product development*, 2007.

[25] T. D. Miller and P. Elgard, "Defining modules, modularity and modularization," in *Proceedings of the 13th IPS research seminar, Fuglsoe*, 1998.

[26] D. K. Duffy and J. W. Jones, *Teaching within the Rhythms of the Semester. The Jossey-Bass Higher and Adult Education Series.* ERIC, 1995.

[27] D. Fink, "Integrated course design," *Idea paper*, vol. 42, 2005.

[28] L. D. Fink, "A self-directed guide to designing courses for significant learning," *University of Oklahoma*, vol. 27, 2003.

[29] J. W. Pellegrino, "Rethinking and redesigning curriculum, instruction and assessment: What contemporary research and theory suggests," *commissioned by the National Center on Education and the Economy for the New Commission on the Skills of the American Workforce*, 2006.

[30] R. A. Streveler, K. A. Smith, and M. Pilotte, "Aligning course content, assessment, and delivery: Creating a context for outcome-based education," *Hershey, Pennsylvania: IGI Global*, 2012.

[31] G. P. Wiggins, J. McTighe, L. J. Kiernan, and F. Frost, *Understanding by design*. Association for Supervision and Curriculum Development Alexandria, VA, 1998.

[32] J. D. Bransford, A. L. Brown, R. R. Cocking *et al.*, "How people learn," 2000.

[33] Y. Torroja, O. Garcia, T. Riesgo, and E. de la Torre, "Teaching embedded systems and microcontrollers using scale models," in *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*. IEEE, 2005, pp. 4–pp.

[34] S. Nooshabadi and J. Garside, "Modernization of teaching in embedded systems design-an international collaborative project," *Education, IEEE Transactions on*, vol. 49, no. 2, pp. 254–262, 2006.

[35] E. E. . C. S. Department, *Microcomputer Project Laboratory (6.115)*. Cambridge, MA: Massachusetts Institute of Technology (MIT), 2015.

[36] ——, *Embedded Systems (EECS149)*. Berkeley, CA: Berkeley University of California, 2015.

[37] Electrical and C. E. Department, *Embedded Systems (EE3376)*. El Paso, TX: University of Texas at El Paso (UTEP), 2015.

[38] ——, *Microprocessors (ECE344L).* Albuquerque, NM: University of New Mexico, 2015.

[39] R. Zurwaski, *Embedded Systems Handbook: Networked Embedded Systems.* CRC Press, 2009.

[40] J. Valvano, *Introduction to Embedded Systems: Interfacing to the Freescale 9S12.* Cengage Learning, 2009.

[41] Y. Meng, K. Johnson, B. Simms, and M. Conforth, "A generic architecture of modular embedded system for miniature mobile robots," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Sept 2008, pp. 3725–3730.

[42] Y. Li, "Teaching embedded systems using a modular-approach microcontroller training kit," *World Transactions on Engineering and Technology Education*, vol. 6, no. 1, p. 135, 2007.

[43] A. Nursal, "Modular embedded system design for mechatronic education 2010 ieee/asme international conference on mechatronic and embedded systems and applications," in *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, July 2010, pp. 109–112.

[44] X. Hu, M. Wang, Y. Xu, and K. Qian, "Modular design and adoption of embedded system courseware with portable labs in a box," in *Proc. World Congress on Engineering and Computer Science (WCECS*, 2012.

[45] T. Henzinger and J. Sifakis, "The discipline of embedded systems design," *Computer*, vol. 40, no. 10, pp. 32–40, Oct 2007.

[46] A. Möller, J. Fröberg, and M. Nolin, "Industrial requirements on component technologies for embedded systems," in *Component-Based Software Engineering.* Springer, 2004, pp. 146–161.

[47] E. Sikora, B. Tenbergen, and K. Pohl, "Requirements engineering for embedded systems: An investigation of industry needs," in *Requirements Engineering: Foundation for Software Quality.* Springer, 2011, pp. 151–165.

[48] J. Patarroyo, G. Beauchamp, and S.-R. Aidsa, "A methodology to teach students to implement digital controllers using embedded systems," in *2015 ASEE Annual Conference & Exposition*, no. 10.18260/p.23407. Seattle, Washington: ASEE Conferences, June 2015, https://peer.asee.org/23407.

[49] L. Jing, Z. Cheng, J. Wang, and Y. Zhou, "A spiral step-by-step educational method for cultivating competent embedded system engineers to meet industry demands," *Education, IEEE Transactions on*, vol. 54, no. 3, pp. 356–365, 2011.

[50] B. S. Bloom, *Taxonomy of Educational Objectives: The Classification of Education Goals. Cognitive Domain. Handbook 1.* Longman, 1956.

[51] R. M. Felder and L. K. Silverman, "Learning and teaching styles in engineering education," *Engineering education*, vol. 78, no. 7, pp. 674–681, 1988.

[52] A. Oosterhof, *Classroom applications of educational measurement.* ERIC, 2001.

[53] R. M. Osman, *Educational Evaluation and Testing.* African Virtual University, 2010.

[54] T. L. Kelley, "The selection of upper and lower groups for the validation of test items." *Journal of Educational Psychology*, vol. 30, no. 1, p. 17, 1939.

[55] G. F. Kuder and M. W. Richardson, "The theory of the estimation of test reliability," *Psychometrika*, vol. 2, no. 3, pp. 151–160, 1937. [Online]. Available: http://dx.doi.org/10.1007/BF02288391

[56] C. S. Wells and J. A. Wollack, "An instructor's guide to understanding test reliability," *Testing & Evaluation Services publication, University of Wisconsin. Retrieved January*, vol. 4, p. 2006, 2003.

[57] G. C. Helmstadter, "Principles of psychological measurement." 1964.

[58] C. H. McGrath, B. Guerin, E. Harte, M. Frearson, and C. Manville, "Learning gain in higher education," 2015.

[59] H. G. Colt, M. Davoudi, S. Murgu, and N. Z. Rohani, "Measuring learning gain during a one-day introductory bronchoscopy course," *Surgical endoscopy*, vol. 25, no. 1, pp. 207–216, 2011.

[60] R. R. Hake, "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses," *American Journal of Physics*, vol. 66, no. 1, 1998.

Appendices

Appendix A
Laboratory Experiment Manual

Embedded Systems Design Laboratory Manual

ICOM4217

Electrical & Computer Engineering Department
University of Puerto Rico at Mayagüez
Mayagüez, PR 00681-9000

Danilo Rojas Luis Francisco Manuel Jiménez

2016

Electrical and Computer Engineering Department
University of Puerto Rico at Mayagüez

ACKNOWLEDGMENT

The authors would like to thank Cesar A. Aceros, for his help creating the Latex template for the manual.

DISCLAIMER

Table Of Contents

Laboratory Rules

Usage of the Microprocessor Interfacing Lab facilities is subject to the abidance of the rules listed below:

1. The entrance to the facilities is reserved exclusively for students enrolled in the **ICOM 4217** course or approved projects. Any authorization for the use of facilities shall be designated by the Laboratory Director. Persons authorized to access the laboratory facilities and resources shall follow the rules and procedures established for the University of Puerto Rico system.

2. Students projects are performed in groups. Each group will be assigned a workspace with a computer and resources for the development of their project. It is the responsibility of each group to maintain and return those resources in good condition. The lab assistant will perform periodic inventory checks to ensure the integrity of loaned laboratory resources.

3. The University maintains a limited stock of resources to support project development. Resources requests can be made via a "Materials Request Form". Requests are granted subject to availability. It is the responsibility of each work group to acquire any materials that could not be provided from the laboratory stock. Materials and resources brought into the lab for the performance of a project must be removed from the premises at the end of the project performance period.

4. The laboratory is monitored 24/7 by security cameras . Anyone agreeing using the laboratory also agrees to the monitoring and recording their behavior by the security cameras.

5. Each authorized user will have access to the facilities using his or her provided access card. The access log will be used as a user registry. This registry could be used to establish responsibilities, if necessary.

6. The laboratory will remain open as long as a lab assistant is present. In his/her absence, authorized students can stop by the campus security office to request that the laboratory wooden door be opened. A student requesting such a service must present his or her student card at the office. The requester will be registered as the person in charge of the lab. If the person in charge leaves the laboratory, he or she must close the lab wooden door and notify the professor via email. If another student wants to stay in the lab, the outgoing student must transfer the responsibility by notifying the campus security office

or the professor via email. Any incident must be reported by the person in charge. The published schedule indicates the availability of a lab assistant in the facilities.

7. The consumption of beverages and/or food within the laboratory facilities is prohibited. This prohibition includes depositing waste food or drinks into the lab trash cans.

8. It is the responsibility of each group to maintain their work space neat and organized. The cutting, grinding or machining of materials that generate particulate is prohibited within the laboratory facilities.

9. Dress code: Students in the laboratory facilities should use proper attire for a sensitive electronics laboratory and particularly clothing that minimizes the generation of static electricity. Refrain from using vinyl clothing, rubber shoes (eg. Crocks) or other parts known to generate high levels of static electricity.

10. The entry of pets into the facilities is prohibited. This restriction excludes guide dogs used by blind people.

11. Removing any resource from the laboratory without written permission of the laboratory director or department director is prohibited.

12. It is forbidden to temporarily or permanently add or bring any type of resource to the lab without prior authorization of the laboratory director or department head. This rule excludes the use of laptop computers for personal use provided they are with their owner, and electronic components used in projects prototypes.

13. Accessing the network must always be done wirelessly. Plugging personal computers to the wired network is a violation of the Laboratory Regulations and will carry penalties.

14. The usage of the laboratory printer is subject to the institutional rules established for computer center printers. The system will deduct the number of printed pages from your print quota.

15. The transfer (loan) of accounts among students is strictly prohibited. If this were the case, the loaned account could be deactivated.

16. Any software installation requires prior authorization from the system administrator. The use, installation, or storage of programs or resources that violate current copyright law is not allowed.

17. Unnecessary noise within the laboratory is prohibited. Using external computer speakers is prohibited, except for projects that require so. In such cases, moderation is advised. The use of hearing aids will be permitted provided that their volume is moderate and does not disrupt the work environment.

18. The act of locking computers is limited to a maximum of ten minutes. If the computer were not unlocked before the timer expires the work session will be automatically terminated and the logged user logged-out without notice.

19. Students must respect the workspaces of their peers and refrain from assessing restricted access areas in the laboratory. Under no circumstances should a student sabotage or modify in any way the project area of other groups or access unauthorized areas.

20. The laboratory has designated seats for people who require special accommodations. Such individuals will have priority in using such resources.

21. It is the duty of every student to report any violation to the rules established herein. Violation of the dispositions contained in this regulation will be sufficient cause to initiate a disciplinary action against the offender; including denial of access to resources, removal from the facilities, and/or any other applicable legal action.

Students with special needs or requiring reasonable accomodation, please contact the laboratory coordinator, the class professor, or the laboratory teaching assistant.

Experiment 1

High-Voltage Safety

Objectives

- Understanding the concept of high voltage and its implications to the human body
- Recognizing dangerous current and voltages levels for the human body and their effects
- Computing the human body impedance and understanding its role in the levels of current flowing trough the human body
- Identifying the potential electric hazards in a laboratory space
- Learning and applying safe practices in a laboratory environment
- Learning how to apply emergency procedures in case an electric shock

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

1.1 Introduction

Common electrical and electronic devices require the use of electrical energy to operate. The very same energy that makes them work can reach levels that could become harmful or even lethal for humans. Electric and electronic devices are commonly enclosed, but in some cases, this enclosure needs to be opened to make adjustments or to perform repair procedures, exposing areas that might be subjected to harmful. Moreover, when working in the development and prototyping of new applications and circuits, such activities expose developers to the same type of risks.

This experiment has as an objective creating awareness of the levels of voltages and currents that could cause harm, how to identify hazards situations, minimize the risk of accidents, and how to responds in the event of an accident.

1.2 Terms and Definitions

The term **High Voltage** refers to electrical energy at a voltage high enough to cause injury or death. In a formal definition *"High Voltage is any voltage exceeding 1000V rms or 1000V dc with current capability exceeding 2mA as or 3mA dc, or for an impulse voltage generator having a stored energy in excess of 10mJ"* according to the *IEEE Trans.Power App . Sys., vol PAS-97, no. 6, 2243, November, 1978.* Although, in a relative sense 50Volts might not be considered strictly a high voltage, it represents the threshold where harmful effects occur in a adult body. For this reason, 50Volts is considered as a danger high voltage for the human body.

High voltages can be found in form of AC (Alternating Current), DC (Direct Current), or momentary pulsed signals. These signals can injure the human body depending on their voltage and current levels, being the AC voltage at 60Hz the worst possible voltage type and frequency for humans. At this frequency the human body is 5 times more sensitive than to direct current. Also, keep in mind that, although some voltage levels in DC are not supposed to be dangerous for the human, in AC, those levels can be fatal.

Other important definitions suitable for electrical and laboratory applications include:

- **Moderate Voltage**: Refers to voltages greater than 120 V rms or 120V dc but less than 1000V. With current capabilities of 2mA ac and 3mA dc respectively.
- **Temporary Setups**: System assembled generally for measurement over a time period that not exceeds three months.
- **Troubleshooting**: Temporarily procedure carried out to repair or diagnose problems in a device or circuit energized with any voltage level.
- **Bare Conductor**: A conductor without covering or electrical insulation.
- **Covered Conductor**: A conductor enclosed within a material not necessarily electrical insulation.
- **Insulated Conductor**: A conductor enclosed within electrical insulation material.

- **Exposed Conductor**: Refers to parts that are not suitable guarded, isolated, or insulated.
- **Enclosed**: An object surrounded by a case, housing, fence, or walls that prevents persons to enter in contact with energized parts.

1.3 Human Body Impedance

The human body has his own resistance, allowing us to interact in some cases with electricity without suffering any type of damage. But, when this resistance is overcome, a current flow can pass through the entire body causing external or internal injuries. Although, on average, the skin resistance has a value between the 1,000 ohms and 100,000 ohms, the internal body resistance is lower with values between 25 to 1,000 ohms. Table 1.1 shows the different skin resistance values depending of the part of the body and some specific conditions.

Table 1.1: Human skin resistance (*Source*: Electric Safety Manual, Berkley Laboratory)

Condition	**Resistance (Ohms)**	
	Dry	**Wet**
Finger touch	40,000 to 1,000,000	4,000 to 15,000
Hand holding wire	15,000 to 50,000	3,000 to 6,000
Finger-thumb gasp	10,000 to 30,000	2,000 to 5,000
Hand holding pliers	5,000 to 10,000	1,000 to 3,000
Palm touch	3,000 to 8,000	1,000 to 2,000
Hand around 1.5 in pipe or drill handle	1,000 to 3,000	500 to 1,500
Two hands around 1.5 in pipe	500 to 1,500	250 to 750
Hand immersed	–	200 to 500
Foot immersed	–	100 to 300
Human body, internal, excluding skin ohms	200 to 1,000	

The skin resistance depends mainly of the parameters that include:

- Area of contact
- Pressure applied
- Amount of current

- Waveform of the current (AC, DC)
- Duration of the shock
- Environmental conditions such as humidity, temperature, and pressure, among others.

The internal body resistance is affected by factors such as:

- Body mass (weight & height)
- Age
- Diseases
- Tissue type and amount

Once the factors that could affect the body resistance are known, the total body resistance can be estimated. The total resistance in the human body can be calculated as:

$$R_{total} = R_{skin(in)} + R_{internal} + R_{skin(out)}, \tag{1.1}$$

where $R_{skin(in)}$ denotes the skin resistance where the electric current enters the body, $R_{internal}$ refers to the resistance where the current flows, and $R_{skin(out)}$ is the resistance where the current leaves the body.

1.4 Current Flow Through the Human Body

When a person receives an electric shock or is electrocuted, it is because the human body works in that moment as a conductor. Whenever a person comes in contact with an energized bare conductor while also in contact with a grounded surface, a conduction path is established and current passes through his or her body.

The current can flow through the body affecting or not the organs in it's path. Although, some organs could be affected due to the current, there are some paths more dangerous than others. One of the most lethal paths is when current passes through the chest affecting the heart. This is usually the worst case scenario as it might interfere with electrical impulses of the heart making it stop. Current through the body can also cause severe injuries such as internal burns resulting the heat generated by the current flow. Table 1.2 shows different injuries caused by different current values. These values are not the same for every person due physiology and environmental factors.

Table 1.2: Electric current effects on the human body (*Source*: High Voltage Safety Manual, Colorado State University)

Effect/feeling	**Direct Current (mA)**		**Alternating Current (mA)**				**Incident Severity**
			60 Hz		**10,000 Hz**		
	150 lb	**115 lb**	**150 lb**	**115 lb**	**150 lb**	**115 lb**	
Slight sensation	1	0.6	0.4	0.3	7	5	None
Perception threshold	5.2	3.5	1.1	0.7	12	8	None
Shock not painful	9	6	1.8	1.2	17	11	Minor
Shock painful	62	41	9	6	55	37	Spasm, indirect injury
Muscle clamps source	76	51	16	10.5	75	50	Possibly fatal
Respiratory arrest	170	109	30	19	180	95	Frequently fatal
≥0.03-s vent. fibril.	1300	870	1000	670	1100	740	Probably fatal
≥3-s vent. fibril.	500	370	100	67	500	340	Probably Fatal
≥5-s vent. fibril.	375	250	75	50	375	250	Probably fatal
Cardiac arrest	–	–	4000	4000	–	–	Possibly fatal
Organs burn	–	–	5000	5000	–	–	Fatal if it is a vital organ

To have an idea of how severe a voltage shock can be, consider the following scenario. Let's suppose that our hands are sweaty and we touch with one hand an energized circuit with 50V and accidentally with the other hand a ground surface (Lethal path!). With the hands sweaty, the skin's resistance can drop to 1,000 ohms. Using the ohms law and supposing an internal resistance of 200Ω, we can calculate a current of:

$$I = \frac{V}{R_{total}} = \frac{V}{R_{skin(in)} + R_{internal} + R_{skin(out)}} = \frac{50V}{1000\Omega + 200\Omega + 1000\Omega} = 22.7mA \tag{1.2}$$

This current level as we can see in Table 1.2 can be potentially harmful for us because it can case us a cardiac arrest.

1.5 Risk Mitigation

To try to avoid electrical risks and hazards we need to take into consideration not only behavioral aspects but also our own senses. They can help us detect possible electric hazards in different ways. Typical indications, depending on the type of sense, include:

Visual indicators:

- High voltage warning signs & labels
- Flashes, arcs, corona discharge
- Cables with damaged insulation
- Burn marks on circuit
- Tripped breakers or GFCIs
- Dim or flickering lights

Audible indicators:

- Sizzles
- Buzzes

Tactile indicators:

- Tingling sensation
- Hot or burning wires, connectors, junctions, or other components

Odor indicators:

- Smell of burning wire or other components

Also, observing safety precautions during the assembly, testing, and debugging process, helps in the prevention of possible hazards. Some tips that could help you during the laboratory work and prototype construction are mentioned below:

- Before energizing your circuit:
  - Make all connections and configurations
  - Have someone else inspect your circuit
  - Locate all breakers and workstation power switches
- When energizing your circuit:
  - Observe: For arcs, sparks, smoke, or sings of heat

  - Listen: For cracking sounds, pops, or hisses
  - Smell: Odor to smoke or burning electronics
  - check: Current and voltage levels into the power supply. Recall the signs of a short circuit. $I \to \infty$ and $V \to 0$.

- Work with energized circuits only for debugging and testing purpose
  - Always be careful

General Measures include:

- Maintain an illuminated and organized workstation
- Keep the floor in your workspace completely dry
- Avoid exposed connections
- Set up your work area away from possible grounds that you may accidentally contact
- Do not reach for something you cannot see (Within an energized circuit or panel)
- Avoid working alone
- Never ignore high voltage warning signs
- Never enter alone into an area containing exposed electrical energy sources
- Wear personal protective equipment associated with the voltage you are handling
  - Goggles
  - Gloves if needed, specially to work with high voltage
  - Insulated shoes and tools
  - Do not wear conductive jewelry
- Use the buddy system
  - It is best to work with someone that has knowledge about what you are doing
- One hand in a pocket rule

  - Never touch an energized circuit with both hands

- Know your equipment's limitations
  - Do not exceed your equipment's insulation capabilities
  - Locate your equipment's power switches
  - Use and electrostatic discharge wrist wrap
- Connect/disconnect any test leads with the equipment unpowered and unplugged. Use clips leads or solder temporary wires to reach cramped locations or difficult to access locations
- Perform as many tests as possible with the power off and the equipment unplugged
- Use only the test instruments, and insulated tools rated for the voltage and current specified
- Keep all electrical cords away from areas where they may be pinched, such as off the floor, out of walkways, and out of doorways.
- Know the emergency procedures to follow in case of an accident

1.6 Emergency Response Procedure

In case of an emergency, there are some step procedures that you have to follow in order to guarantee, not only the life of the affected person, but also your own life. Follow this steps in order:

- Shut off the power source
  - Breakers, power strips, etc
  - Never touch a victim with bare hands before shutting off power the source. Use a nonconductive rod or something similar
- Call for help
- Pry the victim from the circuit
- Use CPR if you are trained or find help

Experiment 2

IDE, GPIOs, and LCD

Objectives

- Understanding the process of assembling, debugging, and executing a program with your microcontroller's IDE
- Identifying the basic structure of an assembly or C program
- Identifying and understanding your microcontroller architecture and main features
- Using I/O ports to interface the MCU to different electronics components
- Interfacing and using an LCD with a microcontroller

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 2.1: Bill of materials for completing Lab. 2

Item #	Qty	Description	Reference
1	1	Development board	
2	1	IDE application	
3	1	LCD display: 2 lines, 16 characters	W/HD 44780 Controller
4	2	Light Emitting Diode	5mm Red LED
5	2	1/4W Carbon fill resistor	330 Ω
6	2	1/4W Carbon fill resistor	4.7 KΩ
7	2	Momentary switch	Pusbutton

2.1 Introduction

2.1.1 Microcontroller IDE

IDE stands for Integrated Development Environment also called Integrated Design Environment or Integrated Debugging Environment.

For all upcoming experiments, the **IDE** of your choseen microcontroller will be used as the compiler, assembler, linker, and code debugger. Some Microcontroller Units (MCU) have more than one IDE choice. Choosing an IDE depends on the main language used to program your microcontroller (assembly, C or another language). Although the main languages to be used will be assembly and C, some IDEs allow you to add compilers for different languages.

Normally an IDE for programming microcontrollers consist of the following tools:

- **A code editor**: The code editor is a type of text editor used to enter and modify the source code in a programming language. It is basically the text processor in an IDE and provides, in the majority of the cases, cross references to the elements in the code.

- **A compiler or/and assembler**: A compiler is a program that translates/-transforms a source code (written typically in high-level) to a target code (typically in low-level). The target code in our case refers to an executable code in machine language that governs the behavior in our MCU.

- **A debugger**: Is a software program used to test other programs and find bugs (errors). The debugger warns the programmer about what types of errors it finds and the exact line number where they are found. Also, it allows to run the source code step by step to help determining execution and logic errors.

- **A download tool to program the MCU flash memory**.

- **Built-in automation options**.

When an IDE is selected, it is important to know the maximum limit of code you can write on it. Most demo versions of IDEs limit the code size to only a few kilobytes of length. Also, it is important to know if your IDE supports the type of JTAG or programming tool available in your MCU. The JTAG interface is used for debugging your code in your embedded system.

2.1.2 General Purpose Input/Output (GPIO)

To communicate with the external world, microcontrollers use input and outputs pins known as **General Purpose Input/Outputs (GPIOs)** that are part of the Peripheral Subsystem. GPIOs have the capability of exchanging information in the form of digital signals (0 or 1) to other devices or systems.

GPIO pins in an MCU are grouped as ports commonly made of 8 pins. Each I/O pin can be programmed independently as an input or output. Generally, each port has a couple of associated registers that allow for configuring the function of the pin (input or output) and determining the logic level that it is reading-in or sending-out. Some ports have specialized capabilities to perform other functions such as internal and external oscillator options, timers functions, hardware for pulse width modulation (PWM), watchdog timer, USART, SPI, I^2C, data converters, and brownout reset circuitry, among others.

An input port always transfers data towards the CPU. Input ports can be either buffered or latched. A buffered input only reads the current status present in the input. A latched input uses a latch to hold the input data until it is read by the CPU. Output ports are in most cases is latched, holding the output data until the next output operation is executed. Figure 2.1 shows the general structure of an I/O pin. The Port Direction latch is used to select the pin direction (through the P_Dir bit) and the latch Output is used to determine the output value (through the P_Out bit). The P_In signal is used to read the data present in the pin. $\overline{\text{Dir\_En}}$ and $\overline{\text{Data\_En}}$ allowing for a physical connection to the processor bus line.

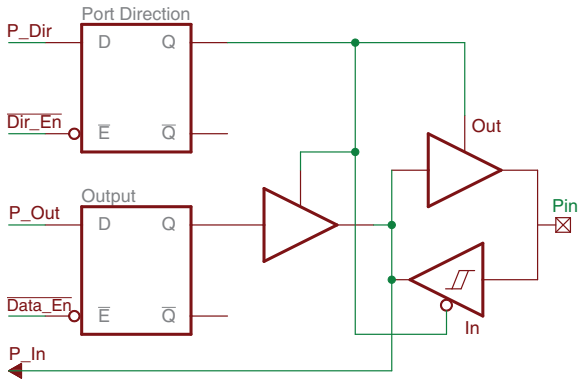


Figure 2.1: Basic structure of an Input/Output pin driver (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

GPIOs have electrical characteristics that define the currents and voltages that can be safely managed by the pin as either input or output. These characteristics and

descriptions include:

- V_{IL} **Input-low voltage**. Establishes the maximum voltage level that can be interpreted as a low by the pin input buffer.
- V_{IH} **Input-high voltage**. Represents the minimum voltage level that can be interpreted as a high by the I/O pin.
- V_{OH} **Output-high voltage**. The voltage level used to represent a logic "High" on an output pin.
- V_{OL} **Output-low voltage**. The voltage level used to represent a logic "low" on an output pin.

These characteristics must be taken into consideration when you are designing interfaces to be connected to I/O pins. Failing to observe such limits can cause malfunction or irreparable damage to the port electronics. See class's book Section 8.2.1 (Electrical Characteristic in I/O pins) for a detailed and extra explanation about electrical pins characteristics.

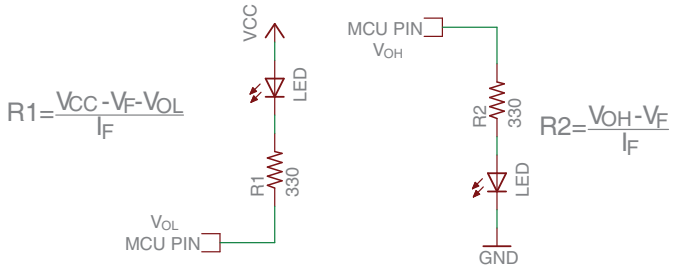
2.2 Basic Exercises

2.2.1 Blinking LED

Make an LED turn On and Off intermittently. The delay time to keep the LED On and OFF shall be about 100ms.

Follow the steps outlined below:

1. Identify the Port and Pin of your MCU that will be used to connect to the LED.
2. Connect the LED to the pin in one of the two possible configurations presented in Figure 2.2. Be sure to use a resistor that limits the current through the LED to a value below its current capacity. Take into account that when the LED is connected as illustrate in Figure 2.2(a) the MCU pin is sinking current, while when connected in as in Figure 2.2(b) the MCU pin is sourcing current. Be sure also, to not exceed the GPIO pin current limitations in terms of the maximum current the pin can source or sink.
3. Open your IDE.



(a) Pull-Up LED (b) Pull-Down LED

Figure 2.2: Pull-up vs. pull-down LED

4. Set the selected port and pin as output. If your are using configuration in 2.2(b), you should send a logic '1' to your pin to turn On the LED and '0' to turn it Off. Use a delay value that allows you to see the LED blinking. Use the inverse logic if you decided to use the configuration Figure 2.2(a). You can use the pseudocode in Listing 2.1 as a guide.

Listing 2.1: Blinking LED Pseudocode

```
;-----------------------------------------------------------
;   Program Start
;   INIT RESET VECTOR
;   INIT STACK POINT, WDT
;-----------------------------------------------------------
Port_bit = output               ;Set port pin as output
Port_pin = 0                    ;Initialize pin to ‘‘low’’

While   TRUE
    Port_pin = NOT(Port_pin)    ;Toogle pin
    wait = 2000h
    While wait>0                ;Delay loop
        wait = wait - 1
    Endwhile
Endwhile
;-----------------------------------------------------------
```

5. Assemble or compile your code and verify that it does not contain errors.

6. To run the code, select 'Run → Debug active project'. The progress information is displayed while the code downloads. Once the download is completed, the debug perspective shall open automatically.

7. Select 'Run' from the 'Run menu' and verify that the LED is blinking.

8. Pause the running programm and use the debugger options to place a break point in the instruction where the pin is toggled. Run the program, and see how the LED state changes.

9. Using the debugging tool to run your code step by step and see the hardware's reaction. Examine the contents of the count register and the port output register as you step through your program.

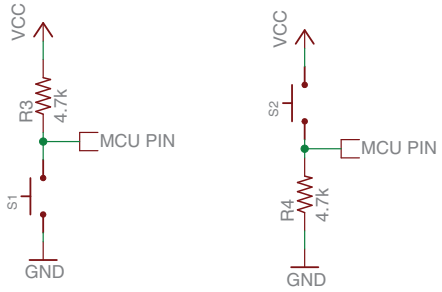
2.2.2 Polling a Switch

Read the state of an input Pin and make your LED turn On and Off accordingly.

Follow the steps outlined below:

1. Identify the Port and Pin of your MCU that will be used to connect the pushbutton.

2. Connect the pushbutton to the selected pin according to the configuration in Figure 2.3. Taking into account that when the pushbutton is connected as shown in Figure 2.3(a) the pin it reads a logic '1' while not depressed and "0" when depressed. The connection in Figure 2.3(b), it will revert the logic values read.



(a) Pull-Up Resistor (b) Pull-Down Resistor

Figure 2.3: Pull-up vs. pull-down resistor

3. Open your IDE.

4. Set the selected port and pin as input. In configuration (a) you should receive a '0' when the pushbutton is depressed, in that moment you have to turn the LED On and keep it in that state while the pushbutton is depressed. Once the

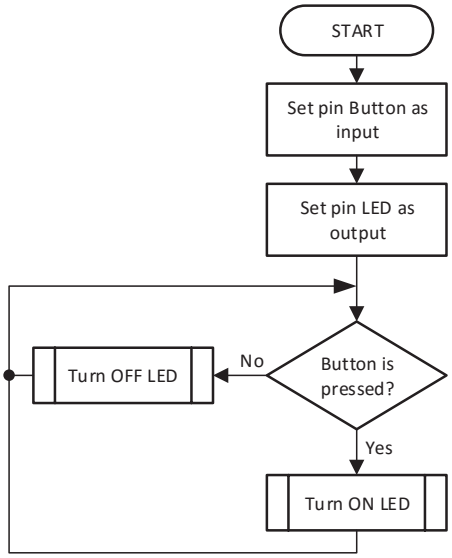


Figure 2.4: Polling a switch flow diagram

pushbutton is released, it should turn Off the LED. You can use the flowchart in the Figure 2.4 as a guide to write your code.

5. Compile your code and verify it does not contain errors.

6. Run your code and verify if the pushbutton is working.

7. Now, connect a second pushbutton and LED to your MCU. For the second pushbutton use configuration (b) and repeat from step 4, making the respective changes in the code for working with this pushbutton configuration.

2.2.3 LCD Configuration

Connect and configure the LCD (LM016L) to work with your MCU.

Follow the steps outlined below:

1. Using the LCD part number search on the web for the LCD's datasheet. Open it and find the device timing diagram.

2. Try to understand the signal sequences, commands, and timing metrics for your LCD and verify it's requirements. Ask your instructor in case of difficulty understanding the datasheet.

3. Connect your MCU to the LCD according to the following block diagram (Figure 2.5). Read the datasheet in order to get a better understanding of the LCD pins meaning.

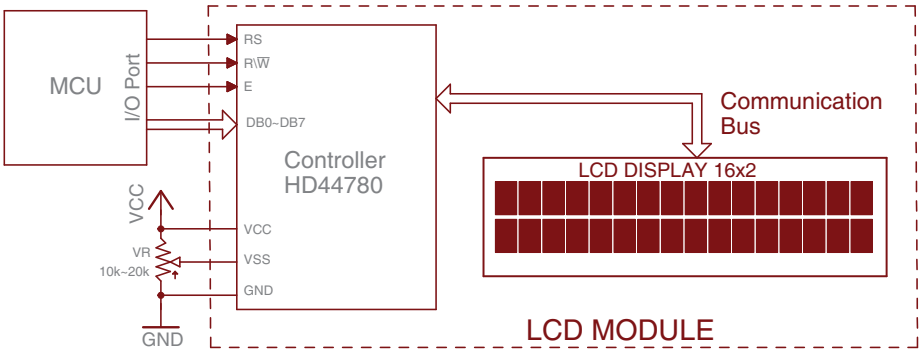


Figure 2.5: LCD block diagram connection

4. Open your IDE.

5. First, write a code to initialize the LCD with: Display ON, two line, 5x8-dot character font,and blinking cursor position character. You can use the flowchart presented in Figure 2.6 as a guide to write your code.

6. Compile your code and verify it does not contain errors.

7. Run your code and verify if the LCD is working as expected. At this point, you should see a blinking cursor on the LCD.

8. Create subroutines for each one of the LCD commands. The subroutines have to perform the following operations:

   - Clear the LCD
   - Set the Cursor to a Position
   - Write a Character
   - Write a Command
   - Write a Message (Use write a character function)

9. Test all your LCD subroutines.

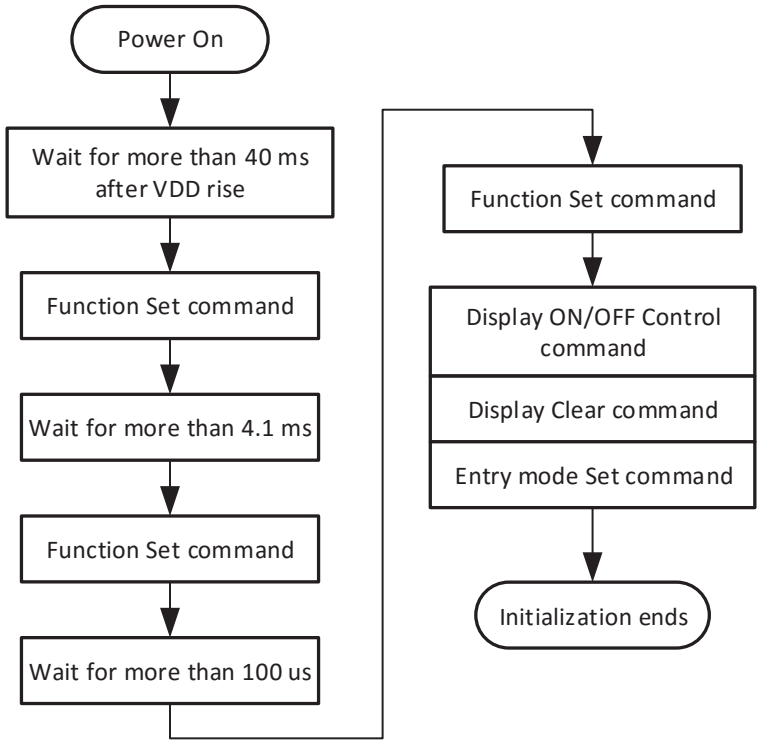


Figure 2.6: LCD initialization procedure (*Source*: HD44780U (LCD-II), Hitachi)

2.3 Complementary Tasks

2.3.1 Scrolling List

The activity consists of generating a circular scrolling list of messages and display them on the LCD. The list must consist of a minimum of 16 messages. For scrolling through the list you shall provide two pushbuttons connected to the MCU as shown in Figure 2.7. Two consecutive messages have to be displayed at the same time on the LCD; use the first line of the LCD for the first message and the second line for the following message.

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made personally in the lab and including the hardware and software components needed

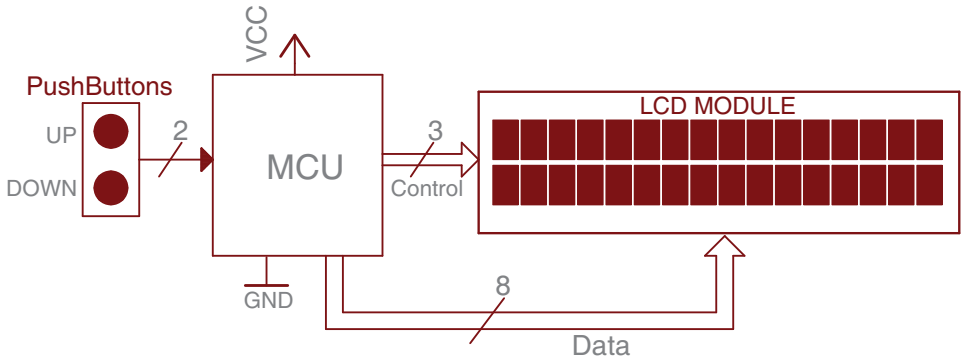


Figure 2.7: Scroll list connection diagram

for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with relevant component calculations (current, voltage, timing values, etc.)
- Code listing with comments.
- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Experiment 3

Interrupts, Switch Debouncing, and Keypad

Objectives

- Understanding the bouncing phenomena and the issues related
- Identifying the main differences between hardware and software debouncing techniques
- Understanding how an interrupt process is carried out in an MCU
- Using interrupt to read keys
- Interfacing and using keypads with a microcontroller

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 3.1: Bill of materials for completing Lab. 3

Item #	Qty	Description	Reference
1	1	Development board	
2	1	IDE application	
3	1	LCD display: 2 lines, 16 characters	W/HD 44780 Controller
4	1	1/4W Carbon fill resistor	220 Ω
5	1	1/4W Carbon fill resistor	2.7 KΩ
6	1	1/4W Carbon fill resistor	3.3 KΩ
7	2	1/4W Carbon fill resistor	4.7 KΩ

Table 3.1: Continued

8	2	1/4W Carbon fill resistor	12 KΩ
9	1	Polarized electrolytic capacitor	4.7 uF
10	2	Momentary switch	Pusbutton
11	1	3 columns by 4 rows Buttons array	Keypad 3x4
12	2	Optoswitch	RPR-220
13	1	Schmitt trigger array	74LS14N

3.1 Introduction

3.1.1 Interrupts

An interrupt is an asynchronous signal produced by an external or internal event in a device that generates an interruption in the execution of a program. At the moment when an interrupt is executed, the processor executes a jump to an interrupt handler routine defined by the programmer. This routine is responsible for serving the interruption. Once the interruption is served, the processor returns to the execution of the interrupted program in the same position where the interrupt was executed. For this reason, interrupts provide one of the most useful features in microprocessors.

Interrupts provide an efficient mechanism to handle the service request from peripheral devices and external events in a computer system, allowing for a much more efficient use of the CPU when compared to polling. In addition, interrupt servicing also provides the following advantages:

- **Compact and modular code**: An Interrupt Service Routine (ISR) induces software modularity and software reusability.
- **Reduce energy consumption**: As ISRs lead to less CPU cycles, this reduces the energy consumed by the application.
- **Faster response time**: Provide a quick response to the triggering event.

To configure interrupts in a typical MCU it is important to follow these steps:

- **Setup the Stack**: If you are using assembly language, you will need to allocate stack space in memory and initialize the stack pointer (this is also necessary when you use call instructions). You do not have to do this if your MCU has a hardware stack or if you are programming in C language (the compiler makes this for you).

- **Write the ISR**: This is the code executed by the CPU to serve the interrupt. Avoid loops or calling subroutines from inside an ISR. Just write a simple and short code. It is important when you use ASM to write the ISR to keep register transparency (push all registers used in the ISR onto the stack at the beginning and pop them at the end of the ISR). If you are using C language the compiler does this for you.

- **Set-up the interrupt table**: Once your ISR is written, enter the ISR location in the interrupt table. This is how the CPU will know where the ISR is located. All MCUs have a table with entries for each interrupt source. In assembly, all you need to do is take the label from the ISR and write an absolute jump instruction to this label in the corresponding table entry.

- **Enable Interrupts**: Make sure that you have enabled the CPU global interrupt flag and the particular enable flag of the service. First set the flags corresponding to each device and then enable the CPU global interrupt flag. Many devices require re-enabling their interrupt flags at the end of the ISR to ensure it can be triggered again.

Interrupts are mainly triggered by hardware events knows as **hardware interrupts** such as a push-button depression, a threshold reached, and timer expiration. Their occurrence is asynchronous, making it impossible to know when it may occur. In contrast, **software interrupts** are predictable and become part of the normal program sequence. These are triggered by software instructions within a program.

Due to the number of different mechanisms able to trigger produce an interrupts in an MCU, they have levels of priorities to determine which interrupt will be served first in the case of two or more interrupts request simultaneously arrive to the CPU. See class's book Section 7.1 (Fundamental Interrupt Concepts) for a detailed and deep explanation about Interrupts.

3.1.2 Switch Bouncing

Switch contacts are usually made of springy metals that are forced into contact by an actuator. When the contacts strike together, their momentum and elasticity act together, causing a bounce phenomena. The result is rapidly sequenced electrical pulses instead of a clean transition from 0 to a logical 1 as we can see in Figure 3.1. The problem may occur in switch closures and openings. The maximum time taken by the contacts in a switch to reach the steady state, is called switch bounce time.

Bouncing causes that a single switch throw be interpreted as multiple operations, causing, in many cases, incorrect system operation particulary when managed by

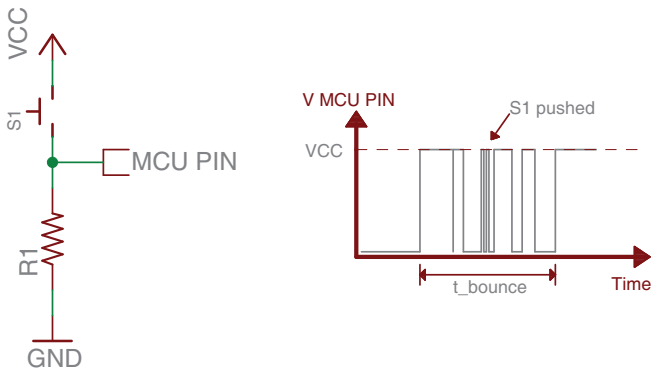


Figure 3.1: Bouncy behavior of a mechanical switch (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

interrupts. However, even when a polling technique is used to read the switch, the process might be affected by bouncing if the polling interval were shorter than the bounce time. To try to avoid or reduce this problem, hardware or software techniques can be implemented.

A **hardware technique** is implemented through the insertion of circuit components such as filters or some form of digital delay to suppress the transient pulses. The implementation of these techniques depends mainly on the type of switch used and the characteristics of the application. The most commonly used techniques include:

- An SR debouncing circuit: Consists of a set-reset (SR) latch between the switch and the digital pin.
- An RC debouncing circuit: Is a cost-effective solution that consists of a Resistor-capacitance network to implement a delay in the switch line.
- An IC debouncer circuit: Consists of a commercial integrated-circuit (IC) such as the MC14490 (Hex contact bounce eliminator). This IC contains six independent debouncing circuits for an equal number of switches.

Software techniques are implemented through the use of extra code lines (subroutines) instead of external components. This code uses CPU cycles to remove the bouncy portion from the switch signal. The most common used techniques include:

- A polling debouncer: This is a simple technique that polls the switch port with a constant polling period longer than the expected switch bouncing time.
- A counter debouncer: This technique consists of assuming the contacts have settled if they have not bounced for a certain number of samples.

See class's book Section 8.3 (Interfacing Switches and Switch Arrays) for a detailed and deep explanation about the bounce phenomena in switches and the techniques used to minimize its effects.

3.2 Basic Exercises

3.2.1 Read a Key Using Interrupts

Read the input state of a pin through an interrupt service routine. The interrupt has to increment the value of an internal variable. This value must be displayed constantly on the LCD screen.

Follow the steps outlined below:

1. Use the set-up developed in Experiment 2 that connects an LCD screen and a switch to build your circuit.
2. Verify if the I/O port, where the switch is connected, has interrupt capabilities. If not, move the switch to an interrupt capable input port.
3. Open your IDE.
4. Create a main program to setup and initialize the stack pointer (if necessary).
5. Write the code for your ISR. The ISR only needs to increment the value of a (global) variable each time it is executed. Remember to make your ISR register transparent. You can use the flowchart in Figure 3.2 as a guide to writing your code.
6. Identify which entry in the MCU jump table corresponds to the port where the switch is connected. Use the label of your ISR filling the interrupt table entry.
7. Insert instructions in your main program to enable interrupts: To do so, first, clear the interrupt flag, enable the PORT interrupt, and then enable the CPU global interrupts.
8. Modify your main program so that every time the global variable is incremented, it's value is displayed on the LCD. You can use the flowchart in Figure 3.3 as a guide to write your code.
9. Compile your code and verify that it does not contain errors.

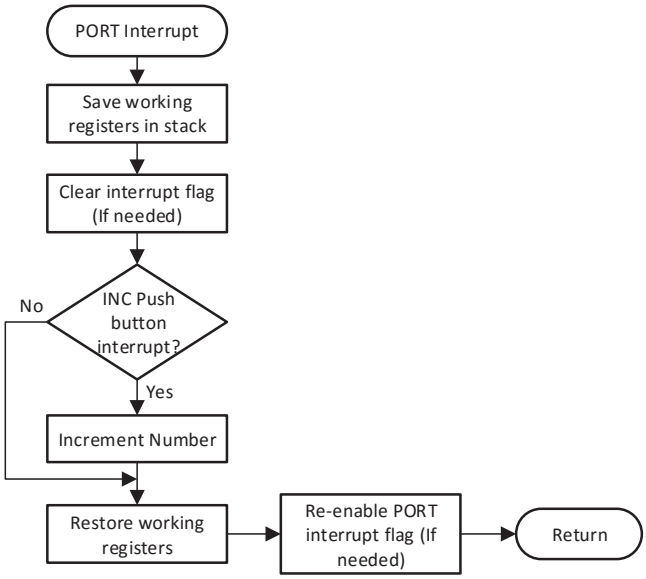


Figure 3.2: ISR flowchart

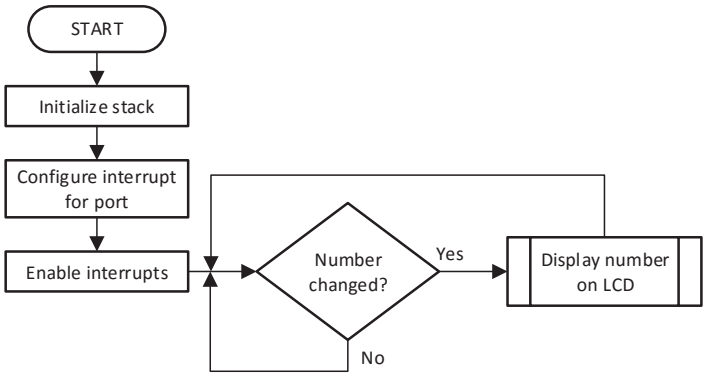


Figure 3.3: Flowchart for the main loop in read a key using interrupts

10. Run your code and verify if the number is incremented when you depress the pushbutton.

3.2.2 Hardware Debouncing

Read the input state of a pin using a hardware debouncing technique.

Follow the steps outlined below:

1. Assemble the circuit shown in Figure 3.4, with R1=2.7KΩ, R2=3.3KΩ and C1=4.7uF. These values were chosen assuming that the push button used, has a bouncing period of about 20ms. Take into account that the inverter buffer has a Schmitt triggered input. If your MCU has Schmitt triggers in it's input ports, the inverter is not needed; otherwise, provide one externally. See class's book Section 8.3.7 (Hardware Debouncing Techniques) for a detailed explanation in how calculate the resistors and capacitors values for the RC debouncing circuit.

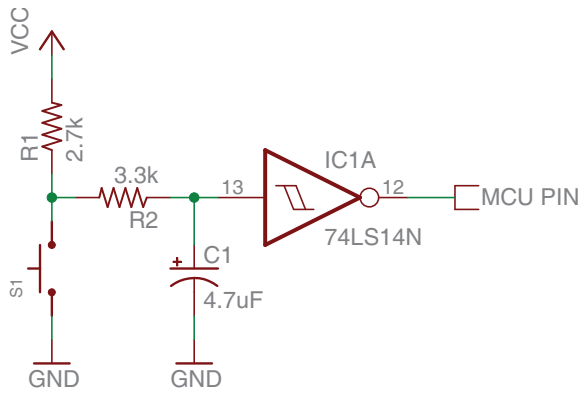


Figure 3.4: Schematic for hardware debouncing circuit

2. Replace the switch used in the previous exercise with the circuit shown in step 1.

3. Open your IDE and run the code created in the previous exercise.

4. Verify if the bounce effect of the switch disappears. If not, determine new values for R1 and R2 using a larger capacitor C1.

3.2.3 Software Debouncing

Read the input state of a pin using a software debouncing technique.

Follow the steps outlined below:

1. Restore to the circuit used in the first Basic Exercise (Pushbutton without hardware debouncing).

2. Open your IDE.

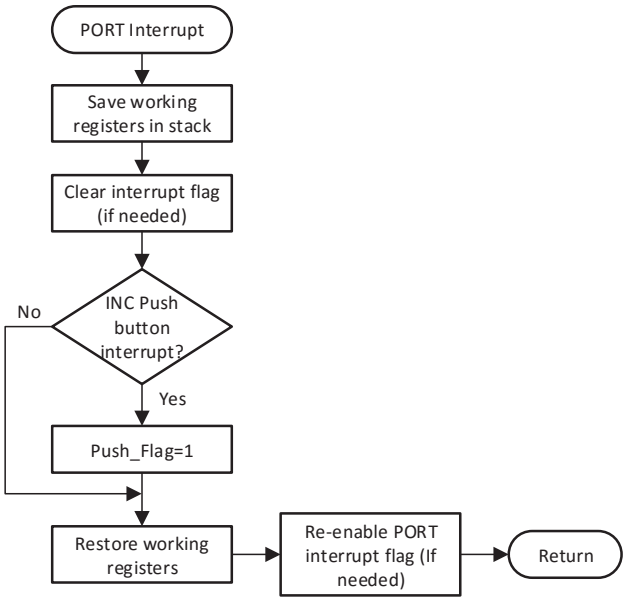


Figure 3.5: Software debouncing ISR flowchart

3. Write a code for your ISR to set a flag when an interrupt is generated. You can use the flowchart in Figure 3.5 as a guide.

4. Modify your main program so that when it confirms that the Push_Flag is set, it waits 30ms and then resets the Push_Flag. Finally, the program has to increment the value and display it on the LCD. You can use the flowchart in Figure 3.6 as a guide to write your code. This routine can be optimized using a timer interrupt to count the 30ms. You will learn this technique in the next experiment.

5. Compile your code and verify that it does not contain errors.

6. Run your code and verify that the number is incremented when you depress the pushbutton.

3.2.4 Reading Keypads Through Interrupts

Read the input state of a keypad using a scan algorithm and display it on the LCD. Follow the steps outlined below:

1. Identify the ports and pins of your MCU that will be used to connect to the

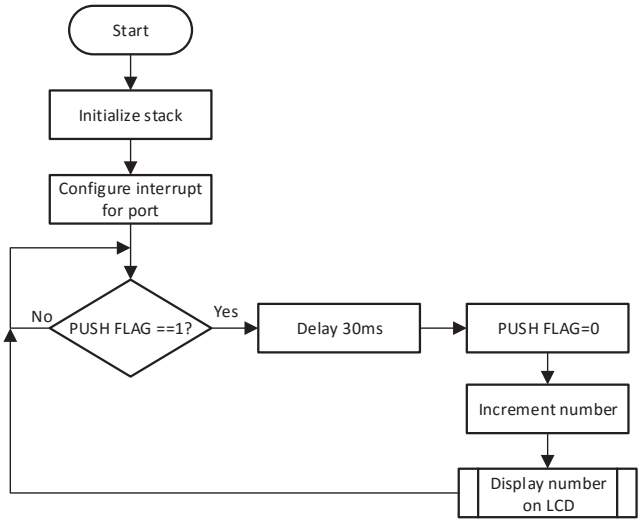


Figure 3.6: Flowchart for the main loop in software debouncing

keypad. Be sure that the pins used in the columns (inputs) have interrupt capabilities. If not, move the connections to an interrupt capable input port.

2. Connect the keypad according to the block diagram shown in Figure 3.7. You can configure and use the pull-down resistors of your MCU instead of the external resistors (You have to verify if your MCU has this capability).

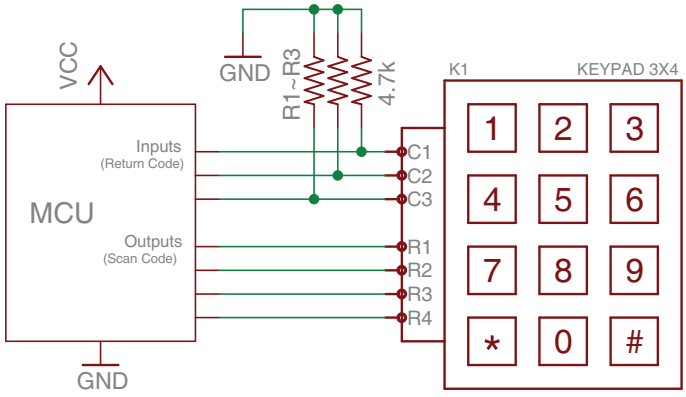


Figure 3.7: Block diagram for keypad connection

3. Open your IDE.

4. Set the pins connected to the rows as outputs and the pins connected to the columns as inputs.

5. Write a code for your ISR which once a key has been depressed performs the scanning keypad algorithm to read the key. You can use the flowchart in Figure 3.8 to write the scanning algorithm. Read the note section at the end of this exercise to have a better understanding on how the scanning method works.

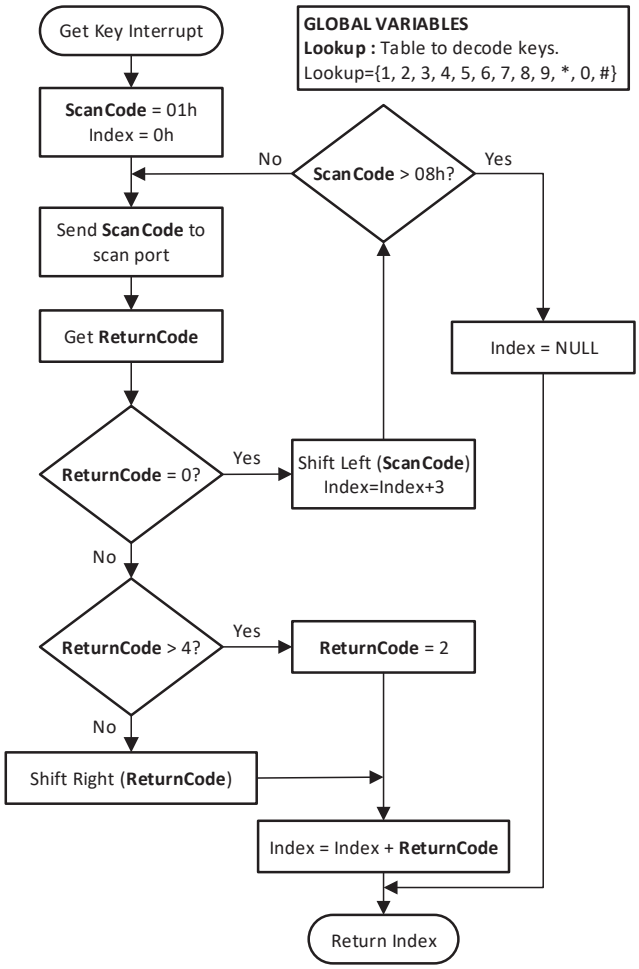


Figure 3.8: Read keypad Flowchart (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

6. Insert instructions in your main program to enable interrupts: PORT interrupt and CPU global interrupts.

7. Initialize the port configured as output to have its lines in high. This procedure will ensure the activation of the interrupts independently of the key depressed. Be sure to maintain the output pins in high once the interrupt have been served.

8. Write a main code to display the keys obtained from the interrupt on the LCD. Each new number has to be displayed on the LCD without erasing the previous number.

9. Write a function for the "$*$" key. This function has to clear the LCD.

10. Write a function for the "#" key. This function has to change the line in the LCD in which the numbers are being displayed.

11. Compile your code and verify that it does not contain errors.

12. Run your code and verify the functionality of your keypad.

Note: The scanning method for keypads consists in setting a scan code with a "Logic high" to the row (Rx) we want to scan and "logic low" the rest of the rows (ScanCode). All columns are read at once (ReturnCode). If a logic 1 is detected in a particular column line (Cy) it means that the key in position (Rx, Cy) is depressed. At the end of the cycle, the returned "Index" will contain a binary number between 00h and 0Bh that represent the key depressed in the keypad. Where the binary numbers represent the keys in the order "123456789*0#". The "Null" character is assigned by the user. These steps are sequentially repeated for each column until the entire keypad is scanned. If several keystrokes are expected, the scanning algorithm is executed in a loop until all keystrokes are received. This explanation assumes pull-down resistors are connected in the return lines. See class's book Section 8.3.11 (Interfacing Keypads) for a detailed explanation about the scanning algorithm.

3.3 Complementary Tasks

3.3.1 Scrolling List With Wheel

The activity consists of generating a scrolling list of messages and display them on the LCD (similar to the Experiment 2 but using a scrolling wheel instead of switches). To detect the movement direction, build an encoder wheel (an optical encoder can convert the angular movement and direction of the wheel into a set of digital pulses). For the wheel encoding pattern use the diagram shown in the Figure

3.11 and affix the pattern to one side of the wheel. Mount the wheel in a base as shown in Figure 3.9. Connect two opto-switches aligned vertically with the two sets of marks in the wheel. You can use the schematic in Figure 3.10 as a reference to connect the opto-switches. Also, investigate how to read a quadrate encoder using a look up table technique or read the Appendix A at the end of the document.

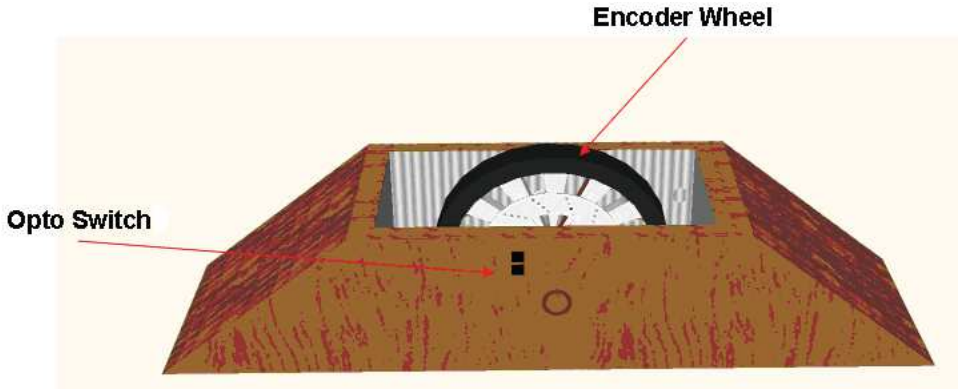


Figure 3.9: Suggested mounting base for encoding wheel

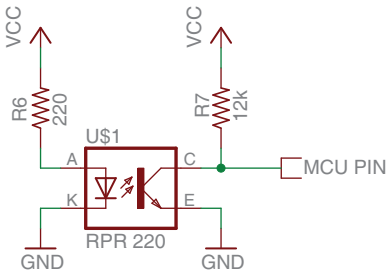


Figure 3.10: Schematic for hardware debouncing circuit

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made personally in the lab and including the hardware and software components needed for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

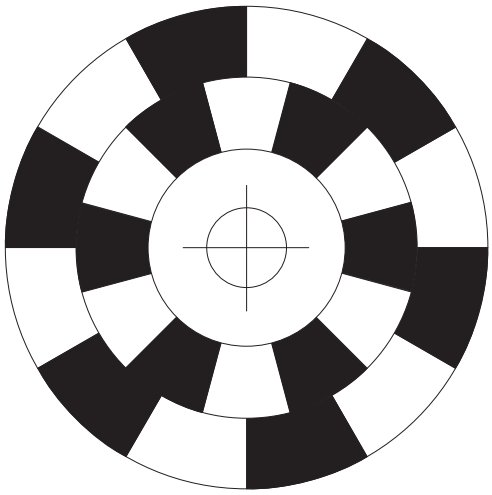


Figure 3.11: Layout of quadrature encoding patterns to be attached to the wheel

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with relevant component calculations (current, voltage, timing values, etc.)
- Code listing with comments.
- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Experiment 4

Timers and LEDs

Objectives

- Understanding the uses of timers in embedded applications
- Identifying and understand timer architectures and operating modes
- Configuring and using the timer modules
- Interfacing 7-segment displays to microcontrollers
- Implementing software techniques to display information in 7-segment displays modules

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 4.1: Bill of materials for completing Lab. 4

Item #	Qty	Description	Reference
1	1	Development board	
2	1	IDE application	
3	1	Dual 7-Segment Common Anode	DA56-11EWA
5	10	1/4W Carbon fill resistor	220 Ω
6	3	1/4W Carbon fill resistor	1 KΩ
7	1	1/4W Carbon fill resistor	4.7 KΩ
8	2	1/4W Carbon fill resistor	12 KΩ
9	1	Momentary switch	Pusbutton
10	2	Optoswitch	RPR-220

Table 4.1: Continued

11	1	Piezoelectric buzzer	Buzzer
12	2	BJT PNP transistor	2N3906

4.1 Introduction

4.1.1 Timers

In its most basic form, a timer is a counter driven by a know clock signal that increase its count with each clock cycle. When the count reaches its maximum value $(2^n - 1)$ the timer generates an overflow signal and restarts counting at 0. The overflow signal can be polled by software or used to trigger an interrupt request (Timer Overflow). The Figure 4.1 shows the basic structure of a timer. Timers are important in MCU systems because they can be used to: implement time-bases, count time between events, an to develop real-time clocks, watchdog timers, pulse-width modulators, and baud rate generation, among many others applications. Microcontrollers may include one or more configurable timer modules among their peripherals.

Timer modules can be found in different number of bits such as 8-, 16-, 24-bit, etc. The number of bits in the timer's counter determines the maximum value it can count to, i.e., the maximum value the timer count register can hold.

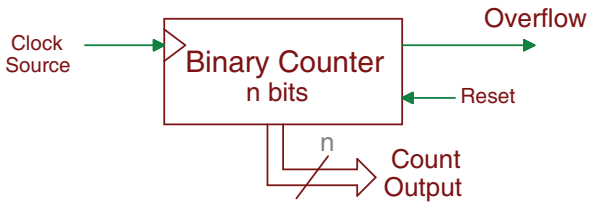


Figure 4.1: Timer overflow structure

As CPU clock frequencies are considered high for most practical applications, timers also include a pre-scaler. A pre-scaler is just a chain of flip-flops that can divide the source clock frequency by values specified through a configuration register. Most typical values are 1, 2, 4, and 8, although this might change from one timer to another. Pre-scalers are useful when you need to extend the length of time between timer overflows.

Another important part of a timer module is the terminal count register. This register, typically of n bits as the timer's binary count, can be loaded with any value $\leq 2^{n-1}$. When the timer's Binary Counter reaches *Compare Register*, a reset signal

is generate that restart the binary counter and a *Top* count signal is generated. The *Top* signal can be polled by software or configured to generate an interrupt request. This is a very useful feature in a timer that can be used for many applications, such as generating periodic signals or pulses of predetermined width. Usually, timer modules include a clock multiplexer that allows to select a clock source from a set of internal or external clock sources through a selector signals (clock source selector). Figure 4.2 shows a complete timer block diagram with both, pre-scaler and terminal count registers.

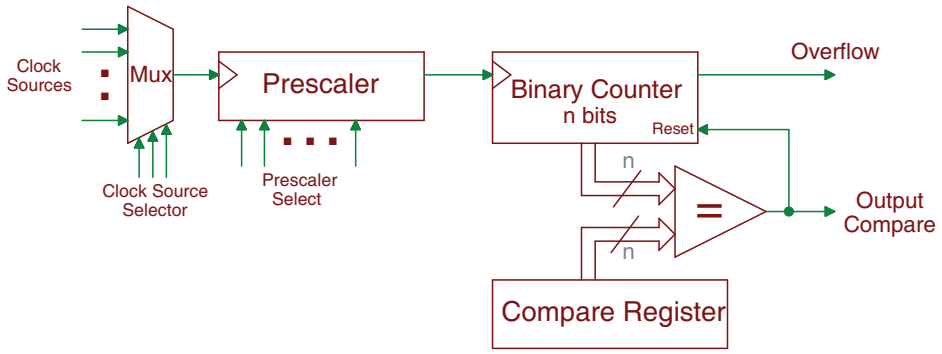


Figure 4.2: Timer basic structure (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

Timers have two basic modes of operations used in the majority of application: **Event counter** and **Interval timer**. When operated as an event counter, a timer simply counts the number of events it detects in its clock input. This clock signal does not necessarily have a periodic behavior. But, when the input clock signal is periodic with a frequency f, the timer can be used to measure time intervals between two events See class's book Section 7.4.2 (Fundamental Operation: Interval Timer vs. Event Counter) for a detailed explanation. Furthermore, if the frequency is f Hz, then the period will be:

$$PERIOD \qquad T = \frac{1}{f} seconds \tag{4.1}$$

Therefore, when the counter shows k pulses, it has registered a duration of $kT = k/f$ seconds.

4.2 Basic Exercises

4.2.1 Timer by Polling

Produce an audible sound using delays generated by polling the timer's Top count flag. Read the note section at the end of this exercise to understand other timer architectures.

Follow the steps outlined below:

1. Connect the buzzer according to the schematic shown in Figure 4.3. Choose R to not exceed your MCU's pin current capacity.

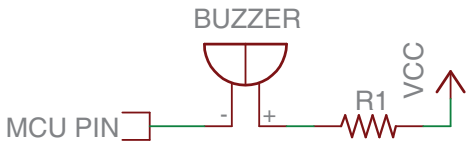


Figure 4.3: Schematic for buzzer connection

2. Open your IDE.
3. First, calculate the compare register value that produces a delay to generate an audible frequency (f=1000Hz). Take into account in your calculation that the signal to be produced must have a duty cycle of 50%.
4. Look into the architecture of your MCU Timer to determine the appropriate timer operation mode to generate the above frequency, and produce a code to configure the timer in that mode.
5. Next, load the compare register value calculated into the timer's configuration registers required.
6. Write a code that continuously is checking the Top count flag and when it is detected, toggle the buzzer pin. You can use the flowchart in the Figure 4.4 as a guide to write your code.
7. Compile your code and verify that it does not contain any errors.
8. Run your code and verify if the buzzer produces an audible sound.
9. After the completion of Table 4.2, modify your code to produce the frequencies listed on the table. To change between the frequencies use a pushbutton connected to the MCU.

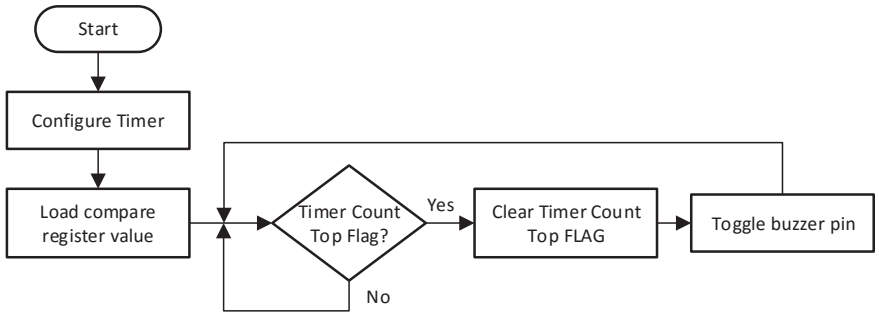


Figure 4.4: Timer by polling flowchart

Complete Table 4.2. Specify the clock period, number of timer bits, and the prescaler currently used.

Table 4.2: Timer MCU values

Clock Frequency:			
Timer's Bits:			
Preescaler used:			
Frequency	Period	Period/2	Compare register value nedded
500 Hz			
1 KHz			
1.5 KHz			
2 KHz			
3 KHz			

Note: The timer architecture presented is commonly used in MSP430 and ARM MCU architectures but other types of MCUs possess different timer architectures. These other architectures allows to pre-load a initial value in the timer's binary count that modify the quantity of clock cycles needed to reach it maximum value and restart in instead of using a value in the compare register to generate a reset in the timer.

4.2.2 Timer by Interrupt

Produce an audible sound using delays generated by the timer's interrupt.

Follow the steps outlined below:

1. Use the same setup as the basic exercise developed before "Timer By Poling".

2. Open your IDE.

3. First, calculate the terminal count value that produces a delay to generate an audible frequency (f=1000Hz). Take into account in your calculation that the signal to be produced must have a duty cycle of 50%.

4. Configure the timer mode necessary to generate the frequency calculate before and load the terminal count value calculated.

5. Write a timer ISR code to generate the audible frequency on the buzzer. You can use the flowchart in the Figure 4.5(a) as a guide to write your code.

6. Next, enable timer and global interrupts.

7. Compile your code and verify that it does not contain any errors.

8. Run your code and verify if the buzzer produces a sound.

9. Now, modify your code to produce the frequencies listed in Table 4.2. To change between the frequencies, use a pushbutton connected to the MCU.

4.2.3 7-Segment Display

Generate a counter from 0 to F displaying the number on a 7-segment display.

Follow the steps outlined below:

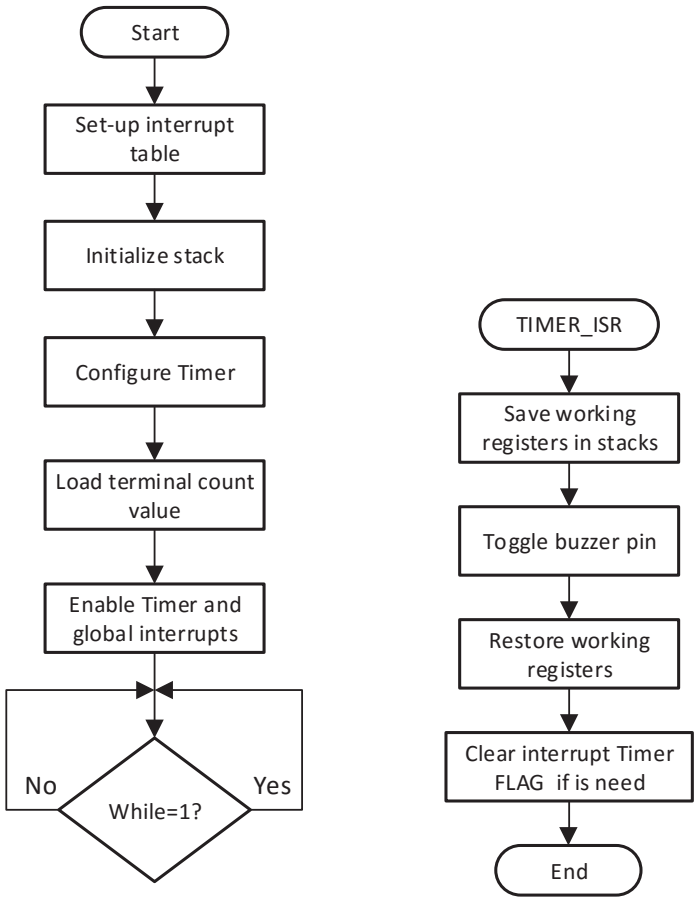
1. Connect a 7-segment display according to the schematic shown in Figure 4.6.

2. Calculate the value of the series resistor to limit the current through the segment to not overload your MCU I/O pin and satisfy the 7-segment requirements. Also, calculate the transistor base resistor to ensure it would saturate with the maximum 7-segment current.

3. Make a table to decode the digits from 0 to F into 7-segment codes. Remember, that if you are using a common anode 7-segment, a logic low value is needed on the data line to power-on the segment. Complete the table 4.3 using as reference the 7-segment labels shown in Figure 4.7.

4. Open your IDE.

5. Make a look-up table with the data completed in Table 4.3.



(a) Main Program (b) Service Routine (ISR)

Figure 4.5: Timer by polling flowcharts

6. Write a program that sends the appropriate code to the 7-segment port to display the digits between 0 and 9 every 1 second. Use a timer ISR to produce a delay time of 1 second and increment the value that will count the numbers. Use the pseudocode listed below as a guide to your main code:

Listing 4.1: 7-segment display Pseudocode

```
1 ;-------------------------------------------------------
2 ;   Program Start
3 ;   INIT RESET VECTOR
4 ;   INIT STACK POINT, WDT
5 ;-------------------------------------------------------
6 lookup = C0h, F9h,....
```

```
7  Number = 0
8
9  While   TRUE
10     Port = @lookup + number
11     Pin_control = 0
12 Endwhile
13 ;-------------------------------------------------------
```

7. Compile your code and verify that it does not contain any errors.

8. Run your code and verify if the numbers in the 7-segments are appearing.

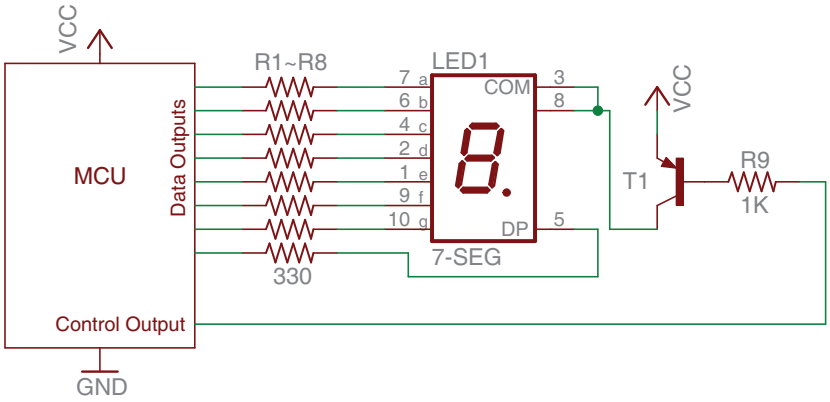


Figure 4.6: Schematic for 7-segment

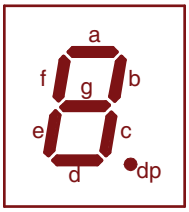


Figure 4.7: Segments names

4.2.4 Multiplexed Display Using a dual 7-segment display

The objective of this section is to generate a counter from 00 to FF using dynamic display techniques. Read the note section at the end of this exercise to have a better understanding on how the dynamic display works and its implications.

Table 4.3: Codes for 7-segment display of digits from 0 to F

#	dp	g	f	e	d	c	b	a	7-seg
0	1	1	0	0	0	0	0	0	C0h
1	1	1	1	1	1	0	0	1	F9h
2									
3									
4									
5									
6									
7									
8									
9									
A									
B									
C									
D									
E									
F									

Follow the steps outlined below:

1. Connect two 7-segment displays according to the schematic shown in Figure 4.8 taking into account the electrical consideration explained before for a 7-segment connection.

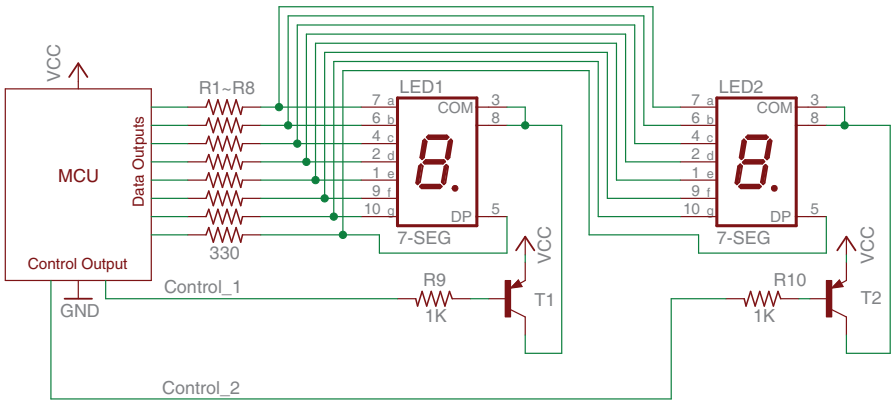


Figure 4.8: Schematic two 7-segments displays

2. Open your IDE.

3. Make a code to implement a dynamic visualization technique to display the two numbers into the 7-segment displays. Use the following steps and pseudocode to produce the your main code:

   (a) Turn off the control signal for both 7-segments

   (b) Send the data to appear in the first 7-segment

   (c) Turn on the first 7-segment control signal

   (d) Delay loop

   (e) Turn off the control signal for both 7-segment

   (f) Send the data to appear in the second 7-segment

   (g) Turn on the second 7-segment control signal

   (h) Delay loop

   (i) Back to step a

Listing 4.2: Dynamic display Pseudocode

```
;-------------------------------------------------------
;   Program Start
;   INIT RESET VECTOR
;   INIT STACK POINT, WDT
;-------------------------------------------------------
lookup = 2Fh, 06h,....
Num_7Seg1=0
Num_7Seg2=0

While   TRUE
    Pin_control_1 = Pin_control_2  = 0  ;turn off both displays
    Port = @lookup + Num_7Seg1          ;obtain 7-seg code for
                                        ;upper digit
    Pin_control_1 = 1                   ;turn on display
    call delay_loop                     ;delay loop

    Pin_control_1 = 0                   ;turn off display
    Port = @lookup + Num_7Seg2          ;obtain 7-seg code for
                                        ;lower digit
    Pin_control_2 = 1                   ;turn of display
    call delay_loop                     ;delay loop
Endwhile
;-------------------------------------------------------
```

4. Implement the delay loop using a timer ISR to refresh the two 7-segments with a refresh rate of 60Hz. The number to be displayed in the 7-segments must be increased every 1 second.

5. Compile your code and verify that it does not contain any errors.
6. Run your code and verify if the numbers in the 7-segments are appearing.

Note: The dynamic display technique is a software technique that allows controlling one or more 7-segment displays at time using the same segment lines for all the display. Through a constantly blinking process in each display, the technique generate a visual effect that allows to see all the displays in On at the same time. This technique is commonly used in display applications because reduce the amount if MCU pins to control serval display quantities. A drawback of this method is the loss of brightness in the display due the display is not in On all the time.

4.3 Complementary Tasks

4.3.1 Digital Tachometer

The activity consists of an implementation of a tachometer for the encoder wheel built in Experiment 3. Use the LCD for displaying in the first line a message with the speed: "Speed=####RPM" (Four units to represent the speed value must be used). In the second line indicate whether the rotation direction is clockwise or counterclockwise. Internally, use the timer and interrupts to determine the speed of rotation of the wheel and its direction.

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made personally in the lab and including the hardware and software components needed for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with relevant component calculations (current, voltage, timing values, etc.)
- Code listing with comments.

- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Experiment 5

Low-Power Modes and PWM

Objectives

- Understanding how low-power modes help to reduce the energy consumption of embedded system
- Using low-power modes to improve the power performance of an embedded application
- Identifying and understanding PWM architectures and operating modes
- Using a PWM module to control electronic devices such as LEDs

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 5.1: Bill of materials for completing Lab. 5

Item #	Qty	Description	Reference
1	1	Development board	
2	1	IDE application	
3	1	LCD display: 2 lines, 16 characters	W/HD 44780 Controller
4	1	Light Emitting Diode	5mm Red LED
5	1	Light Emitting Diode	5mm RGB LED
6	1	1/4W Carbon fill resistor	220 Ω
7	3	1/4W Carbon fill resistor	510 Ω
8	4	1/4W Carbon fill resistor	4.7 KΩ
9	1	Momentary switch	Pusbutton
10	1	3 columns by 4 rows Buttons array	Keypad 3x4
11	1	Multimeter	Fluke 179

5.1 Introduction

5.1.1 Low-Power Modes

Power consumption in embedded systems is an important design factor that affects a wide range of aspects, from battery life in portable applications to issues such as reliability, cost, size, and environmental impact. Therefore, using MCUs with low-power modes and learning how and when to activate and use those modes becomes of utmost importance in the design of embedded systems.

The activation and use of low-power modes in a microcontroller unit involves minimization of the individual current consumption of its internal peripherals, minimization of the CPU activity, and optimization of the code running during active periods. Depending on the particular MCU, the activation of a low-power mode can involve: disabling the CPU by turning it off or sending it to a standby mode, reducing the CPU clock frequency and, changing the clock source, among other strategies.

An effective way of incorporating low-power modes into an application is by configuring the code running on the CPU to operate using interrupts. Figure 5.1 shows a flowchart of a program that uses low-power modes. Basically, the program starts with the initialization of peripherals and system components, continues enabling interrupts of expected events and finally sends the CPU into a sleep mode. Every time an enabled interrupt is triggered, it will wake-up the CPU to serve the event and go back to sleep. This procedure saves more energy because instead of continuously polling for the expected event to occur, the CPU is only active when the interrupts mark the event that needs to be served.

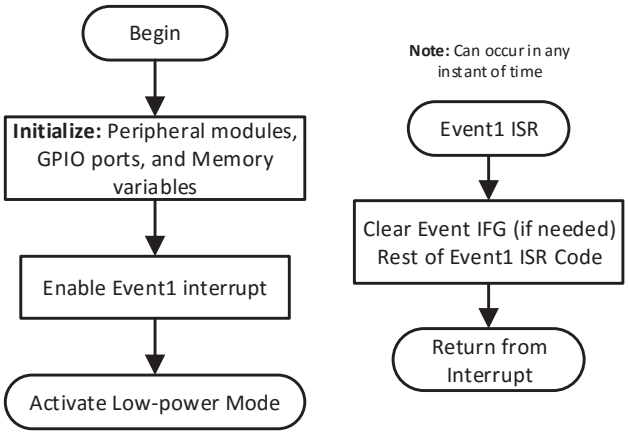


Figure 5.1: Main Program using low-power mode and a single event ISR

Low-power modes are of utmost importance in battery-powered applications due to the dramatic reduction in power consumption they induce. As illustrated in Figure 5.2, depending on the low-power mode activated, the system could achieve different levels of power consumption.

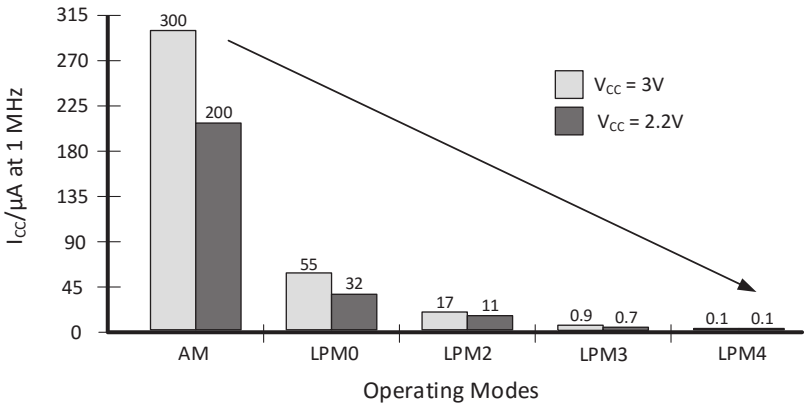


Figure 5.2: Current Consumption of MSP430F21x1 devices in different operating modes (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

5.1.2 Pulse Width Modulation

Pulse Width Modulation (PWM) is another useful timer application in embedded systems. A PWM module produces a square wave signal with a predefined and controlled duty cycle, allowing the generation of a signal with different pulse widths in different time periods as we can see in Figure 5.3.

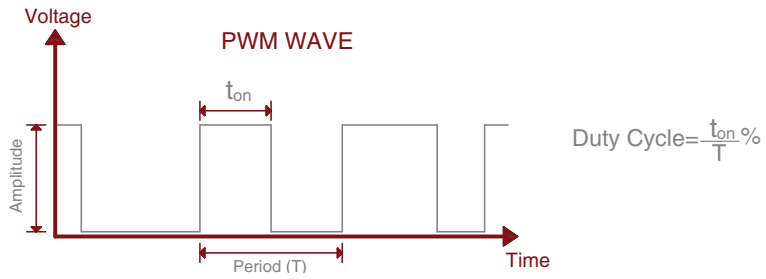


Figure 5.3: PWM signal parameters

A basic PWM module structure is shown in Figure 5.4. This structure is similar to the basic timer structure presented in the previous lab, with the difference that a

PWM module uses a High count register together with an n-bit hardware comparator instead of the compare register to generate the square wave signal. It fundamentally contains an n-bit timer (with clock selector and Prescaler), whose count is compared in hardware to the contents of a "high-count register" (*hc*). Figure 5.5 illustrate the behavior of a PWM module in which, while the timer has a value less than *hc* the PWM output is high, otherwise, the output is low. This mode of operation depends basically on the PWM module architecture. Some MCUs incorporate an enhanced PWM architecture to allow the user to use different modes of operation and interrupt sources.

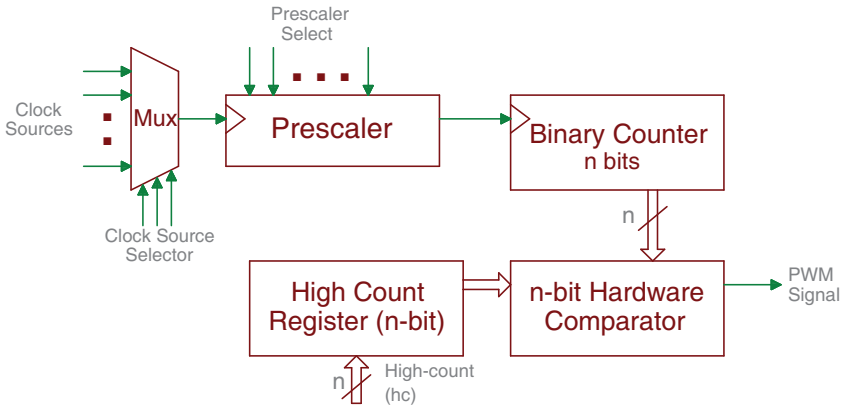


Figure 5.4: PWM basic architecture (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

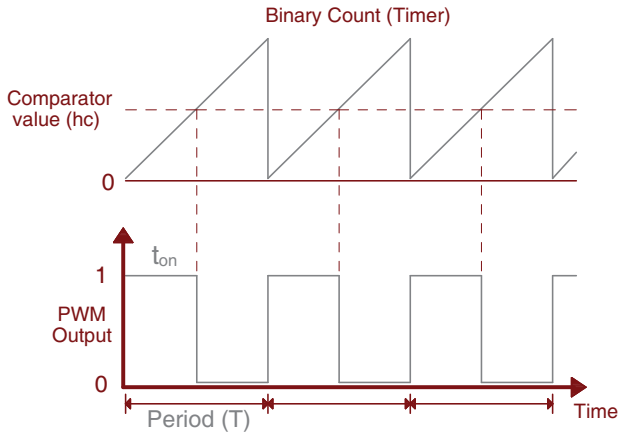


Figure 5.5: PWM signal parameters

Applications like DC motor speed, heater's temperature, light intensity in LEDs,

and even musical tones can be implemented with PWM. These many applications make PWM modules a useful addition to the list of MCU peripherals. See class's book Section 7.4.3 (Signature Timer Applications) for a detailed explanation about the PWM architecture and applications.

5.2 Basic Exercises

5.2.1 Low-Power Modes

Compare the low-power current consumption of your MCU versus that in normal mode.

Follow the steps outlined below:

1. Assemble the setup used for the scrolling list developed in Experiment 2 to scroll messages in an LCD display using two keys.

2. Load the software version of your code that reads the switches by polling. Make sure that your development board is only connected to the host computer through the programming adapter, with no other external power supply connected to it through the downloading process.

3. Remove the programming adapter and all connections to the host computer. Your program will stay safely in the MCU flash memory.

4. Connect your development board with an external power supply allowing the connection of an amp-meter to measure the MCU's current consumption. Connect the LCD to the power supply such that its current does not pass through the MCU amp-meter. Make sure the multi-meter is in current mode and the leads are connected to the amp-meter inputs. Use the block diagram shown in Figure 5.6 as a guide to make the necessary connections.

5. Power-Up your development board and LCD, then measure the MCU current several time and take the average value.

$$Average\ I_{CC}1: \_\_\_\_\_$$

6. Scroll the list up/down a few times to see the average load current measured by the amp-meter.

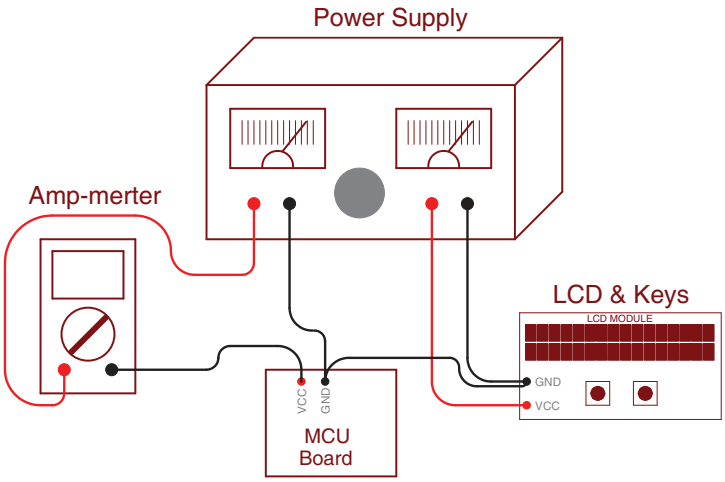


Figure 5.6: Connection diagram for MCU with power supply and amp-meter

7. Turn the power supply off, remove the power supply cables from the board and re-attach the programming adapter.

8. Edit your code to serve the keys by interrupts. Also, modify the main program to make the MCU enter a low-power mode, right after the system set-up is completed. Remember to enable global and peripheral interrupts.

9. Download the code to your system and with the programming adapter still connected (do not connect your MCU to the power supply yet!), debug your code and make sure it works as expected.

10. Once you have a working version of your code, remove the programming adapter and re-connect the external power supply to the boards, making sure to connect the multimeter as illustrate in Figure 5.6.

11. Measure the MCU's current consumption using the amp-meter (again, make multiple measurements and take the AVG).

$$Average\ I_{CC}2: \_\_\_\_\_$$

12. Compare the two current values measured, did you notice any change between the two measurement? For the two current values measured calculate the expected battery life for a 1500 mAh battery.

5.2.2 PWM Signal Generation

Produce a square wave signal using a PWM module.

Follow the steps outlined below:

1. Connect the oscilloscope to your MCU according to the schematic shown in Figure 5.7.

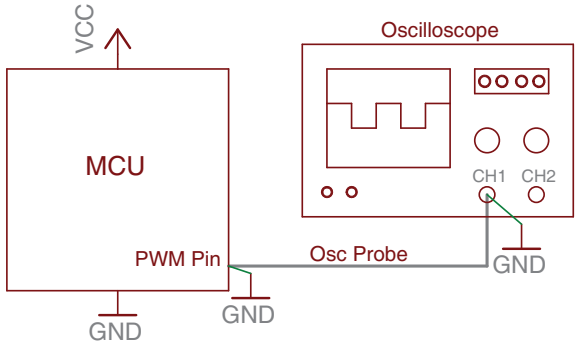


Figure 5.7: Block diagram for oscilloscope connection

2. Verify that the I/O port where the oscilloscope is connected to has PWM capabilities, if not, move the oscilloscope probe to an appropriate pin.

3. Open your IDE.

4. Identify the PWM module to be configured according to the pin previously selected and configure it to use one of the MCU's clock sources. Look for the register associated with the Period and Pulse width in your PWM module.

5. Calculate the terminal count value that produces a signal with frequency of 1000Hz, this value must be loaded into the period register. Also, calculate the value to be loaded in the pulse width register to produce a duty cycle of 50%.

6. Determine the appropriate operating mode of your PWM module, and configure the PWM registers necessary.

7. Next, enable your PWM module.

8. Compile your code and verify that it does not contain any errors.

9. Run your code and verify that a signal is visible on the oscilloscope.

10. Extract the waveform signal from the oscilloscope. Obtain the period and duty cycle of the signal on the oscilloscope.

11. After the completion of Table 5.2, modify your code to produce each frequency listed in the table. Use the oscilloscope to save each waveform observed. Do not forget measuring the duty cycle and period of each signal.

Complete the following Table 5.2. Specify the clock period and the count values for the PWM registers. Later, complete Table 5.3 with the values measured for each signal and calculate the % of error in the duty cycle.

Table 5.2: Timer MCU values

Frequency	Period (T)	Period register value	50% Duty cycle (DC) register value
500 Hz			
1 KHz			
2 KHz			
4 KHz			
8 KHz			

Table 5.3: Timer MCU values

Frequency	Measured T	Measured DC	% Error DC
500 Hz			
1 KHz			
2 KHz			
4 KHz			
8 KHz			

5.2.3 Generating colors with an RGB LED

The purpose of this section is generating different colors using an RGB LED with PWM signals.

Follow the steps outlined below:

1. Connect the RGB LED according to the schematic shown in Figure 5.8. Be sure that your MCU has three pins with PWM capabilities.

2. Calculate the series resistor value to obtain the maximum brightness possible of each LED color without overloading the pin's current.

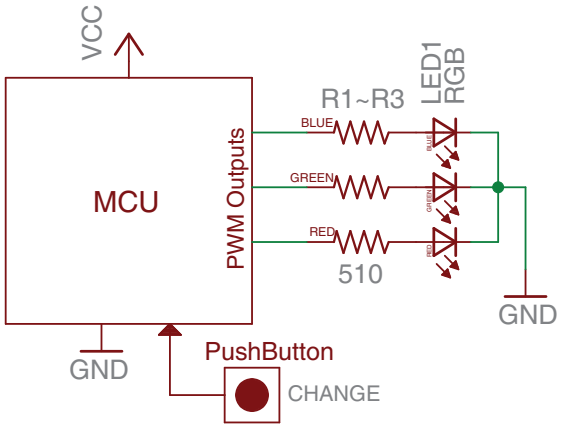


Figure 5.8: Schematic RGB LED Connection

3. Open your IDE.

4. Configure the PWM modules to produce the signals with a frequency of 1000Hz.

5. Make a look-up table with the duty cycle values of RED, BLUE, and GREEN corresponding to each color shown in Table 5.4. Take into account that a value of 255 in the color represents a 100% duty cycle for your output signal and a value of 0 represents 0% duty cycle.

Table 5.4: RGB Color Values

	R	G	B
1	0	0	255
2	0	255	0
3	255	0	0
4	255	30	217
5	30	222	252
6	240	200	40
7	255	123	33
8	255	255	255

6. Write an ISR code such that every time a pushbutton is depressed, changes the values in the PWM signals to produce a different color.

7. Enable the interrupt flags and PWM modules.

8. Now, write a main code to make the MCU enter a low-power mode right after the system set-up is completed and the interrupts have been enabled.

9. Compile your code and verify that it does not contain any errors.

10. Run your code and verify if the LED color changes when you depress the pushbutton.

5.3 Complementary Tasks

5.3.1 Digital Dimer

The activity consists of making a digital dimer for an LED. The system must allow changing the LED brightness from 0% (LED Turn-Off) to 100% (Maximum brightness), in increments of 10%, for a total of 11 levels of luminosity. The level is selected by the user through a keypad where each key corresponds to one level (e.g. $0 \rightarrow 0\%$, $1 \rightarrow 10\%$, $2 \rightarrow 20\%$, $\cdots$, $9 \rightarrow 90\%$, and $\# \rightarrow 100\%$) once the key is depressed. The LCD must display the current level selected by the user and its corresponding percentage of brightness. The LED must by driven by a PWM signal generated from the MCU. Use the block diagram shown in Figure 5.9 as a reference to connect your system.

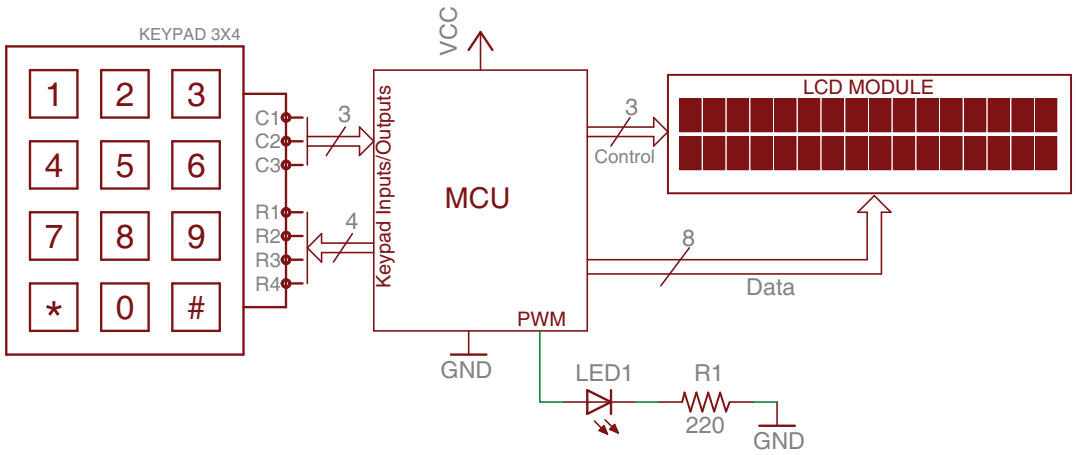


Figure 5.9: Digital dimmer connection diagram

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made personally in the lab and including the hardware and software components needed for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with relevant component calculations (current, voltage, timing values, etc.)
- Code listing with comments.
- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Experiment 6

Motor Interfacing

Objectives

- Understanding the operating principles of electric motors used in embedded applications
- Recognizing electromechanical characteristics of DC motors, stepper motors, and servomotors
- Employing H-bridges and current drivers to control DC and Stepper motors
- Employing PWM signals to control servomotor position angle

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 6.1: Bill of materials for completing Lab. 6

Item #	Qty	Description	Reference
1	1	Development board	Tool
2	1	IDE application	Tool
3	1	LCD display: 2 lines, 16 characters	W/HD 44780 Controller
4	2	1/4W Carbon fill resistor	22 Ω
5	4	1/4W Carbon fill resistor	330 Ω
6	2	1/4W Carbon fill resistor	4.7 KΩ
7	2	1/4W Carbon fill resistor	12 KΩ
8	4	P-N junction diode	1N4004
9	2	BJT NPN transistor	MPSA42
10	2	BJT PNP transistor	MPSA92

Table 6.1: Continued

11	2	Momentary Switch	Pushbutton
12	2	Optocoupler	4N25
13	1	Half H-bridge Driver	L293D
14	1	Darlington Transistor Array	ULN2803
15	2	Optoswitch	RPR-220
16	1	DC Motor, 6VDC, 9100rpm, 0.14Oz-in	711 Motor
17	1	Stepper, 5VDC, Unipolar, 11.25° step angle	28BYJ-48
18	1	Servo, 6VDC, 38Oz-in, 180° Range	900-00005

6.1 Introduction

An electric motor can be defined as an electromechanical device capable of transforming electrical power into mechanical power. This kind of devices is extensively used in embedded systems applications where precise mechanical movement is required. Some examples include plotters, inkjet printers, and CNC (computer numerical control) machines. Depending on the type of application, different types of electric motors can be found. The three most common include: DC motor, servomotors, and stepper motors. Although these three types of motors all transform electricity into mechanical power, there are fundamental differences among them in terms of how they can be controlled.

6.1.1 Direct Current Motors

A direct current (DC) motor continuously spins when energy is applied to their electrical terminals. The speed of a DC motor is generally a function of the applied voltage. Thus, they can be used in applications where only the spinning speed is important.

Due to the electrical limitations of MCU's pins in terms of voltage and current, a direct connection between an MCU and a motor can rarely be done. Motor drivers are required to manage the motor load, speed, and the direction of rotation. These motor drivers can be implemented using discrete components or acquired in the form of integrated circuits (IC). See class's book Section 8.10.1 (Working with DC Motors) for a detailed explanation about the characteristic and interfacing of DC motors.

A common motor driver is provided by an H-Bridge. A H-Bridge is composed of NPN and PNP transistors as illustrated in Figure 6.1. The rotation direction is

determined by the signal combination used to turn-on and turn-off the transistors. When signal Rev is low and Fwd is high, transistors Q1 and Q3 are simultaneously turned On, allowing the current to flow from left to right in the motor. Making Rev high and Fwd low will activate transistor Q2 and Q4 instead, reversing the current direction and, therefore, reversing the polarity of the voltage applied to the motor. This causes a change in the direction of rotation. The transistors used to construct an H-Bridge are selected to satisfy the motor specifications (current and voltage).

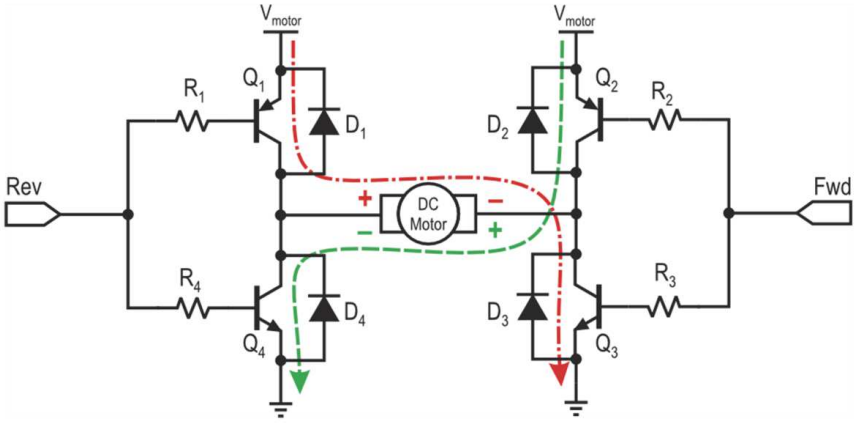


Figure 6.1: Transistor H-Bridge (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

A L293D is a motor driver with four channels capable of supplying a current up to 600mA per channel. This driver incorporates diode protection in each channel to avoid damage by the inductive turn-off transient generated by the motor. Each channel is controlled by TLL signals and each pair of channels has an enabling signal to connect and disconnect the channels.

The DRV8833 is another IC driver composed by dual CMOS H-bridges. This chip contains two full H-bridges capable of supplying a current up to 3A at a voltage up to 10.8V. This driver also provides short circuit protection, thermal shutdown, and supports low power modes.

6.1.2 Servo-Motors

A servo-motor is a DC motor with a feedback control that allows for a precise position control. A servo-motor is composed of four main components as illustrated in Figure 6.2: a DC motor that provides the basic electromechanical conversion, a control board housing the feedback electronics, a set of gears that slow-down the

DC motor rotation speed and increases the torque, and a position sensor, typically a potentiometer.

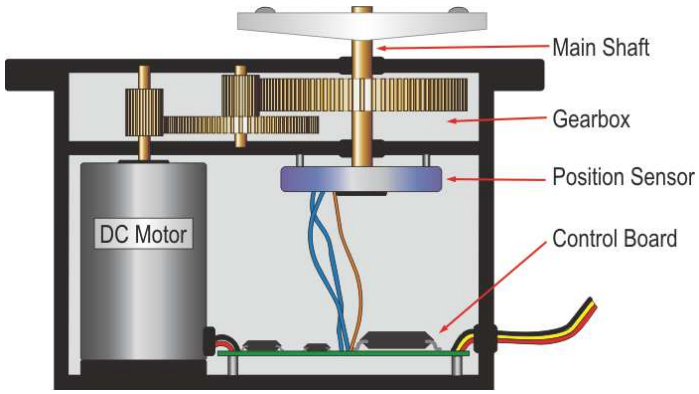


Figure 6.2: Servo-motor internal composition (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

The operating voltage of a servo-motor is generally in the range of 4 to 8 volts. A servo is controlled by a pulse-width modulation (PWM) signal that determines the position of the servo. Basically, the duration of the signal in high (duty cycle) determines the angular position of the motor, as illustrated in Figure 6.3. The position sensor in the servo continuously indicates the shaft angular position to the control board. A servo motor has a restricted travel angle of about 200° or less (typically of 180°) due the gearbox attached to the DC motor.

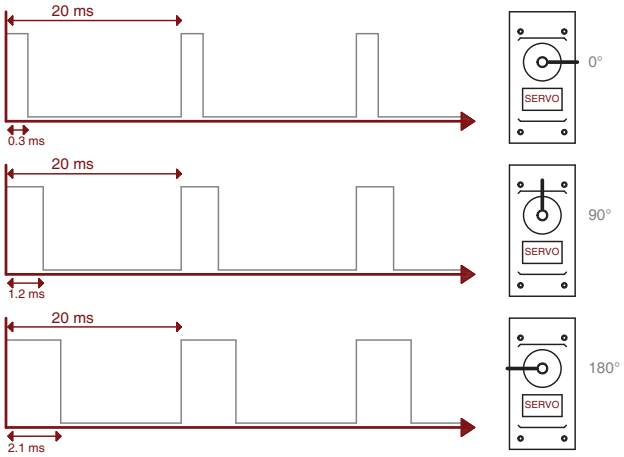


Figure 6.3: Servo-motor position determined by the signal pulse width (*Source*: Microcontrolador PIC16F84. Desarrollo de Proyectos, E. Palacios)

The external interface of a servo-motor has only three wires: V_{DD}, GND, and a *Control* signal. Power is applied trough the V_{DD} and GND terminals while the desired position is specified through the *Control* pin via a PWM signal. Each servo-motor has his own operation range that corresponds to the maximum and minimum pulse-width that the servo understands. See class's book Section 8.10.2 (Servo Motor Interfacing) for a detailed explanation on how to interface and control servo-motors.

6.1.3 Stepper Motor

A stepper motor is a type of electric motor that possesses a shaft which moves in discrete increments. The movement is the product of digital pulse sequences applied from a controller. Each pulse produces a precise angular displacement known as a step. Rotation increments or steps are measured in degrees. The structure of a stepper motor is different from of a DC motor, as it incorporates multiple windings to make possible the stepping behavior. Depending on how the rotor and stator are designed, stepper motors are classified in three types: **variable reluctance**, **permanent magnet**, and **hybrid**. See class's book Section 8.10.3 (Stepper Motor Interfaces) for a detailed explanation about the types of stepper motors.

A variable reluctance stepper motor has a soft Iron, non-magnetized, multitoothed rotor and a wounded stator with three to five windings (unipolar). The number of poles in the stator is larger than the number of teeth in the rotor. Torque is developed when the poles and teeth seek to minimize the length of the magnetic flux path between the stator poles and rotor teeth. In a permanent magnet stepper, the rotor is built using permanent magnets without teeth and the stator is constructed using multiple windings. In this case torque occurs when the excited stator poles attract opposite magnet poles in the rotor while repulsing similar poles. Hybrid steppers combine features from variable reluctance and permanent magnet motors. Hybrid motors have the ability of producing high torque at low and high speeds through the use of two multi-toothed, soft iron disks with a permanent magnet between them. See class's book Section 8.10 (MCU Motor Interfacing) for a detailed explanation.

Stepper motors come in a variety of step resolutions, ranging from 0.72° to 22.5° per step (500 and 16 steps per revolution).

The most important parameters specifying stepper motors include:

- **Working Torque**: The maximum momentum that the motor can reach while responding to an impulse excitation. If the torque of the load is larger than the working torque, the motor will not move.
- **Dynamic Torque**: The torque that the motor possess at a defined speed.

This torque may vary depending on the load attached to the motor and the driver used to control the motor.

- **Holding Torque**: Is the amount of torque needed to move the motor when the windings are energized but the motor's rotor is not moving.
- **Maximum pull-in/out**: Is defined as the maximum number of steps per second that the motor can perform.
- **Step resolution**: Is the angular displacement experienced by the motor with each excitation pulse, measured in degrees. This parameter can also be specified as the number of full steps per revolution. Table 6.2 shows common stepper motors angles. To calculate the number of steps for a stepper motor, the Equation 6.1 can be used:

$$SN = \frac{360}{\alpha}, \tag{6.1}$$

where SN is the number of steps per revolution and α is the step angle.

Table 6.2: Common resolution in commercial stepper motors

Degrees per excitation pulse	N° steps per revolution
0.72°	500
1.80°	200
3.75°	96
7.50°	48
15.00°	24

6.2 Basic Exercises

6.2.1 DC Motor Driven with Transistors

In this exercise, you will implement a DC motor driver using discrete components such as transistors. The control signals, to define the motor rotation direction, will come from a set of pushbuttons.

Follow the steps outlined below:

1. Connect the DC motor according to the schematic in Figure 6.4. This schematic is an isolated H-drive that incorporates two optocouplers in the control signals

to prevent propagating the noise generated by the motor into the MCU. Set VCC according to the DC motor specifications. Two complementary transistors were chosen with a collector current (I_C) of 0.5Amp. This I_C is required to withstand the current peaks generated by the motor when an instantaneous change of direction is required or when a load is applied to the motor. Verify each connection twice before powering the circuit.

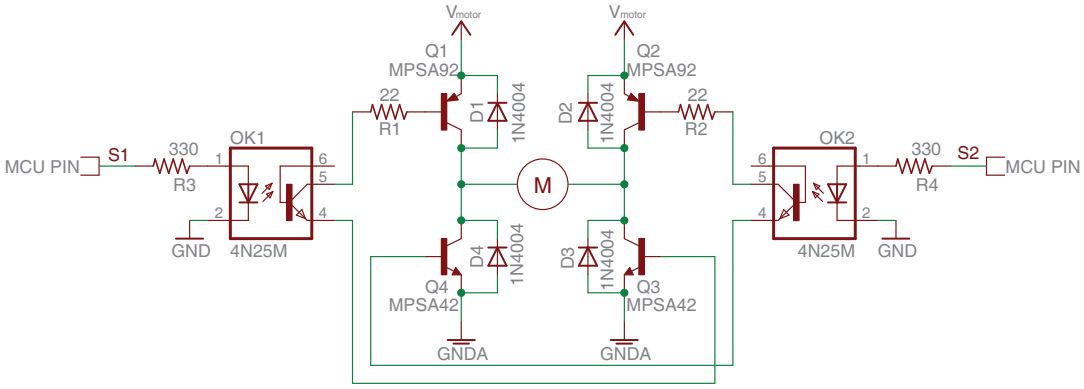


Figure 6.4: Isolated transistor H-Bridge schematic

2. Open your IDE.
3. Configure the I/O pins connected to the optocouplers as outputs.
4. Now, connect three pushbuttons and an LCD to the MCU. Each button must perform one of the functions described in Table 6.3. The LCD must be used to display the current motor state and the pushbutton depressed. S1 and S2 represent the logic values to be sent through the outputs connected to the optocouplers.

Table 6.3: Motor States

Motor			
Button	Action	S1	S2
1	Stop Free	0	0
2	Rot. Left	0	1
3	Rot. Right	1	0
X	Stop Forced	1	1

Do not attempt to send this command as it creates a short circuit in the H-drive and burns ALL transistors

5. Make a program to continuously perform the function selected by each button until another function is selected. The program must start with the motor in the "stop free" condition.

6. Compile your code and verify that it does not contain any errors.

7. Run your code and verify if the DC motor performs the function selected.

6.2.2 DC Motor Controlled Through Driver IC

The objective of this exercise is controlling the direction of rotation and velocity of a DC motor through a L293D IC driver. The L293D is an IC driver that contains four half H-bridges designed to provide bidirectional drive currents up to 600mA.

Follow the steps outlined below:

1. Connect the DC motor according to the schematic in Figure 6.5. Set the VCC voltage according to the DC motor specifications. The IC is designed to work with 3.3V or 5V logic.

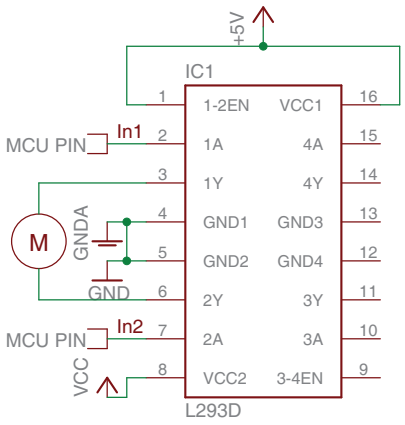


Figure 6.5: L293D H-Bridge schematic

2. Connect In1 to the pin where the S1 signal was generated in the previous exercise. Repeat the same for In2 and S2.

3. Open your IDE and perform the steps, 2 through 5, describe in the basic exercise 6.2.1.

4. Run your code and verify if the DC motor performs the function selected.

5. Now, modify your program to increment or decrement the velocity of the motor in steps of 20% using the first and second pushbutton. The third pushbutton must be used to toggle between the motor states. The change in velocity must be allowed in both directions of rotations. Hint: Use two PWM channels instead of the GPIOs selected.

6.2.3 Servo-motor Interfaces

The purpose of this section is controlling a servo motor using a PWM signal.

Follow the steps outlined below:

1. Connect a 900-00005 servo-motor according to the schematic in Figure 6.6.

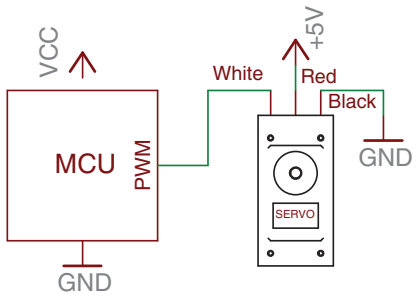


Figure 6.6: Servo-motor connection schematic

2. Verify that the I/O port where the servo-motor signal is connected has PWM capabilities. Otherwise move it to a pin able to produce PWM signal from an internal MCU timer.

3. Open your IDE.

4. Configure the PWM module to produce a square signal of 50Hz. Do not forget to configure the Timer associated to the PWM module.

5. Make a program to produce the angle displacements listed in Table 6.4. Use a timer function to change between the positions every 2 seconds. Complete the missing information of the table. Remember do not exceeds the pulse-widths values for the 0° (0.375ms) and 180° (2.1ms) in order to avoid damages and excessive current consumption in the servomotor.

6. The system must start with the servomotor in 0° position. Take into account that you have to permanently send the pulse width that corresponds to the

Table 6.4: Servo-motor angles routine

Angle Displacement	Servo Angle position	Necessary Pulse-width
22.5 to the left		
90.0 to the left		
22.5 to the right		
45.0 to the left		
90.0 to the right		
135.0 to the left		
22.5 to the right		
90.0 to the right		
67.5 to the right		

position desired to hold the servo-motor in that position. Design a setup with marks to verify if the servo reaches the desired angle.

7. Compile your code and verify that it does not contain any errors.

8. Run your code and verify if the servo performs the programmed routine.

6.2.4 Stepper Motor Interfaces

In this part, you will implement a full step and a half step sequence to control a unipolar, two-winding stepper motor.

Follow the steps outlined below:

1. Search information related to unipolar stepper motors and how they work. Also, search information about the 28YBJ-48 stepper motor and its characteristics, and IC driver ULN2803.

2. Connect the stepper motor according to the schematic in Figure 6.7.

3. Open your IDE.

4. Configure the four MCU I/O pins connected to the ULN2803 as outputs.

5. Write a program to perform the signal sequence described in Table 6.5. Take into account that the ULN2803 inverts the logic of its input signals (a high voltage in the input produces a low voltage at the output). The sequence is a one-phase activation that allows moving the motor shaft while saving energy in comparison with a two-phase activation. See class's book section

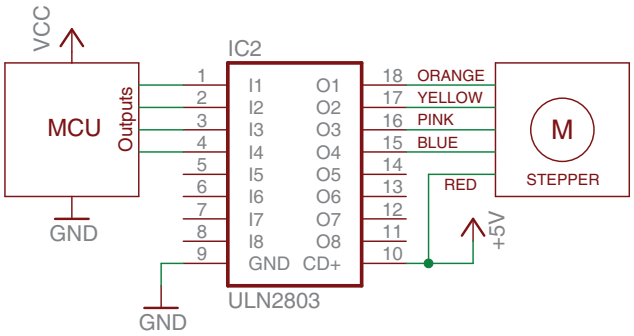


Figure 6.7: Stepper motor connection schematic

8.10.5 (Permanent Magnet Stepper Motors) for a detailed explanation about one-phase and two-phase activation.

Table 6.5: Full-Step Sequence

	Motor Coils			
Step	(4) Orange (A)	(3) Yellow (B)	(2) Pink (A')	(1) Blue (B')
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Read Note section for full-step sequence explanation

6. Use a delay of 10ms between steps.

7. Compile your code and verify that it does not contain any errors.

8. Run your code and verify that the stepper motor works as expected.

9. Now, modify your code to implement the signal sequence describe in Table 6.6 that corresponds to a Half-step sequence.

10. Use the same delay established previously (10ms).

11. Run your code and verify if the stepper motor works as expected.

12. Finally, modify your code to rotate the stepper motor 270 to the left, later on 180 to the left, and finally 90 to the right. Use marks to verify if the stepper reaches the desired angle.

Table 6.6: Half-Step Sequence

	Motor Coils			
Step	(4) Orange (A)	(3) Yellow (B)	(2) Pink (A')	(1) Blue (B')
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

Read Note section for half-step sequence explanation

Note: In stepper motors, a full-step sequence allows the motor to advance in angular increments equal to its nominal resolution. In the case of a half-step sequence, the motor resolution is double because the motor advances only half of its nominal angular resolution. If a motor has an angular resolution of 11.25° per step, with a half-step sequence the displacement will be 5.625° per step.

6.3 Complementary Tasks

6.3.1 Stepper Motor Characterization

The activity consists in determining the maximum input signal frequency in which the stepper motor can work without missing steps. The system shall use keys UP and DOWN to increase and reduce the motor speed, and START/STOP to turn the motor on and off.

An oscilloscope shall be used to observe and measure the motor input signal. The control signals must start at the 1st value in Table 6.7 as the initial speed of the motor. When the START/STOP key is depressed the motor must perform two complete rotations (720°) at the selected speed, based on the # of steps required for a revolution. A mark must be placed on the motor shaft to observe is the motor is able to perform the two revolutions, see Figure 6.9 for an example on how to do the angle measurements. Once the two rotations are completed, the following input signal period shall be selected using the UP key. The input signal periods for the motor tests are defined in Table 6.7. The missing information in Table 6.7 must be completed and calculated for each test. The motor speed must be derived from the input signal frequency. Plot a graph where the relationship between the motor

velocity and input period signal can be observed. Be careful while the motor is being tested; if the motor emits inappropriate sounds or the angular movement is irregular, the test must be stopped using the STARt/STOP key and the frequency should be marked unsuccessful in the Table (Critical Frequency column). You can use the block diagram shown in Figure 6.8 as a reference to connect your system.

Table 6.7: Frequencies to be tested

Sig. Period (ms)	Sig. Frequency (Hz)	Motor Speed (rpm)	Critical Frequency (Yes/No)
100			
50			
20			
10			
5			
2			
1			
0.5			
0.2			
0.1			

Note: Sig. means Signal

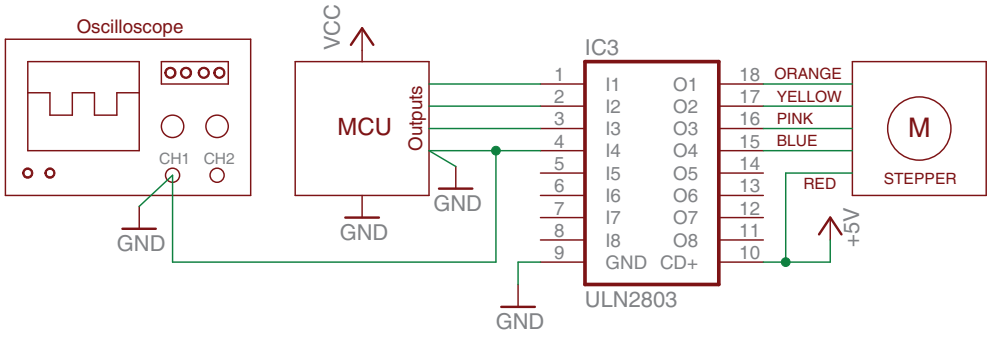


Figure 6.8: Stepper motor input frequency measurement diagram

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made

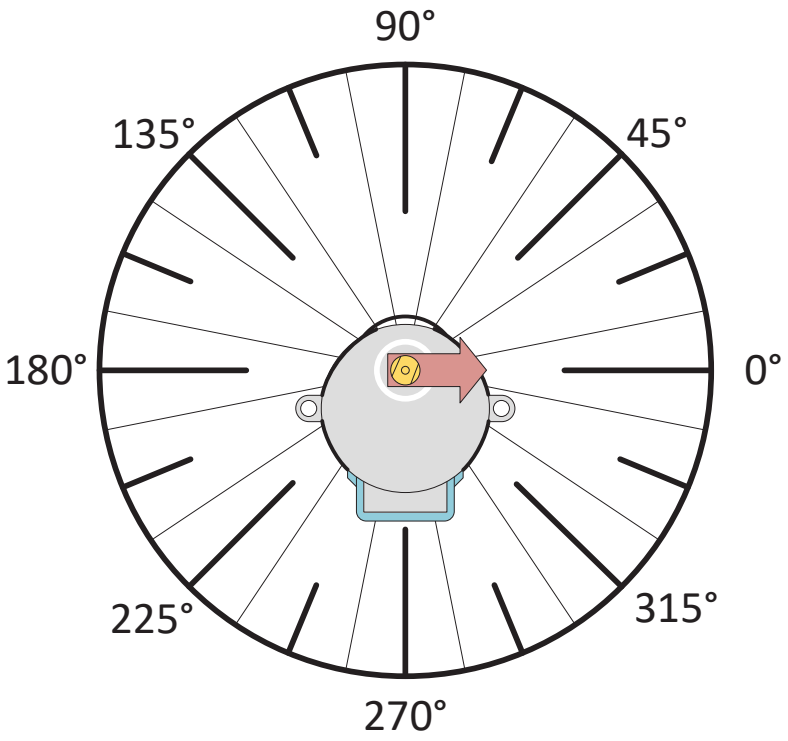


Figure 6.9: Stepper motor measurement circle example

personally in the lab and including the hardware and software components needed for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with component calculations
- Code listing with comments.
- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Experiment 7

Serial Communication

Objectives

- Identifying and understanding the standard formats for serial communications
- Recognizing the differences between synchronous and asynchronous serial communications
- Understanding the physical requirements and operation of an USART interface
- Using an USART interface to transmit and receive information to/from a personal computer and your MCU
- Understanding the connection and operation of an I^2C interface
- Employing an I^2C protocol to share data with a real-time clock device

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 7.1: Bill of materials for completing Lab. 7

Item #	Qty	Description	Reference
1	1	Development board	
2	1	IDE application	
3	1	LCD display: 2 lines, 16 characters	W/HD 44780 Controller
4	1	1/4W Carbon fill resistor	1 KΩ
5	5	1/4W Carbon fill resistor	4.7 KΩ
6	3	Momentary Switch	Pusbutton
7	1	I^2C Real-Time clock calendar	DS1307

Table 7.1: Continued

8	1	Quartz crystal	32.768KHz
9	1	3V Lithium Battery	CR2032
10	1	Piezoelectric buzzer	Buzzer
11	1	USB-To-UART converter cable	FTDI TTL-232R

7.1 Introduction

Serial channels are extensively used to establish communications between devices and computers, working under different formats and protocols in a wide range of applications, and transmitting information between points that can be from a few centimeters to hundred or thousand of kilometers apart.

In a serial communication channel, data is sequentially transmitted, one bit at a time, over a single data line. Each bit transmitted over a serial channel takes a predetermined amount of time ($tbit$) to be transmitted. Thus, transmitting an $n - bit$ character will take $n \cdot tbit$.

The number of bits transmitted in a serial channel per unit of time determines the transmission rate of the channel. Two commonly used metrics include:

- Bit rate: Number of bits-per-seconds (bps) transmitted over the channel.
- Baud rate: Number of symbols per seconds transmitted over the channel.

When the channel modulation scheme assigns one bit to each signal transmitted over the channel, the terms bit and baud rate are interchangeable. Modern modulation schemes commonly assign multiple bits per signal. In such cases, the terms have different meanings.

The most basic structure of serial channel calls for a transmit signal (TxD), a receive signal (RxD), a ground reference, and some form of clock synchronization, as illustrated in Figure 7.1. Depending on the protocol, additional signals might also be necessary for handshaking, synchronization, or data flow regulation. Common serial protocols and physical standards include RS-232, RS-485, SPI, I^2C, CAN-BUS, and 1-wired.

7.1.1 Types of Serial Channels

The serial channels could be Simplex, Half Duplex, or Full Duplex where the difference between them is the connectivity used to sent the information:

Simplex serial channel: The transmission process is only in one direction.

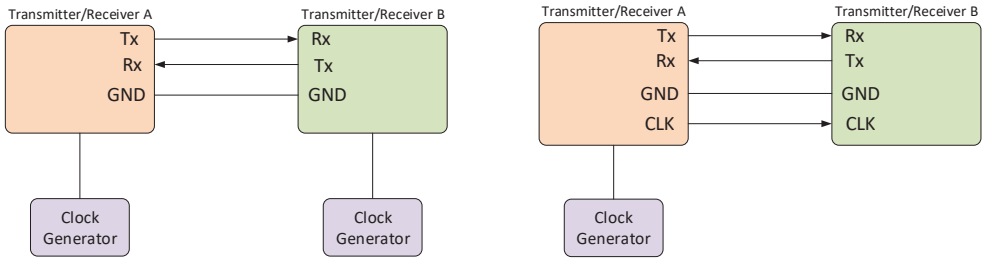
Half duplex serial channel: The transmission process could be in either direction, but only in one direction at a time.

Full Duplex serial channel: The transmission process could be in either direction simultaneously due to the usage of separate links (one for transmitting and one for receiving).

See class's book Section 9.2 (Types of Serial Channels) for a detailed explanation on this topic.

7.1.2 Synchronous Vs. Asynchronous Serial Communication

Generally, a serial channel requires the usage of a clock signal to synchronize the data transmission and reception ends. Depending on how the transmission clock is handled, the communication can be asynchronous or synchronous. In asynchronous communications, individual clock generators are used on each end of the channel, as shown in Figure 7.1(a). In synchronous communications, the clock signal is transmitted through the channel, as illustrate in Figure 7.1(b).



(a) Asynchronous (b) Synchronous

Figure 7.1: Synchronous vs asynchronous serial communication.

In both cases, asynchronous and synchronous channels, the transmitted message is divided into fundamental units known as packets or datagrams. Commonly, a packet contains three parts:

- A header that indicates the beginning of the packet.

- A body that contains the information or message being transmitted.
- A footer that delimitates the data. In most cases, the footer also includes redundant information that can be used for error checking & in some cases error correction.

7.1.3 Serial Interfaces

A Universal Synchronous/Asynchronous Receiver/Transmitter (USART) controller is a fundamental module that can generate the necessary signals for synchronous or asynchronous communication, one mode at a time. In asynchronous mode, an USART becomes an UART module. In synchronous mode, it can support one of several synchronous protocols, such as SPI, I^2C, and others.

USARTs contain multiple functional units and registers, which may vary from one architecture to another. However, there are a few basic components, as illustrated in Figure 7.2, that are fundamental to its operation. These includes:

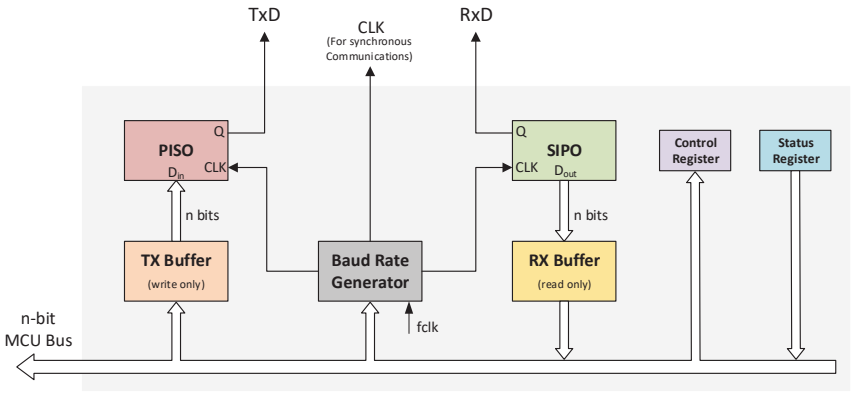


Figure 7.2: Minimum USART components

- A Baud Rate Generator: A timer that generates the clock frequency necessary for setting the transmission and reception speed (baud rate). The clock signal may be transmitted through the channel or generated at each end, depending on whether the channel is synchronous or asynchronous.
- Parallel Input Serial Output Shift Register (PISO): Converts n-bit parallel data from the CPU into a serial stream.
- Transmit Buffer (TX Buffer): Holds the data to be transmitted. Also called "Data-out register".

- Serial Input Parallel Output Shift Register (SIPO): Converts the serial input stream into parallel data.
- Receive Buffer (RX Buffer): Accommodates newly received characters for the CPU to read. Also called "Data-in register".

Two additional registers are necessary to operate a USART. These include a **control register** and a **status register**. A control register allows configuring the USART in the desired operating mode. For example, it allows choosing either synchronous or asynchronous mode, enabling the transmitter or receiver, enabling USART interrupts, setting number of bits to be transmitted, selecting error check, etc. The status register contains information that indicates the current USART status. Indicators, such as when the TX Buffer can be written, when the RX Buffer can be read, or when an error has occurred.

Status bits TxR (Transmitter ready) and RxR (Recover ready) are essential for a UART operation. TxR signals are used when the Data-out register is empty in order to accept new characters for transmission. RxR indicates that a new character has been received and is ready for reading in the Data-in buffer. These flags can be used in a polled fashion to operate the channel or be enabled to trigger interrupts, alluding to a more efficient way to operate the UART.

The baud rate generator, as any timer, is configured using the system clock frequency, a divider (n), and a prescaler value. The desired baud rate is obtained as:

$$BaudRate = \frac{fclk}{PrescalerValue * (n + 1)} \tag{7.1}$$

UART Operation

An asynchronous frame is composed of a start bit, multiple data bits (five- to eight-bit characters), an optional parity bit, and a stop bit as illustrated in Figure 7.3. The parity check is a simple mechanism for detecting errors in the channel. See class's book Section 9.3.5 (UART Structure and Functionality) and Section 9.3.7 (UART configuration and Operation) for a detailed explanation of UART hardware interface and operation.

I^2C

The Inter-Integrated Circuit bus (I^2C) is a synchronous serial protocol developed by Philips in the early 1980s to support board-level interconnections.

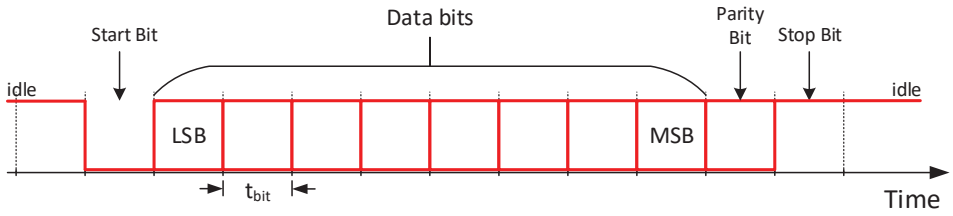


Figure 7.3: Typical asynchronous serial transmission frame (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

I^{2}C uses two signal lines to connect with other devices: Serial DAta line (SDA) and Serial CLock line (SCL), both ground (GND) referenced. The SCL line synchronizes all bus transfers while SDA carries the transferred data. Both SDA and SCL are open collector lines, requiring external Pull-up resistors. Figure 7.4 shows a basic I^{2}C topology with two masters and three slaves interconnected.

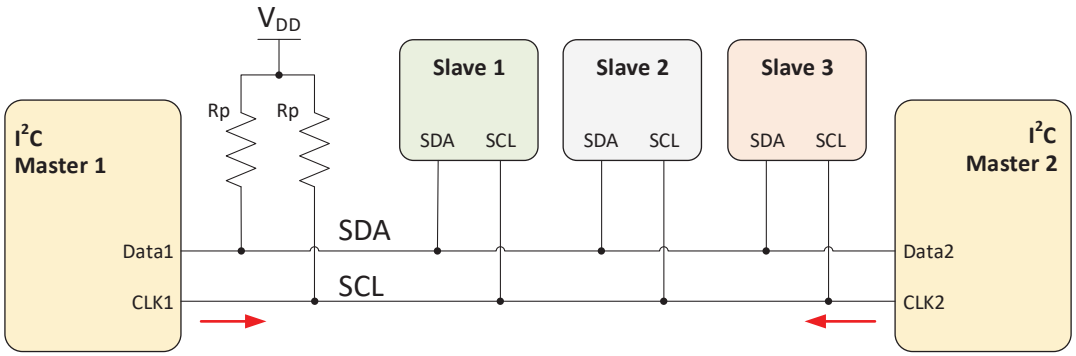


Figure 7.4: I^{2}C multimaster-multislave structure (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

In an I^{2}C protocol, the devices are software addressable, through a 7- or 10-bit address field. These number denote the maximum quantity of devices that can be accommodate on the bus but also, the number of devices is limited by the total bus capacitance. This capacitance also limits the maximum speeds that can be reached by the bus. Some predefined speeds for the protocol: are the standard speed (100Kbps) and the fast speed (400Kbps).

In I^{2}C, the master device controls the communication process. It defines the slave to communicate with, whether the data will be transmitted or received, and generating the necessary clock signals for the data transmission.

A typical structure of an I^{2}C packet is shown in Figure 7.5. The Figure denotes the

start condition sent by the master, the slave address field followed by the bit that indicates the communication direction (read or write). Then, the ACK sent by the slave, the data packets sent by the master, the ACK response from the slave in each packet received, and the stop condition.

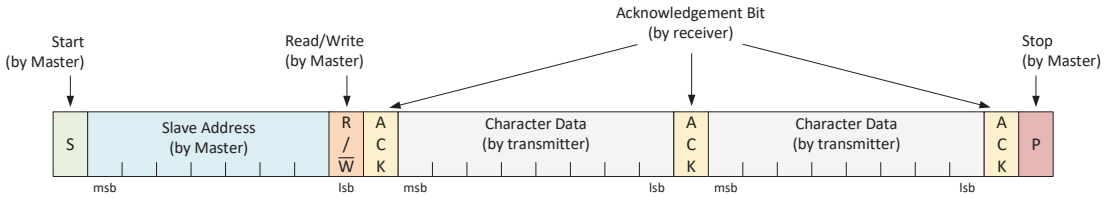


Figure 7.5: I^{2}C message structure (*Source*: Introduction to Embedded Systems, M. Jiménez, R. Palomera, I. Couvertier)

See class's book Section 9.4.2 (The Inter-Integrated Circuit Bus: I^{2}C) for a detailed and deep explanation about an I^{2}C architecture and interface.

7.2 Basic Exercises

7.2.1 Asynchronous Serial Communication (UART)

In this exercise, we will use a USB-to-UART cable for sending characters to a personal computer (PC) using asynchronous serial communication.

Follow the steps outlined below:

1. Locate the TX and RX signal pins in your MCU and connect them to the computer using a USB-to-UART cable. Be sure that the voltage logic of the cable corresponds to the operating voltage of your MCU. Do not forget to connect the GND to the USB-to-UART cable.

2. Open a HyperTerminal on your computer or other RS-232 communication program (Putty) and configure it for:

   - Baud rate: 9600 bauds
   - Data bits: 8
   - Parity bit: none
   - Flow control: none

3. Open your IDE.

4. First, configure the baud rate on your MCU (The serial configuration in the MCU must match those of the other device we wish to communicate to):
   - Configure the baud control register; that is, select the clock source and specify the prescaler and divider values.
5. Next, configure the UART for asynchronous transmission:
   - In the control register specify the mode to be asynchronous.
   - Enable the transmitter and global USART module.
   - Configure the pins selected on your MCU to work with the UART module.
6. Write a program to write a character into the UART transmitter via polling. You can use the flowchart in Figure 7.6 as a guide for your code.

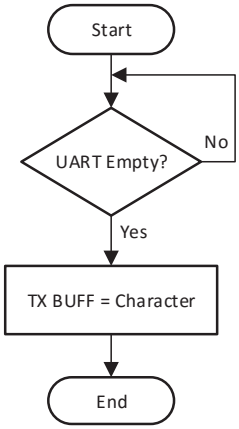


Figure 7.6: UART data transmit flowchart (polling)

7. Compile your code and verify that it does not contain any errors.
8. Run your code and verify if the PC received the transmitted character.
9. Now, modify your code to send over the UART the string: "Hello World!". Develop a subroutine for your code. Use a delimiting character to denote the end of the message (eg. EOT or CR).

7.2.2 Sending and Receiving Characters via UART

In this part, you will send and receive characters from a PC using an asynchronous serial communication.

Follow the steps outlined below:

1. Use the same setup as the basic exercise developed before "Asynchronous Serial Communication (UART)".
2. Connect an LCD to your MCU as you did in previous experiments.
3. Open your IDE.
4. Modify the configuration procedure outlined in the previous section to enable, in the control register, the receiver side of the UART module.
5. Write a program that receives a character using interrupts and displays them on the LCD. To complete this task, you could write the received character directly onto the LCD upon reception. Another way could be using a memory buffer, where received characters are stored in the buffer and having a function to dump the buffer contents into the LCD.
6. Compile your code and verify that it does not contain any errors.
7. Run your code and verify if that the PC is able to send a character to your MCU and the LCD shows the received character.
8. Now, modify your code to receive a 16-character-lower-case message from the PC and display it on the LCD. The message must be returned to the PC via UART in upper case.

7.2.3 Synchronous Serial Communication (I^2C)

In this section you will use an I^2C channel to read the time registers from a Real-Time Clock (RTC) device (DS1307). The DS1307 is a real-time clock-calendar chip that communicates with the MCU through I^2C.

Follow the steps outlined below:

1. Identify and available I^2C port in your MCU and connect the RTC to it according to the schematic shown in Figure 7.7.
2. Open your IDE.
3. Configure the baud rate of your MCU in *low speed* mode (10 kbps):
    - Configure the baud control registers with the prescaler value and baud rate clock source.

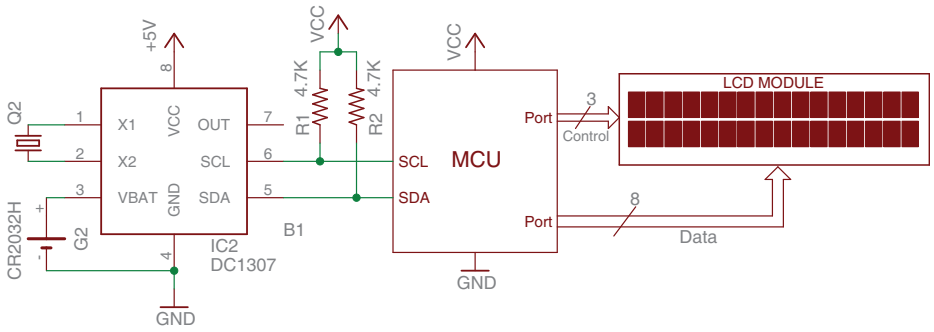


Figure 7.7: Interface for a DS1307 to your MCU via I^{2}C bus.

- Configure the baud rate generator number necessary for the desired baud rate. If your MCU uses a timer for generating the baud rate, programming that timer.

4. Configure the USART for synchronous I^{2}C communication:
   - In the USART control registers choose a synchronous operating mode.
   - Program the corresponding pins to work with I^{2}C protocol.
   - Configure your MCU in Master Mode.
   - Note that to read data from the DS1307 it is necessary to initially send the device's address byte with bit0 in 1. Then, send the register address to be read. Figure 7.8 shows a timing diagram of the reception of a byte from a slave device.
   - Note that to write data on the DS1307 it is necessary to send the device's address byte with bit0 in 0. Then send the register address to be modified and the data that will be stored in the register. Figure 7.9 shows a timing diagram of the transmission of a byte to a slave device.
5. Now, write a program to ask the DS1307 for its current time and display it constantly on the LCD in format "HH:MM:SS". Note that this exercise did not set the time (and date) on the chip, therefore the time value read will correspond to the time elapsed after the last power-up.
6. Compile your code and verify that it does not contain any errors.
7. Run your code and verify if the LCD displays the read time.
8. Later, modify your code to show the time in the first line and the date in the second line of your LCD.

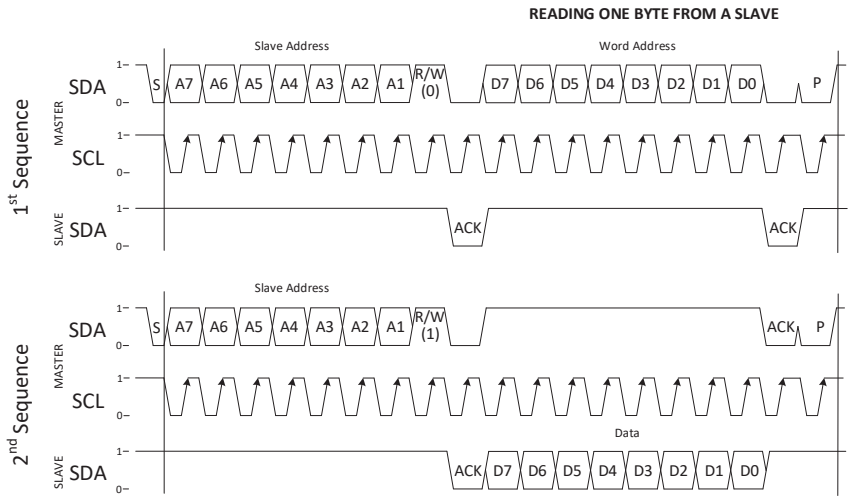


Figure 7.8: Timing diagram to read a byte from a slave (*Source*: Mastering the I^{2}C Bus, V. Himpe)

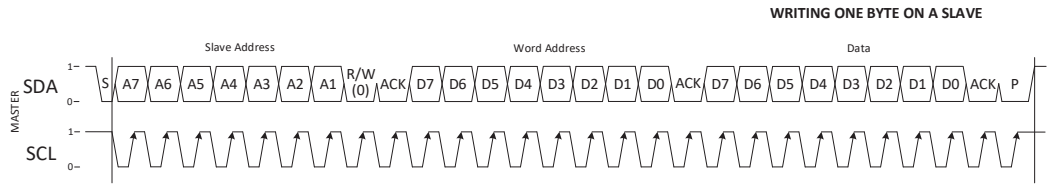


Figure 7.9: Timing diagram to write a byte to a slave (*Source*: Mastering the I^{2}C Bus, V. Himpe)

7.3 Complementary Tasks

7.3.1 Digital Alarm Clock

The activity consists of using a DS1307 to make a programmable alarm clock. The clock shall use keys UP, DOWN, and ENTER to configure the current time, date, and desired alarm upon reset. When the system turns-on, a user shall be able to configure the alarm. Upon setup, the current date shall be displayed on the top line of the LCD and the time on the bottom line. Use the ENTER key to toggle between alarm and current time&date. When the current time matches the alarm time, an audible sound must be produced through a buzzer to indicate that the set time has been reached. The ENTER key must turn the alarm sound off. You can use the block diagram shown in Figure 7.10 as a reference to connect your system.

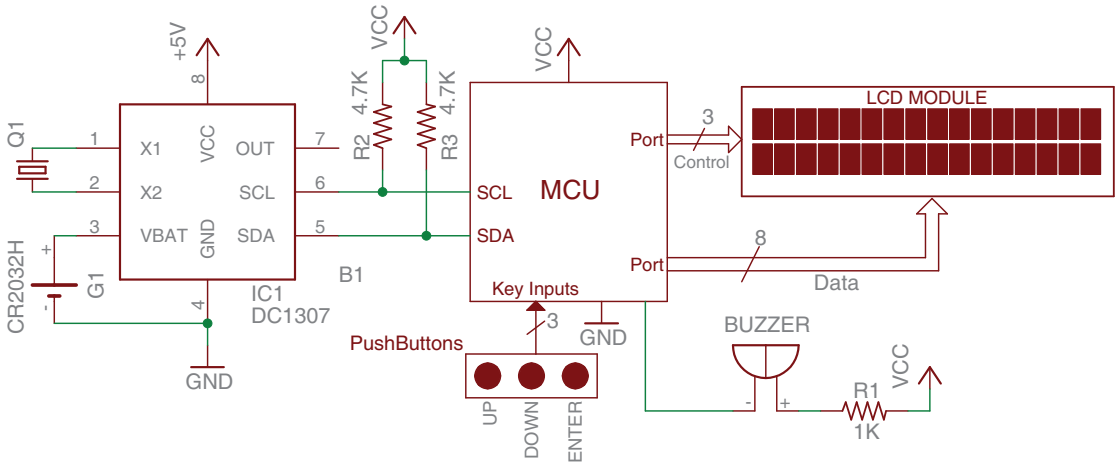


Figure 7.10: Digital Alarm Clock Diagram

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made personally in the lab and including the hardware and software components needed for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with component calculations
- Code listing with comments.
- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Experiment 8

Data Converters (DAC & ADC)

Objectives

- Understanding the uses of DAC and ADC in embedded applications
- Understanding how to operate a DAC
- Using a DAC to generate different voltages and signal waveforms
- Identifying and understanding the architecture of an ADC and how to operate it from an MCU
- Using an ADC module to read analog signals that comes from electronic devices such as sensors

Duration

- 2 Hours in the laboratory and extra time for complementary tasks

Materials

Table 8.1: Bill of materials for completing Lab. 8

Item #	Qty	Description	Reference
1	1	Development board	
2	1	IDE application	
3	1	LCD display: 2 lines, 16 characters	W/HD 44780 Controller
4	1	Light Emitting Diode	5mm Red LED
5	1	1/4W Carbon fill resistor	330 Ω
6	4	1/4W Carbon fill resistor	2.4 KΩ
7	1	1/4W Carbon fill resistor	4.7 KΩ
8	1	1/4W Carbon fill resistor	10 KΩ

Table 8.1: Continued

9	1	Non-polarized ceramic capacitor	0.1nF
10	1	Momentary switch	Pushbutton
11	1	Operational amplifier	LM358
12	1	Analog temperature sensor	LM35
13	1	Digital-to-analog converter	DAC0808

8.1 Introduction

8.1.1 Data Converters

Data converters allow for interfacing digital systems to the analog world. Many embedded applications need to interact with analog processes to either receive information about their status, level, or behavior; or to control their status, level or how they behave. In either case, the discrete nature of a digital system requires converting the data format from or to the analog domain to enable operation. This requirement calls for the usage of data converter circuits. Figure 8.1 illustrates a typical signal processing chain denoting the position of the required data converters.

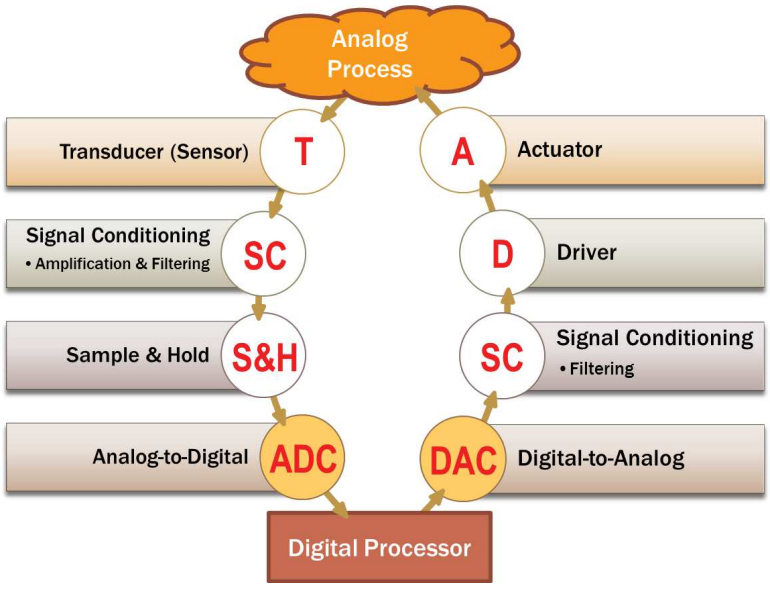


Figure 8.1: Signal Processing Chain

Data converters can be found in the form of Digital-to-Analog Converter or Analog-to-Digital Converter where the first is used for transforming a digital code into a

analog voltage and the second is used for converting an analog voltage into a digital code.

8.1.2 Digital-To-Analog Converters (DAC)

A digital-to-analog converter (DAC) is a device that converts a binary code presented to its input into a discrete voltage value. The specific voltage resulting from a particular digital code will depend on the DAC resolution and the reference voltage it uses. Embedded microcontrollers usually do not include DAC modules. DACs are usually provided via external ICs interfaced to MCUs. A block representation for an n-bit DAC is presented in Figure 8.2 where a digital input is transformed into an analog output.

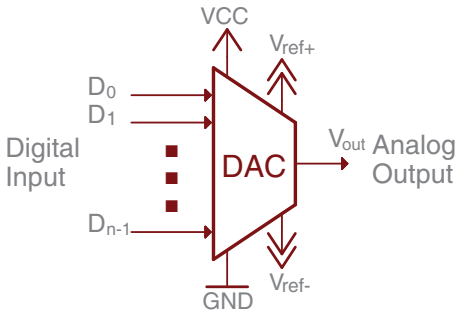


Figure 8.2: DAC Block Diagram

Internal DAC Structure

The internal structure of a DAC is conceptually simple. It includes a voltage reference, a resistor network to break down the reference voltage into binary-weighted voltage values, and some form of analog accumulation to add up the binary-weighted voltages according to the binary code being converted into analog. The most intuitive way of implementing a DAC is provided by a binary-weighted resistor network.

A **Binary-Weighted resistor** DAC uses an operational amplifier in adder configuration to perform the accumulation of the voltages provided by the resistor network as shown in Figure 8.3. In this configuration, the output voltage is composed of the sum of the input voltages where each input uses a different resistor value to represent the binary weights. The output voltage is:

$$Vo = \frac{2R_f}{R} V_R N, \tag{8.1}$$

where N is the digital code represented by the inputs in the DAC and V_R the reference voltage. The major problem with this configuration is the wide range of resistors needed for its construction.

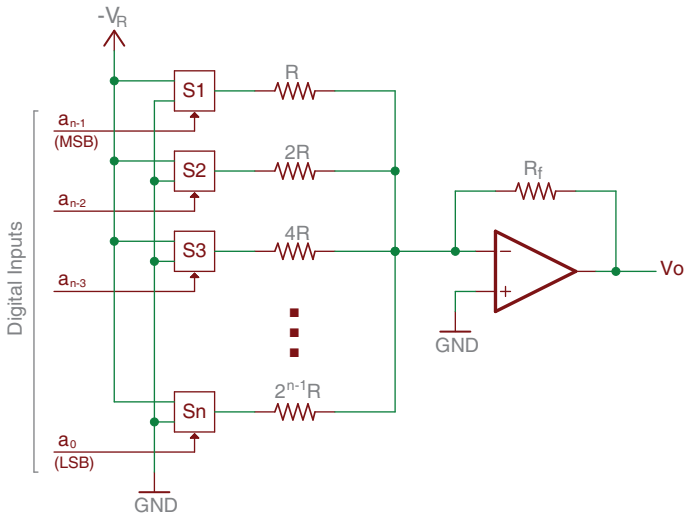


Figure 8.3: Binary-Weighted resistor diagram

An **R-2R or resistor ladder** DAC uses a resistor network made-up of only two different resistor values as shown in Figure 8.4. The set of switches $S_1 \ldots S_n$ allow for connecting each resistor to the Op-Amp adder inputs to generate an output voltage that corresponds to the digital input code specified with lines $a_0 \ldots a_{n-1}$. In this configuration the output voltage is:

$$Vo = \frac{V_{Ref}}{2^n}\left(2^{n-1}a_{n-1} + 2^{n-2}a_{n-2} + \cdots + 2a_1 + a_0\right) \tag{8.2}$$

The advantage of this configuration with respect to the Binary-Weighted resistor circuit is the use of only two resistor values independently from the number of digital inputs.

8.1.3 Analog-To-Digital Converters (ADC)

An analog-to-digital converter (ADC) is a device that converts an input voltage into a digital code. The specific code resulting from a particular analog voltage will depend on the ADC resolution and the reference voltage it uses. Figure 8.5 shows a basic block representation for an ADC module where an analog input, selected through a multiplexer, is converted into a n-bit digital code. The ADC module uses

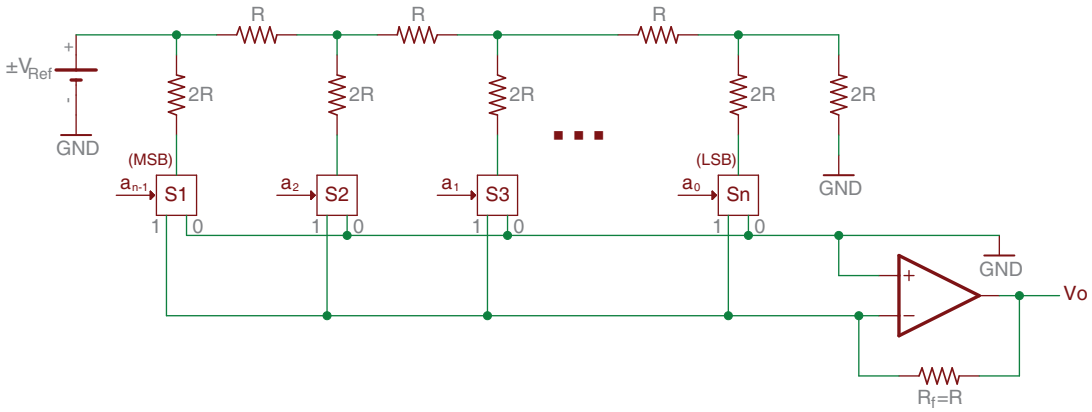


Figure 8.4: R-2R diagram

an input signal to start the conversion (START) and it generates a flag when the conversion has finished (EOC). Most microcontrollers include multi-channel ADC modules among its embedded peripherals but commercial off-the-shelf ADC chips are also available to be interfaced with MCUs.

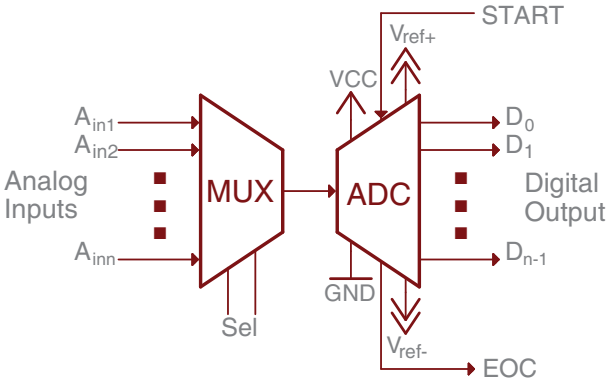


Figure 8.5: ADC Block Diagram

ADC Topologies

ADCs have been implemented using different types of circuits topologies where each topology has its own characteristics, advantages, and disadvantages. For example, Flash ADCs are the fastest ones but consume a large amount of power due the usage of $2^n - 1$ comparators. Likewise, there are other topologies such as two-steps ADC, pipeline ADCs, Slope ADC, sigma-delta converters, among others.

One of the most popular ADC architecture, embedded as a module in many microcontrollers, is the **Successive Approximation (SA) ADC**. This module is composed of a successive approximation register (SAR), a clock signal, a DAC module, and an operational amplifier in comparative voltage mode, as shown in Figure 8.6. In a SA ADC, the n-bits are determined from MSB to LSB in n steps by comparing the analog input with mid levels of successive region intervals.

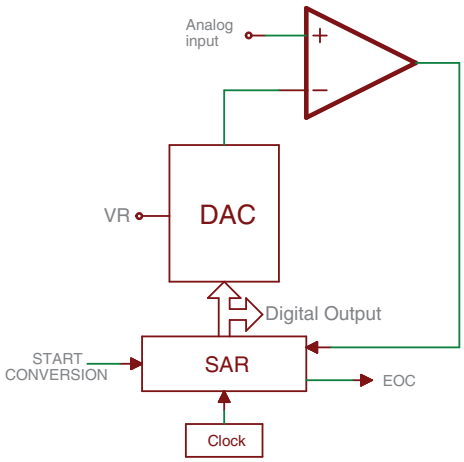


Figure 8.6: Successive Approximation ADC block diagram

The main advantage of the SA ADC is that the circuit complexity and power dissipation are less than those found in most other types of ADC topologies. One of its drawbacks is that eventually the comparator must do a comparison within 1LSB of precision, and precautions must be taken to deal with noise.

The sequential steps in a SA DAC to convert a voltage value is illustrated in Figure 8.7, which corresponds to 4-bit DAC. In step 1, the input voltage is compared with the mid level 1000 of the full scale region. Due to the input voltage being lower than the DAC output, the MSB is turned to 0. In step 2, the DAC output corresponds to a 1/4 of the scale (mid level of the already determined region 0100) and as the input voltage is greater, the third bit remains as 1. In step 3, the input voltage is less than the DAC output (0110) converting the second bit from 1 to 0. Finally, in step 4, as the input voltage is greater that the DAC output (0101), the LSB remains 1. This indicates that an N-bit SAR ADC will require N comparison periods. The increment or decrement rate of the bits is controlled by the clock.

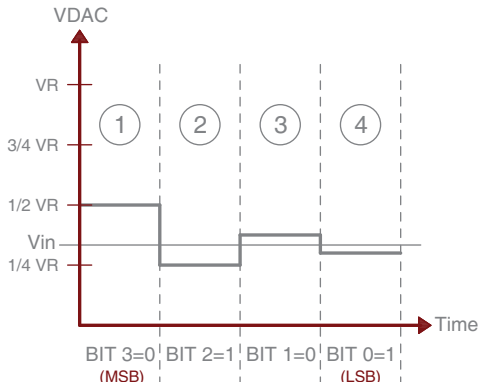


Figure 8.7: SAR Operation

8.2 Basic Exercises

8.2.1 Generating Voltages Using a DAC

The purpose of this exercise is to demonstrate the operation of a DAC by generating different voltage levels using the DAC0808. A DAC0808 is a 8-bits digital-to-analog converter with an analog output current.

Follow the steps outlined below:

1. Connect the DAC to your MCU according to the schematic shown in Figure 8.8. Define the reference voltages for the DAC with VCC as the positive voltage and GND for the negative. The circuit is composed by a DAC0808 that requires a 5V, -15V, and a 8-bit input signal for its operation. An operational amplifier, in current to voltage converter configuration, must be connected to the DAC output to convert the output current in a voltage value.

2. Open your IDE.

3. Configure the MCU pins connected to the DAC as outputs.

4. Make a look-up table with the hexadecimal values in Table 8.2.

5. Write a program that sends the appropriate hexadecimal value to the DAC using a timer function. The timer must change the value that appears in the output port each one second. To handled the timer, you must created an ISR. The binary value sent to the DAC must be displayed on the LCD.

6. Compile and verify that your code does not contain any errors.

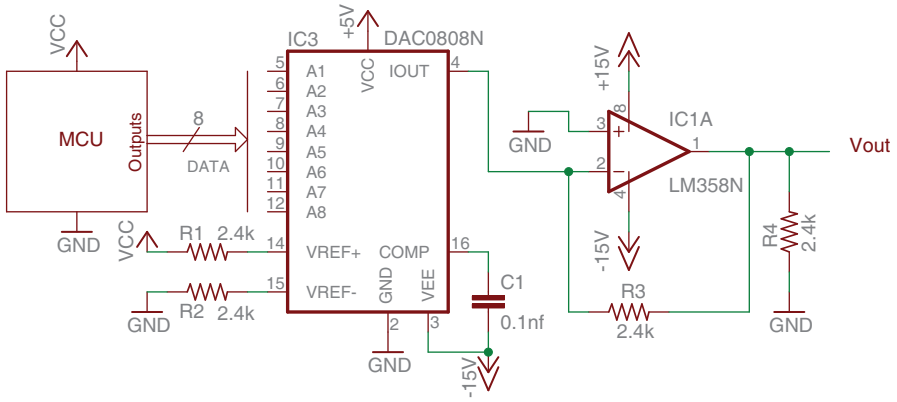


Figure 8.8: Block diagram for DAC connection

7. Run your code and verify if the voltage in the DAC changes.

8. Measure the DAC output voltage in each case and compare it with the expected value from the DAC according to binary number presented in its input. Calculate the percentage of error for each measurement. Complete the Table 8.2 and plot the V_out. Investigate how to calculate the expected output voltage from the DAC ($V_{out} = f\{A_i, VREF\}$).

Table 8.2: DAC Values

Hex Value	Expected Voltage	Measured Voltage	% Error
00			
17			
2E			
45			
5C			
73			
8A			
A1			
B8			
CF			
E6			
FF			

9. Now, modify your code to produce a sinusoidal wave with a frequency of 500Hz and peak-to-peak voltage of 3.3V.

8.2.2 Reading Voltages

In this part, you will Read different voltages from analog devices such as potentiometers using the MCU internal ADC peripheral.

Follow the steps outlined below:

1. Connect the potentiometer to your MCU according to the block diagram in Figure 8.9. The voltage source to be connected to the potentiometer must match the voltage range selected for your MCU ADC module.

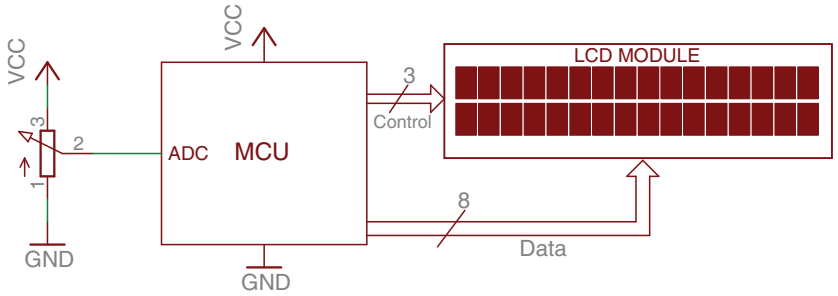


Figure 8.9: Schematic Potentiometer to ADC

2. Verify if the I/O port where the potentiometer is connected has ADC capabilities, if not, move it to an input that does.

3. Open your IDE.

4. Configure the ADC to use full resolution. Take into account the number of bits of your ADC module.

5. Write a code to read the ADC and display the read value in hexadecimal format into the LCD. Use a refresh ratio of 1 second to read the ADC value.

6. Compile and verify that your code does not contain any errors.

7. Run your code and verify if the value displayed in the LCD change when you turn the potentiometer.

8. Now, modify your code to display in the second line of the LCD the decimal voltage value that corresponds to the hexadecimal value being read. Complete the Table 8.3 and compare the voltage that appears on the LCD with the ADC input voltage for each case. Calculate the percentage of error for each measurement and explain the mismatch.

Table 8.3: ADC Values

Input Voltage	Decimal Value	Measured Voltage	% Error
VCC*0.1			
VCC*0.2			
VCC*0.3			
VCC*0.4			
VCC*0.5			
VCC*0.6			
VCC*0.7			
VCC*0.8			
VCC*0.9			
VCC*1.0			

8.2.3 Analog-Digital Dimmer

The objective of this part is controlling the brightness of an LED using a reference voltage from a potentiometer.

Follow the steps outlined below:

1. Connect the potentiometer and the LED according to the schematic in Figure 8.10. Take into account the same consideration, mentioned in the previous exercise, for the voltage source to be connected to the potentiometer.

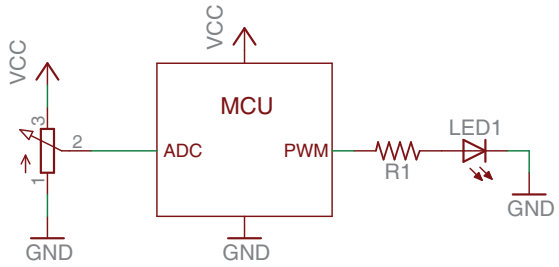


Figure 8.10: Schematic Potentiometer and LED to the MCU

2. Verify if the I/O port where the LED is connected has PWM capabilities, if not, move the LED to an output pin that has.

3. Open your IDE.

4. Configure the PWM module to produce a square signal of 1000Hz. Do not

forget to configure the Timer associated and the correct PWM mode of operation.

5. Configure the ADC to use the full resolution. Take into account the number of bits of your ADC Module.

6. Produce a code that modifies the LED brightness proportionally to the voltage value being read by the ADC module in your MCU. When the ADC reads a decimal value of 0V, the LED brightness must be set to 0% and when it reads a value that corresponds to VCC, the brigthness must be set to 100%.

7. Compile your code and verify that it does not contain any errors.

8. Run your code and verify if the brightness in the LED changes with the corresponding reference voltage.

9. Now, modify your code and circuit to include an LCD. The LED brightness level must appear on the first line and a warning message must appear on the second line when the lower or maximum brightness level are reached.

8.3 Complementary Tasks

8.3.1 Digital Temperature Meter

The activity consists of using the ADC module in your MCU and the temperature sensor LM35 to create a digital temperature meter. The system must have the capability of displaying the current ambient temperature into an LCD in Celsius (°C) or Fahrenheit (°F) units. To select the temperature unit, provide a pushbutton that allows toggling between the two temperature units. Take into consideration that the temperature range for the application is between 0°C (32°F) to 45°C (113°F). You must use the full range of your ADC to measure the temperature; The minimum value of your ADC must correspond to the lowest temperature and the maximum value must correspond to the highest temperature. You shall use a signal conditioner to adapt the signal from the temperature sensor to your ADC range. Investigate how to implement a signal conditioner (sc) with operational amplifiers. You can use the block diagram shown in Figure 8.11 as a reference for connecting your system.

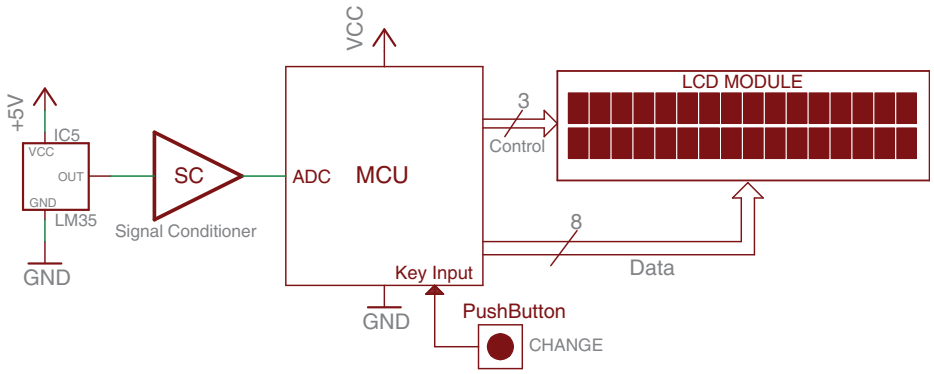


Figure 8.11: Digital temperature meter diagram

Presentation and Report

Each basic exercise and complementary task must be presented to the TA before the initiation of the next laboratory experiment. The demonstration must be made personally in the lab and including the hardware and software components needed for its completion.

An electronic report that includes the following information about the complementary task must be presented to the TA:

- Software plan and explanation (pseudocode or flowchart)
- Connection schematic with component calculations
- Code listing with comments.
- Additional information used to complete the task (Web pages, datasheets, books, etc.)

Appendix A

Using an MCU to Read Incremental Encoders

An incremental encoder is an electromechanical device that can be used to measure the movement, position, and displacement of a rotational or linearly moving mechanical component. In rotations parts, an incremental encoder can measure rotation directions and converts angular position and angular displacement into digital pulses. An incremental encoder has two outputs in quadrature, i.e., two outputs with a 90° phase shift between them. These outputs results from the optical or mechanical detection of two patterned tracks with a 90° geometric shift between them. Sample tracks and pulse trains are illustrate in Figure A.1.

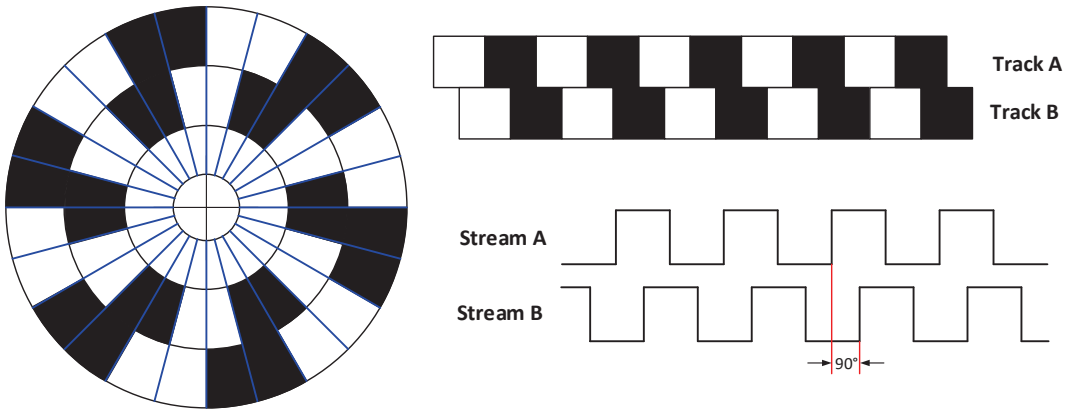


Figure A.1: Output waveforms of an incremental encoder

Detecting movement in either direction from the signal stream only requires determining the order of signal edges in streams A and B and encoding them with a binary code. The positions of the shaded regions generate a Gray binary code.

Figure A.2 shows the waveform streams A and B labeled for rotations sequences in counter clockwise (CCW) and clockwise (CW) directions. Red and blue arrows denote the edge sequences for each direction.

A sequence begins when signals A and B are in the same state. For example, in

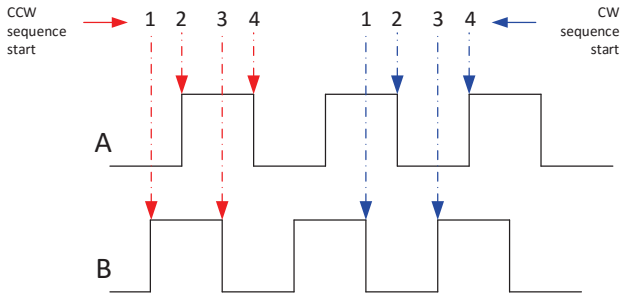


Figure A.2: Rotation sequence for CCW and CW directions

Figure 2, the CCW sequence begins at state '00' while the CW begins at '11'. New sequence values are detected through signal edges in either of the streams. A sequence ends when its four states have been generated. At this point a new sequence begins by repeating the codes from the initial states. Measuring the time between state changes allows obtaining the rotation speed. Tables A.1 and A.2 below show the sequences for CCW and CW rotations.

Table A.1: CCW sequence values

A	B	Value
0	0	0
0	1	1
1	1	3
1	0	2

Table A.2: CW sequence values

A	B	Value
1	1	3
0	1	1
0	0	0
1	0	2

To read the state values and detecting the sequence changes with an MCU it requires using two interrupts enabled i/O lines. The I/O lines levels indicate the state code and the interrupt capability allows detecting code changes. The interrupt needs to be configured so that it is triggered by any change in the encoder lines. This shall allow detecting both, the rising and falling edges of either signal. If the rotational speed were also interest, a timer could be used to measure the time between edges. Multiplying the time between edges (t_{edge}) by the number of the steps in the wheel yields the rotational period. For the sample wheel illustrated in Figure 1, two consecutive interrupts represent 1/16 of the wheel rotation, thus 16 times t_{edge} is one revolution.

To determine the rotation direction it is required to know the state of A and B before and after the occurrence of an edge. Combining the two two-bit codes, a four-bit identifier is obtained, which provides for any possible result in the sequence. Tables A.3 and A.4 list the values for the CCW and CW sequences. Not that each 4-bit code represents a state change in which only one bit changes before and after the edge. Recall that the encoder produces a Gray sequence, and therefore only one

bit is allowed to change between consecutive states.

Table A.3: 4-bit codes for CCW sequence

A_{old}	B_{old}	A_{new}	B_{new}	#(old:new)
0	0	0	1	1
0	1	1	1	7
1	1	1	0	14
1	0	0	0	8

Table A.4: 4-bit codes for CW sequence

A_{old}	B_{old}	A_{new}	B_{new}	#(old:new)
1	1	0	1	13
0	1	0	0	4
0	0	1	0	2
1	0	1	1	11

To write a software function for determining the rotation direction and wheel position, we could use a lookup table (LUT). The 4-bit values resulting from the before and after codes could be used as to index the table, and the entries would be either +1, -1, or 0 for representing CW, CCW or no movement conditions, respectively. Table A.5 shows such a LUT. Assuming the wheel started moving form a known "home" position (index hole and detector might be needed), adding the value fetched from the lookup table on each edge interrupt, we can have the absolute wheel position any time.

Table A.5: Lookup table to detect the direction of rotation and absolute position

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	0	+1	-1	0	-1	0	0	+1	+1	0	0	-1	0	-1	+1	0

Bibliography

[1] Enrique Palacios Municio, Fernando Remiro Domínguez, and Lucas J López Pérez. *Microcontrolador PIC16F84: desarrollo de proyectos.* México, DF:. Alfaomega;, 2009.

[2] Manuel Jimenez, Rogelio Palomera, and Isidoro Couvertier. *Introduction to Embedded Systems using Microcontrollers and the MSP430.* Springer, 2014.

[3] Eduardo García Breijo. *Compialdor C CCS y simulador Proteus para Microcontroladores PIC.* España, Barcelona: Marcombo S.A., 2009.

[4] Maxim Integrated. Understanding sar adcs: Their architecture and comparison with other adcs. Technical report, 2001.

[5] Firas Mohammed Ali. *Experiments in Computer and Microcontroller Applications.* Department of Electrical Engineering, University of Technology, 2013.

[6] Berkley Lab. *Electrical Safety Manual.* Lawrence Berkley National Laboratory, 2015.

[7] EUV. *High Voltage Safety Manual.* Colorado State University.

[8] NIST. *EEEL Safet Rules for Moderate and High Voltages.* National Institution of Standars and Technology, 2008.

[9] Environmental Health & Safety. *Electrical Hazards.* Standford University, 2004.

[10] Samuel M. Goldwasser. *Safety Guidlines for High Voltage and/or Line Powered Equipment*, 2010.

[11] Raymond M Fish, Leslie Alexander Geddes, and Charles F Babbs. *Medical and bioengineering aspects of electrical injuries.* Lawyers & Judges Publishing Company, 2003.

[12] American Heart Association, International Liaison Committee on Resuscitation, et al. Guidelines 2000 for cardiopulmonary resuscitation and emergency cardiovascular care, an international consensus of science. *Circulation*, 102, 2000.

[13] Department of Physics and Astronomy. *Electric Shock*. Georgia State University, 2014.

[14] Hd44780u (lcd-ii). HITACHI. https://www.sparkfun.com/datasheets/LCD/HD44780.pdf. Accessed: 2016-03-25.

[15] Lcd 16x2 (wh1602b2-tm1-et#). http://www.mouser.com/ds/2/272/-364177.pdf. Accessed: 2016-03-25.

[16] Ps1240p02ct3. TDK. https://product.tdk.com/info/en/catalog/datasheets/ef532_ps.pdf. Accessed: 2016-03-25.

[17] Dc56-11ewa. KINGBRIGHT. http://www.us.kingbright.com/images/catalog/spec/DC56-11EWA.pdf. Accessed: 2016-03-25.

[18] Wp154a4sureqbfzw. KINGBRIGHT. https://www.kingbrightusa.com/images/catalog/spec/WP154A4SUREQBFZGW.pdf. Accessed: 2016-03-25.

[19] Rpr-220. ROHM. http://rohmfs.rohm.com/en/products/databook/datasheet/opto/optical_sensor/photosensor/rpr-220.pdf. Accessed: 2016-03-25.

[20] Ds1307. MAXIM INTEGRATED. http://datasheets.maximintegrated.com/en/ds/DS1307.pdf. Accessed: 2016-03-25.

[21] Max3232. MAXIM INTEGRATED. http://pdfserv.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf. Accessed: 2016-03-25.

[22] Parallax standard servo (#900-00005). PARALLAX. https://www.parallax.com/sites/default/files/downloads/900-00005-Standard-Servo-Product-Documentation-v2.2.pdf. Accessed: 2016-03-25.

[23] L293d. TEXAS INSTRUMENTS. http://www.mouser.pr/ProductDetail/Texas-Instruments/L293DNE/?qs=sGAEpiMZZMtYFXwiBRPs0wSafWlCmJbc. Accessed: 2016-03-25.

[24] Dac0808. TEXAS INSTRUMENTS. http://www.ti.com/lit/ds/symlink/dac0808.pdf. Accessed: 2016-03-25.

[25] Lm35. TEXAS INSTRUMENTS. http://www.ti.com/lit/ds/symlink/lm35.pdf. Accessed: 2016-03-25.

Appendix B
Laboratory Tests

B.1 Control Group Laboratory Pre- and Post-tests

LABORATORY N°1
INTRODUCTION AND HIGH VOLTAGE SAFETY TUTORIAL

1. Explain with your own words, what is high voltage?

2. Indicate the average value of the human skin resistance.

3. Write a formula to calculate the total human body resistance.

4. Indicate at least two fatal high voltage injuries and briefly explain them.

LABORATORY N°2
IDE, ASM/C Programming & IO

1. Explain with your own words:
What is a code editor?

What is a compiler?

2. Illustrate the two basic configurations to connect a push button to a MCU input pin.

Pull-Up	Pull-Down

3. Sketch a flow diagram to turn on and turn off an LED every 0.5 seconds.

4. Complete the following information related with the LCD used in the laboratory experiment (Liquid Crystal Display).

Number of Control Pins _____
Number of Data Pins _____
Number of screen lines _____
Number of Character per line _____
Number of operation modes _____

LABORATORY N°3
INTERRUPT & SWITCH DEBOUNCING

1. Explain with your own words:
What is an interrupt?

2. Describe briefly the procedure to configure interrupts in a typical MCU.

3. Explain one of the two techniques used to prevent the debouncing phenomena in a switch or button.

4. Compare the two common techniques used to prevent the bouncing in term of external components needed. Which is better and explain why.

LABORATORY N°4
TIMERS AND APPLICATIONS

1. Explain in your own words:
What is a Timer and how does it works?

2. If you have a source clock of 1MHz, estimate the maximum delay time that you can obtain if you are using a 10-bit timer?

3. Sketch a flow diagram for the operation of a timer by polling.

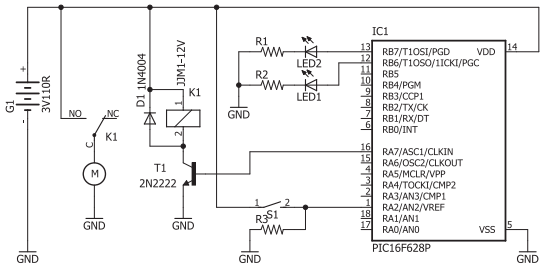
4. Taking into account the basic elements of a timer (Pre-scaler, counter, comparator and clock sources) sketch the block diagram of a timer module.

LABORATORY N°5
LOW-POWER MODES, LED DISPLAY TECHNIQUES & KEYPADS

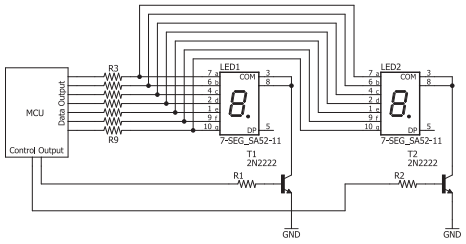
Random ID: _____

1. Explain in your own words:
What is the function of the low-power mode? List three modules or parts of an MCU affected by the low-power mode.

2. Given the following circuit, enclose the path that you consider the most suitable for measuring the MCU's total current and explain your choice.



3. Given the following circuit, list or deduct the sequence of steps to display the number 15 into the two 7-segments.



a.
b.
c.
d.
e.
f.
g.
h.
i.

4. Given a 3x4 keypad, sketch a schematic to properly connect a keypad to the MCU.

LABORATORY N°6
INTRODUCTION TO SERIAL COMMUNICATIONS

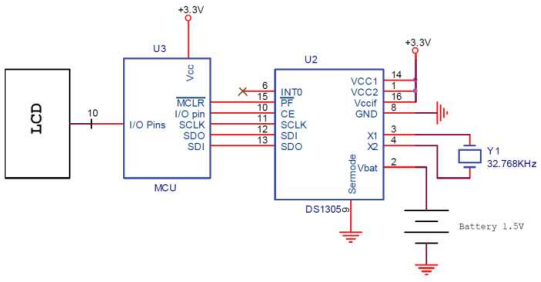
Random ID: _____

1. Explain in your own words:
What is the difference between Synchronous and Asynchronous Serial communications? Sketch the block diagram for both cases.

2. List at least 4 serial communication protocols
   a.
   b.
   c.
   d.

3. A communication process needs to be established between two devices that are separate by a distance of 50 meters. One device has 5 free pins to establish communications, but the other only have 3 pins. The wire connection is carried out with a low capacitance cable. Which serial communication protocol would you recommend to use, and why?

4. In the SPI serial communication protocol, there are many necessary signals to establish a communication between two devices as shown in the figure below. Explain the function of each signal.



B.2 Experimental Group Laboratory Pre- and Post-tests

LABORATORY N°1
HIGH VOLTAGE SAFETY TUTORIAL

1. Explain with your own words, what is high voltage?

2. Indicate the average value of the human skin resistance.

3. Write a formula to calculate the total human body resistance.

4. Indicate at least two fatal high voltage injuries and briefly, explain them.

LABORATORY N°2
IDE, GPIO, and LCD

1. Explain with your own words:

- What is a code editor?

- What is a compiler?

2. Illustrate the two basic configurations to connect a push button to an MCU input pin.

Pull-Up	Pull-Down

3. Sketch a flow diagram to turn On a LED during 0.25 seconds and later turn Off the LED during 0.5 seconds. Do not forget the peripheral configuration phase.

4. Complete the following information related with the LCD used in the laboratory experience (Liquid Crystal Display).

- Number of control pins _____
- Number of data pins _____
- Number of screen lines _____
- Number of character per line _____
- Number of operation modes _____

LABORATORY N°3
INTERRUPT, SWITCH DEBOUNCING, and KEYPAD

1. Explain with your own words:

- What is an interrupt?

2. List the steps to configure interrupts in a typical MCU.

a) _____
b) _____
c) _____
d) _____

3. Explain one of the two common techniques used to prevent the bouncing phenomena in a switch or button.

4. Given a 4x3 keypad, sketch a block diagram that describe the connections to be made if the keypad will be connected to an MCU. Do not forget the supply voltages and extra components such as diodes and resistors.

LABORATORY N°4
TIMERS and LEDs

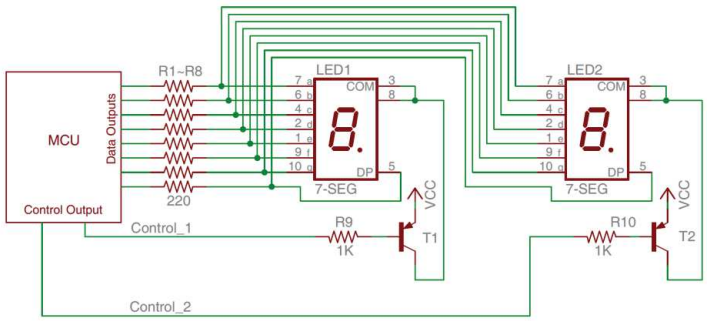
1. Explain in your own words:

- What is a Timer and how does it work?

2. Taking into account the basic elements of a timer (Pre-scaler, binary counter, comparator, compare register, and clock sources) sketch the block diagram of a Timer module.

3. Sketch a flow diagram to turn On and turn Off a LED attached to a pin using a timer by polling. Do not forget the peripheral configuration phase.

4. Given the following circuit, list or deduct a sequence of steps to display the number 15 using two 7-segments.



a)
b)
c)
d)
e)
f)
g)
h)
i)

LABORATORY N°5
LOW-POWER MODES and PWM

1. Explain in your own words:

- What is the function of the low-power mode? List three modules of an MCU affected by a low-power mode.

2. Explain in your own words:

- What is PWM? Explain the meaning of the acronym and the function of a PWM module

3. Taking into account the basic structure of a Timer module, sketch the block diagram of a PWM module. Do not forget include the High count register and the n-bit hardware comparator.

4. List three application that can be implemented with a PWM module.

a) _____

b) _____

c) _____

LABORATORY N°6
MOTORS INTERFACING

1. Explain in your own words:

- What is an electric motor? Also, list three machines or applications that use electric motors.

2. A DC motor can be controlled an H-bridge. Sketch the schematic of an H-Bridge using transistors, diodes, and resistors. Do not forget the supply voltages, and input signals.

3. List the four main components that are part of a Servo-motor and explain, briefly, how a servo-motor is controlled.

a)
b)
c)
d)

4. Based on the given steps, deduct the missing steps for completing a half step sequence used in the control of a stepper motor. Also, sketch the schematic used to connect a stepper motor to an MCU. Do not forget the supply voltages, and input signals.

	Steps							
Half Step	1	2	3	4	5	6	7	8
(4) Coil	1		0					1
(3) Coil	0		1					0
(2) Coil	0		0					0
(1) Coil	0		0					1

LABORATORY N°7
COMMUNICATIONS & I2C

1. Explain in your own words:

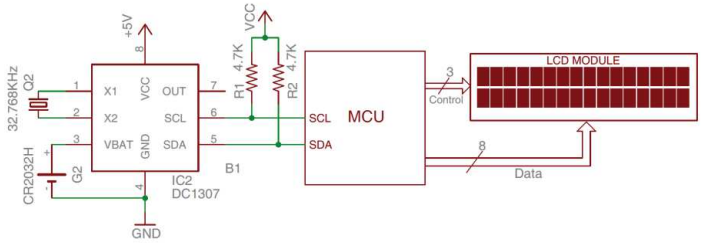
- What is the difference between Synchronous and Asynchronous Serial communications? Sketch the block diagram for both cases.

2. List at least 4 serial communication protocols

a) _____
b) _____
c) _____
d) _____

3. A communication process needs to be established between two devices that are separate by a distance of 50 meters. One device has 5 free pins to establish communications, but the other only have 3 pins. The wire connection is carried out with a low capacitance cable. Which serial communication protocol would you recommend to use, and why?

4. In the I^2C serial communication protocol, there are many necessary signals to establish a communication between two devices as shown in the figure below. Explain the function of each signal.



LABORATORY N°8
DATA CONVERTERS (DAC & ADC)

1. Explain in your own words:

- What is the importance of using DAC and ADC modules in embedded systems? explain the meaning of acronyms

2. Explain the main difference between a DAC and an ADC. Sketch the respective block diagram for each one.

3. An ADC module in an MCU is reading an analog voltage that comes from a temperature sensor that has a resolution of 10mV/°C. The MCU uses a 2.5V as a voltage reference for the ADC module. There are no intermediate components between the MCU and sensor. If the ADC module is generating the following binary number as output "0001 1101", which is the temperature read by the sensor? Express your result in °C

4. A pressure sensor needs to be interfaced with and MCU. The sensor has an output of 1V/KPa and the maximum expected value to be read is 2.5KPa. The requirements specify the uses of a non-inverter amplifier for conditioning the incoming signal. The ADC module is configured to use the full resolution. The MCU uses a 5V supply for its operation. The ADC reference voltages are 0V and 5V. Sketch the schematic for this implementation.

Appendix C
Laboratory Tutorial Presentations

Table C.1 : Laboratory tutorials sections

Tutorials	Sections
High-Voltage Safety	What is High Voltage? Human impedance to current flow High voltage risk and hazards Hazard mitigation Emergency response procedures
Soldering	Soldering gun Soldering rework station Soldering wire Soldering and corrosion Soldering flux Keeping soldering tip clean Preparing wires Soldering throug hole Soldering SMD Desoldering
EagleCAD	Eagle interface Design example Creating a schematic Creating a PCB layout Creating a new component
Logic Analyzer	What is a logic analyzer? Logic analyzer parts Logic analyzer specifications When to and not use logic analyzer Trigger conditions The logic analyzer architecture The logic analyzer operation Clock modes The logic analyzer MS006112A Using the MSO6012A as oscilloscope Using the MSO612A as logic analyzer

Appendix D
Laboratory Modules Reference Manual

Embedded Systems Design
Modules Reference Manual

ICOM4217

Electrical & Computer Engineering Department
University of Puerto Rico at Mayagüez
Mayagüez, PR 00681-9000

Danilo Rojas Manuel Jiménez

2016

Electrical and Computer Engineering Department
University of Puerto Rico at Mayagüez

ACKNOWLEDGMENT

The authors would like to thank Cesar A. Aceros, for his help creating the Latex template for the manual.

DISCLAIMER

Table Of Contents

Module 1

Basic I/O Module

1.1 Materials

The materials needed to assembly the Data Converters module is listed in Table 1.1.

Table 1.1: Boom of materials Module 1

Item	Qty	Description	P/N reference	Supplier
1	4	Pushbutton Case	SW259-ND	DIGI-KEY
2	4	Jumper x5	970	POLOLU
3	2	Piezoelectric Buzzer	810-PS1240P02CT3	MOUSER
4	1	0.1uF Tantalum	581-TAP104K035SCS	MOUSER
5	1	1x40 Male Header	965	POLOLU
6	1	2x40 Male Header	966	POLOLU
7	1	Diode 1N4004	512-1N4004	MOUSER
8	1	Display LCD 2x16	932-MIKROE-55	MOUSER
9	1	Green LED 5mm	67-1098-ND	DIGI-KEY
10	1	Yellow LED 5mm	67-1111-ND	DIGI-KEY
11	2	Red LED 5mm	67-1105-ND	DIGI-KEY
12	1	10K Round Pot	652-3319P-1-103	MOUSER
13	1	1k Ohm Resistor	660-MF1/4LCT52R102J	MOUSER
14	1	2.2k Ohm Resistor	660-MF1/4LCT52R222J	MOUSER
15	1	510 Ohm Resistor	660-MF1/4LCT52R511J	MOUSER
16	2	4.7k Ohm Resistor	660-MF1/4LCT52R472J	MOUSER
17	4	330 Ohm Resistor	660-MF1/4LCT52R331J	MOUSER
18	4	Pusbutton	653-B3F-4050	MOUSER
19	1	2N3904 NPN Transistor	512-2N3904BU	MOUSER

1.2 Description

The basic I/O module is composed of 4 main blocks. It provides access to the most common electronic peripherals such as pushbuttons, LEDS, display LCD, and Buzzer. Figure 1.1 shows the block diagram of the basic I/O module and its four

blocks (16x2 LCD display, 4 LEDS, 4 Pushbuttons, and Buzzer). This module is intended to work with either 3.3V and 5V microcontrollers and it is also designed to allow direct interfacing to the MCU with no intermediate components.

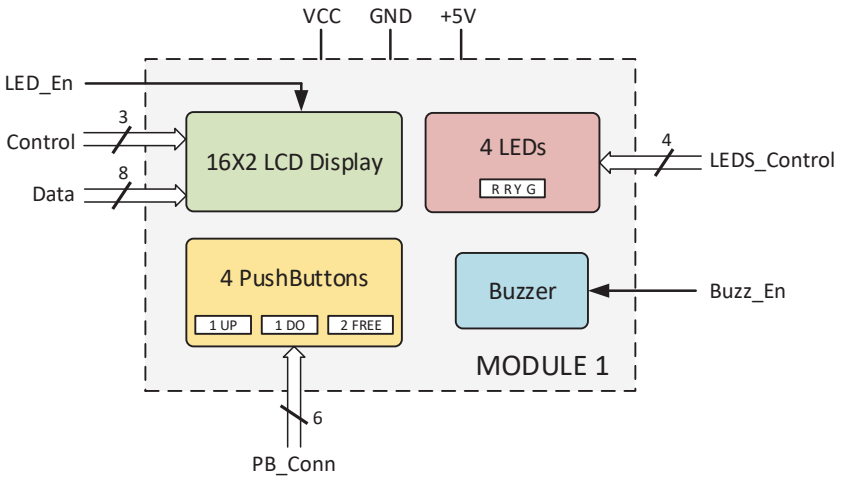


Figure 1.1: Module 1 Block diagram

The schematic of the module is illustrated in Figure 1.7 where all the connections between the different electronics components are depicted. In addition the board layout, divided into components layer, top layer, and bottom layer, can be observed in Figure 1.8, Figure 1.9, and Figure 1.10 respectively. Finally, a real board representation of the EM is presented in Figure 1.11. This representation shows the real board to be used in the development of the laboratory experiments.

1.2.1 Power Supply Setup

To setup the different voltages needed for the proper operation of the system, you must take into account the operation voltage of your MCU. Although some components in the module were chosen to work with both 3.3V and 5V microcontrollers, some of them require a 5V power supply. If your MCU works at 3.3V, you must connect all the power supply pins labeled on the board: 5V, VCC, and GND (using 3.3V for VCC). Also, the jumper in pinheader JP6 must be removed. But, if you are using a 5V microcontroller, you only need to connect one of the two power supply pins (5V or VCC) and put a jumper in pinheader JP6. The power supply pins are highlighted in Figure 1.2. Remember to connect the GND, of your MCU, to the GND in the module to establish the same reference voltage for the entire system.

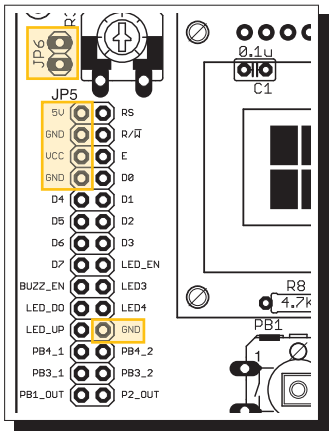


Figure 1.2: Module 1 Power supply pins

1.2.2 16X2 LCD Display

This block is provided with a removable LCD screen (2 lines x 16 characters), 1 10KΩ potentiometer, and a 2N3904 NPN transistor as depicted in Figure 1.3.

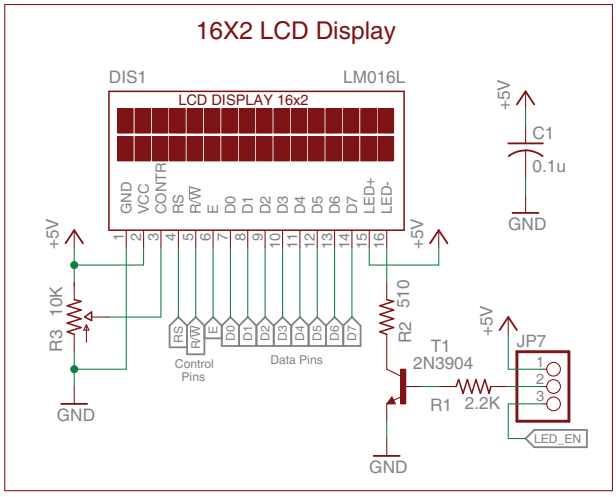


Figure 1.3: 16X2 LCD Display Schematic

To use this block, you need to adjust the LCD's brightness, define the desired operation of the LCD (4-bits or 8-bits), and determine if you want to control the LCD's back-light operation. Follow the steps outlined below to setup this block:

1. To adjust the LCD brightness, you must turn-right or turn-left the potentiometer R3.

2. To define the operation of the back-light, you need to put a jumper between 2 of the three pins on the pinheader JP7. If you put the jumper between the middle pin and the 5V pin, the back-light will permanently be On. But, if you insert the jumper between the middle pin and the LED_EN pin, it will provide you control to the turn On/Off operation. This turn On/Off operation is carried out by a 2N3904 NPN transistor (T1) and a 510Ω resistor (R2). This interface allows to turn-On the back-light with a logic 1 and turn it Off with a logic 0, through the LED_EN pin.

3. To control the LCD display, you must define the desired operating mode. If you want to use the 4-bit mode you must connect the control lines (RS, R/$\overline{W}$ and E) and the four higher Data lines (D4-D7) to your MCU. But, if you want to use the 8-bit mode you must connect the control lines and all Data lines. The pinheader JP5 allow the connection between the LCD pins and your MCU pins (see Figure 1.7).

1.2.3 4 LEDs

This block provides access to 4 LEDs, two in fixed configuration and two in re-configurable configuration, as depicted in Figure 1.4.

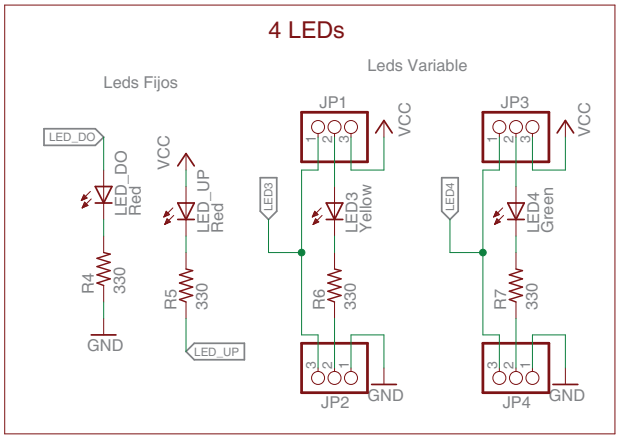


Figure 1.4: 4 LEDs Schematic

To use this module you need to connect your MCU to all the LEDs and set the operating mode of the two re-configurable LEDs. Follow the steps outline below to setup this block:

1. To use the fixed LEDs you must connect your MCU to the pins LED_DO and LED_UP located in pinheader JP5. The LED1 is wired in a pull-down

configuration meaning that it needs a logic 1 to turn it On and logic 0 to turn it Off. On the other hand, LED2 is wired in a pull-up configuration meaning that it needs a logic 0 to turn it On and logic 1 to turn it Off.

2. To setup the re-configurable LED3 you need to insert jumpers in pinheaders JP1 & JP2 and insert jumpers in JP3 & JP4 to setup the LED4. If you want to use the LED3 in pull-down mode, you need to put a jumper between the middle pin and the LED3 pin in pinheader JP1 and a jumper between the middle pin and the GND pin in pinheader JP2. But, if you want to use the LED in pull-up mode, you need to insert a jumper between the middle pin and the VCC pin in JP1 and a jumper between the middle pin and LED3 pin in JP2. To setup the LED4 you have to perform the same procedure outlined to setup the LED3. Remember that the LED4 uses the pinheader JP3 and JP4 in instead of JP1 and JP2.

1.2.4 4 Pushbuttons

This block provides access to 4 pushbuttons, two in fixed configuration and two in free configuration, as depicted in Figure 1.5.

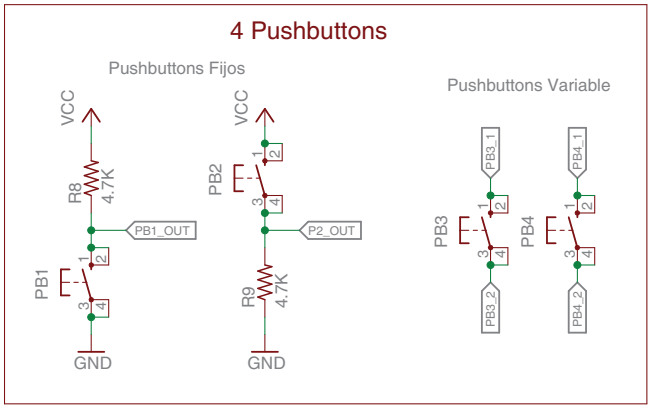


Figure 1.5: 4 Pushbuttons Schematic

To use this module, you need to connect your MCU to all the pushbuttons and set the configuration for the two free pushbuttons. Follow the steps outline below to setup this block:

1. To use the fixed pushbuttons (PBs) you must connect your MCU to the pins PB1_OUT and PB2_OUT in pinheader JP5. The S1 is a PB with a pull-up resistor that constantly sends a logic 1 when it is not depressed and a logic 0

when it is depressed. The output signal can be read in pin PB1_Out. On the other hand, the S2 is a PB with a pull-down resistor meaning that it constantly sends a logic 0 when it is not depressed and a logic 1 when it is depressed. The output signal can be read through pin PB2_out.

2. The two remaining PBs are in free configuration, which means you have to provide external components to use them. The connections pins for each PB are located in JP7. The S3 uses pins PB3_1 and PB3_2 and the S4 uses pins PB4_1 and the PB4_2 for their connections.

1.2.5 Buzzer

This block provides access to a buzzer as depicted in Figure 1.6.

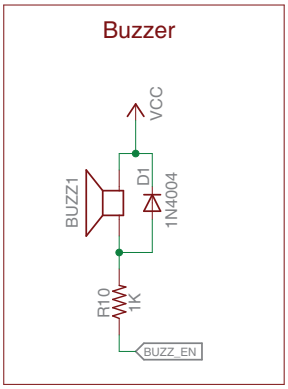


Figure 1.6: BUZZER Schematic

To use this module, you need to connect a pin of your MCU to the BUZZ_EN pin in pinheader JP7, to drive the buzzer. The buzzer needs a sequence of pulses to produce an audible sound. The VCC pin needs to be at the same voltage level of your MCU's operating voltage.

1.3 Schematic

The Figure 1.7 shows the complete schematic of the module were all the connections are described.

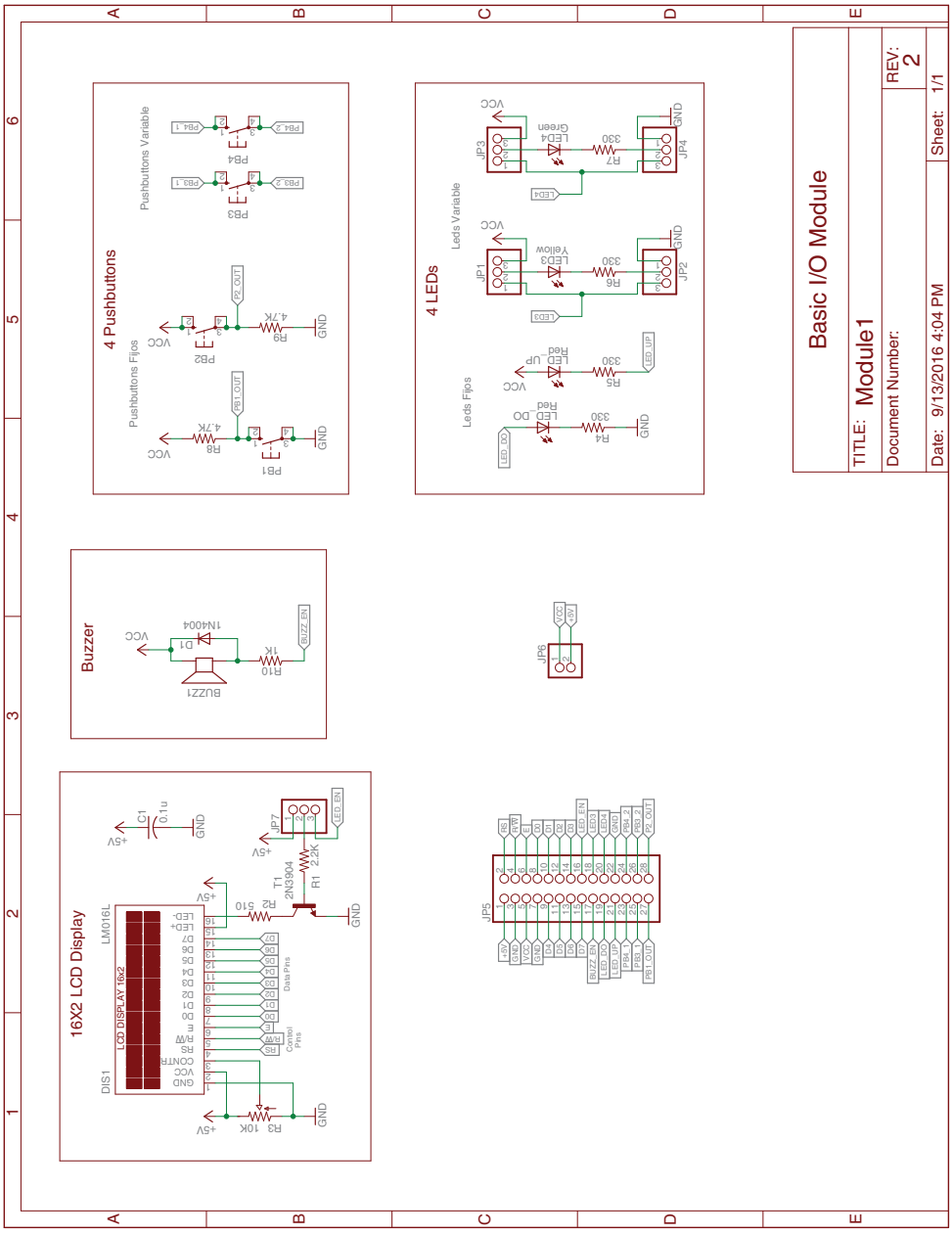


Figure 1.7: Module 1 Schematic

1.4 Board

The Figure 1.8 represent the component layer of the module. This Figure shows how the different elements are arranged on the PCB. The measures are in millimeters.

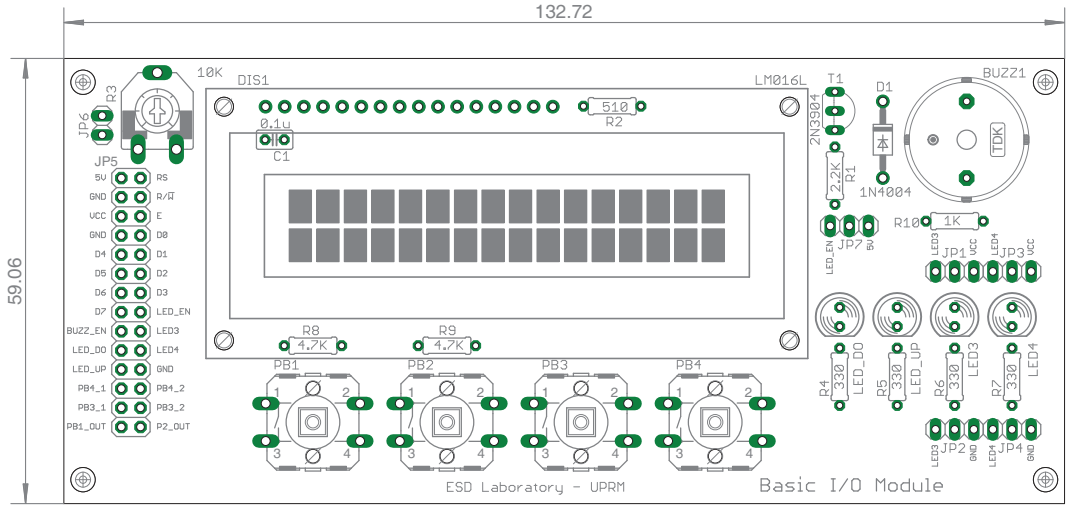


Figure 1.8: Module 1 board - components layer

The Figure 1.9 and Figure 1.10 represents the Top and Bottom layer of the module respectively. These images show the different physical connections between the different components in the module.

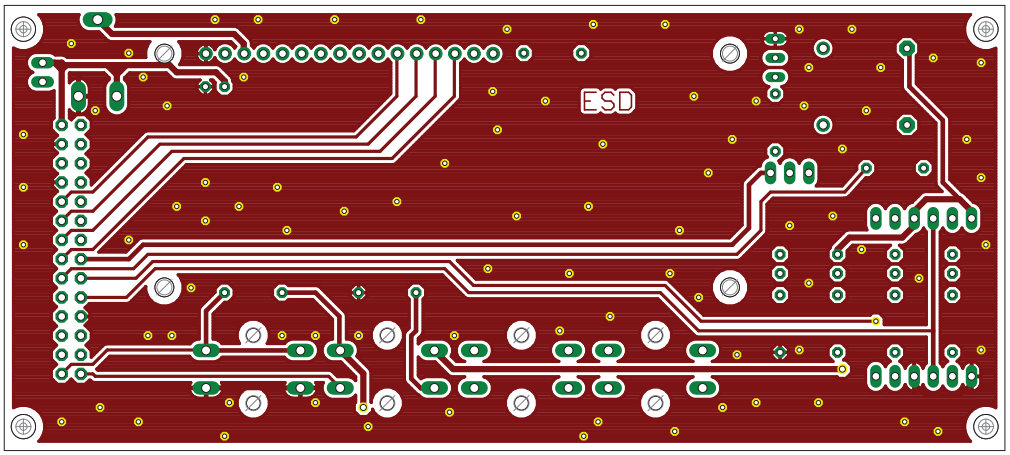


Figure 1.9: Module 1 board - top layer

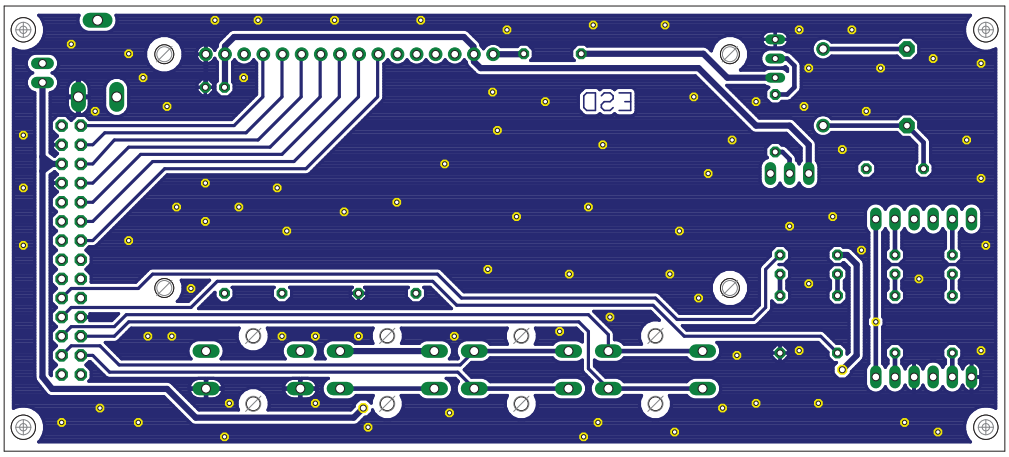


Figure 1.10: Module 1 board - bottom layer

The Figure 1.11 illustrate a real representation of the module's PCB.

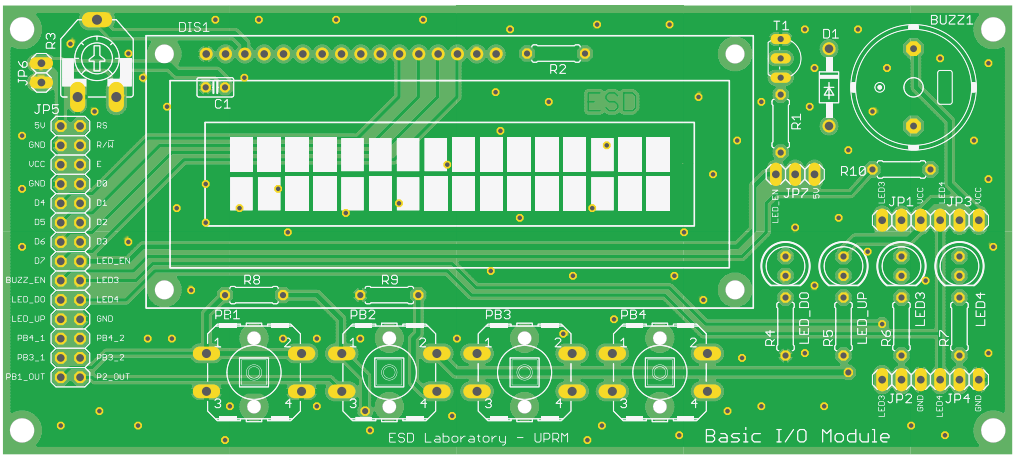


Figure 1.11: Module 1 board

Module 2

Keypad Module

2.1 Materials

The materials needed to assembly the Data Converters module is listed in Table 2.1.

Table 2.1: Boom of materials Module 2

Item	Qty	Description	P/N reference	Supplier
1	1	Jumper x5	970	POLOLU
2	1	1x40 Male Header	965	POLOLU
3	1	1x7 Female Header	1017	POLOLU
4	4	Diode 1N4004	512-1N4004	MOUSER
5	3	4.7K Ohm Resistor	660-MF1/4LCT52R472J	MOUSER
6	1	3x4 Button Keypad	KP-22	All Electronics

2.2 Description

The 3X4 keypad module is composed of 1 main block. It provides access to a one key matrix (keypad). Figure 2.1 shows the block diagram of 3X4 keypad module. This module is intended to work with either 3.3V and 5V microcontrollers and it is also designed to allow direct interfacing to the MCU with no intermediate components.

The schematic of the module is illustrated in Figure 2.4 where all the connections between the different electronics components are depicted. In addition the board layout, divided into components layer, top layer, and bottom layer, can be observed in Figure 2.5, Figure 2.6, and Figure 2.7 respectively. Finally, a real board representation of the EM is presented in Figure 2.8. This representation shows the real board to be used in the development of the laboratory experiments.

2.2.1 Power Supply Setup

To setup the different voltages needed for the proper operation of the system, you must have into account the operation voltage of your MCU. Although some compo-

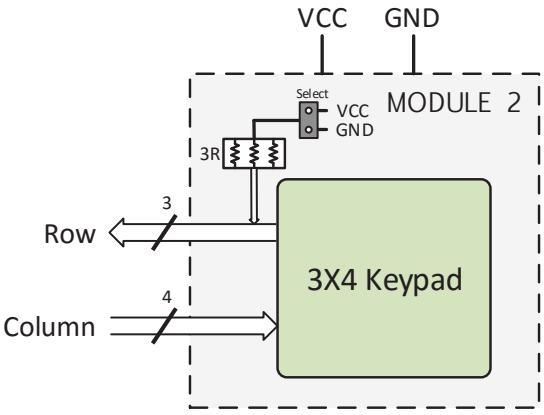


Figure 2.1: Module 2 Block Diagram

nents in the module were chosen to work with both 3.3V and 5V microcontrollers, some of them require a 5V power supply. If your MCU works at 3.3V or 5V, you must connect all the power supply pins labeled on the board: VVC and GND (using the operation voltage of your MCU in VCC). The power supply pins are highlighted in Figure 2.2. Remember to connect the GND, of your MCU, to the GND in the module to establish the same reference voltage for the entire system.

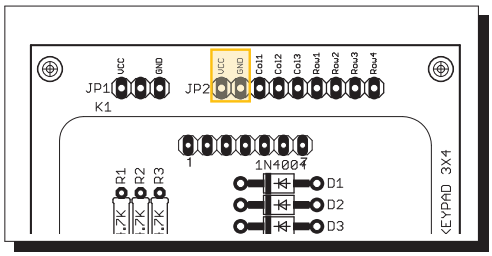


Figure 2.2: Module 2 Power supply pins

2.2.2 3x4 Keypad

This block is provided with a removable 3X4 keypad with 3 pull-up/down resistors, and 4 protection diodes as depicted in Figure 2.3.

To use this block, you need to connect your MCU to all the rows and column pins and select the desired operation for the pull-up/down resistors. To define the operation mode of the resistors, you need to insert a jumper between 2 of the three pins on the pinheader JP1. If you put the jumper between the middle pin and the VCC pin,

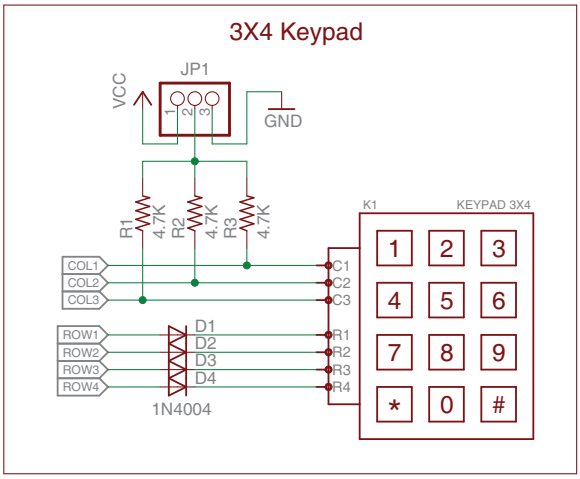


Figure 2.3: 3x4 Keypad Schematic

the resistors will work as pull-up resistors. But, if you insert the jumper between the middle pin and the GND pin, they will work as a pull-down.

Due the configuration of the keypad, if you want to use the resistor in pull-up mode, the diodes (D1 to D4) need to be replaced by shortcuts in their terminals.

2.3 Schematic

The Figure 2.4 shows the complete schematic of the module were all the connections are described.

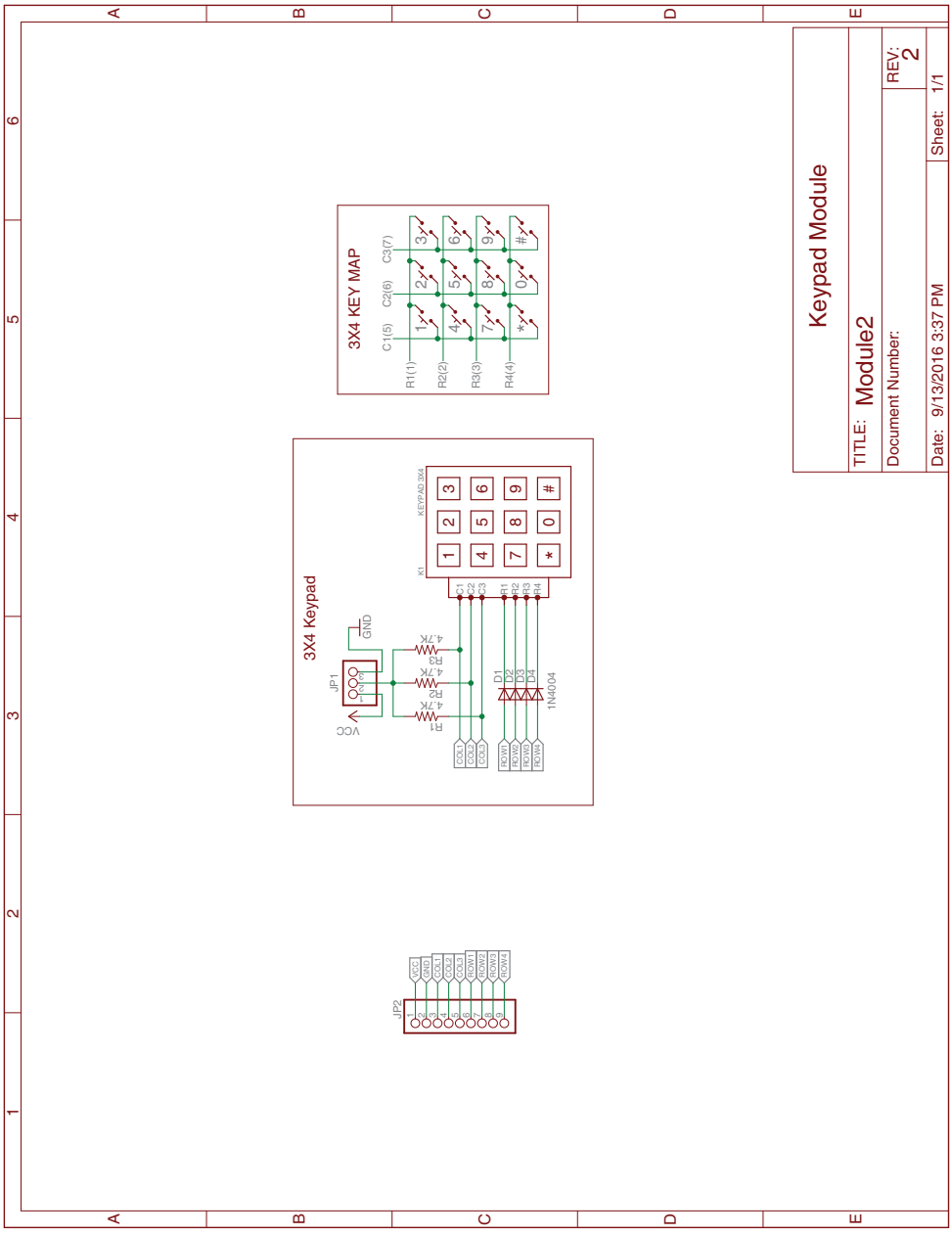


Figure 2.4: Module 2 Schematic

2.4 Board

The Figure 2.5 represent the component layer of the module. This Figure shows how the different elements are arranged on the PCB. The measures are in millimeters.

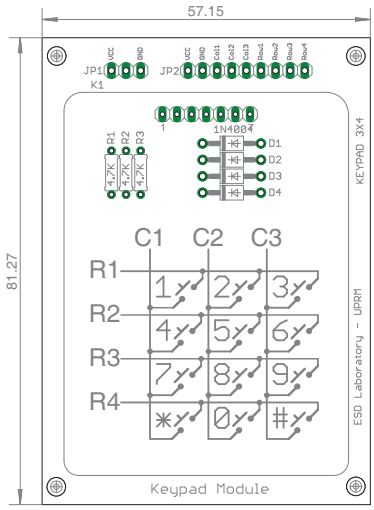


Figure 2.5: Module 2 board - components layer

The Figure 2.6 and Figure 2.7 represents the Top and Bottom layer of the module respectively. These images show the different physical connections between the different components in the module.

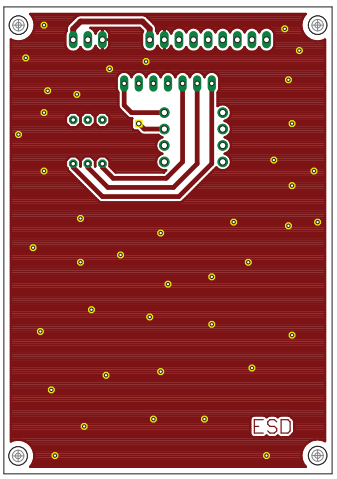


Figure 2.6: Module 2 board - top layer

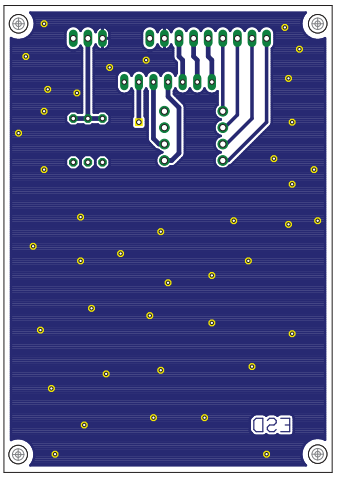


Figure 2.7: Module 2 board - bottom layer

The Figure 2.8 illustrate a real representation of the module's PCB.

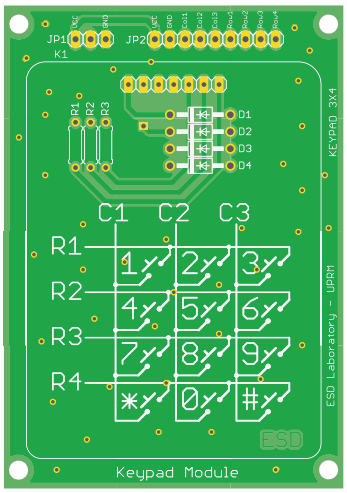


Figure 2.8: Module 2 board

Module 3

Seven Segment Module

3.1 Materials

The materials needed to assembly the Data Converters module is listed in Table 3.1.

Table 3.1: Boom of materials Module 3

Item	Qty	Description	P/N reference	Supplier
1	1	Jumper x5	970	POLOLU
2	1	1x40 Male Header	965	POLOLU
3	1	2x40 Male Header	966	POLOLU
4	1	1x2 Female Header	1012	POLOLU
5	1	1x9 Female Header	1019	POLOLU
6	1	Double 7-Segment	604-DC56-11EWA	MOUSER
7	1	RGB LED 5mm	604-WP154A4SUREQBFZW	MOUSER
8	1	Green LED 3mm	1497-1022-ND	DIGI-KEY
9	1	RPR-220 Optoswitch	755-RPR-220	MOUSER
10	1	12K Ohm Resistor	660-MF1/4LCT52R123J	MOUSER
11	2	220 Ohm Resistor	660-MF1/4LCT52R221J	MOUSER
12	2	1k Ohm Resistor	660-MF1/4LCT52R102J	MOUSER
13	3	510 Ohm Resistor	660-MF1/4LCT52R511J	MOUSER
14	9	330 Ohm Resistor	660-MF1/4LCT52R331J	MOUSER
15	2	2N3906 PNP Transistor	512-2N3906BU	MOUSER

3.2 Description

The seven-segment module is composed of 3 main blocks. It provides access to opto-electronic peripherals such as 7-segment displays, RGB LED, and opto-switches. Figure 3.1 shows the block diagram of the seven-segment module and its three blocks (Dual 7-SEGMENT display, RGB LED, and OptoSwitch). This module is intended to work with either 3.3V and 5V microcontrollers and it is also designed to allow direct interfacing to the MCU with no intermediate components.

The schematic of the module is illustrated in Figure 3.6 where all the connections

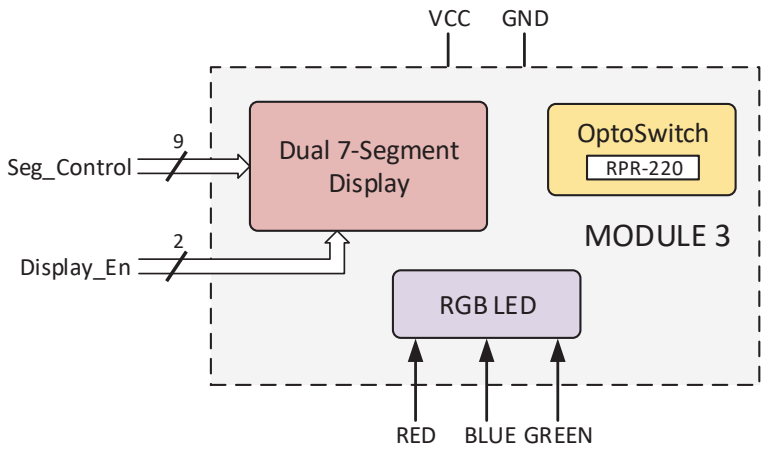


Figure 3.1: Module 3 Block Diagram

between the different electronics components are depicted. In addition the board layout divided into components layer, top layer, and bottom layer, can be observed in Figure 3.7, Figure 3.8, and Figure 3.9 respectively. Finally, a real board representation of the EM is presented in Figure 3.10. This representation shows the real board to be used in the development of the laboratory experiments.

3.2.1 Power Supply Setup

To setup the different voltages needed for the proper operation of the system, you must have into account the operation voltage of your MCU. Although some components in the module were chosen to work with both 3.3V and 5V microcontrollers, some of them require a 5V power supply. If your MCU works at 3.3V or 5V, you must connect all the power supply pins labeled on the board: VVC and GND (using the operation voltage of your MCU in VCC). The power supply pins to be used are highlighted in Figure 3.2. Remember to connect the GND, of your MCU, to the GND in the module to establish the same reference voltage for the entire system.

3.2.2 Dual 7-Segment Display

This block is provided with a double 7-segment display, nine limiting current resistors, and two 2N3906 PNP transistor as depicted in Figure 3.3.

To use this block you need to connect your MCU to all the 7-segment segments and determine if you want to control the turn On/Off operation of both 7-segment displays. Follow the steps outlined below to setup this block:

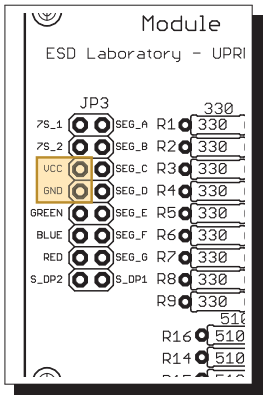


Figure 3.2: Module 3 Power supply pins

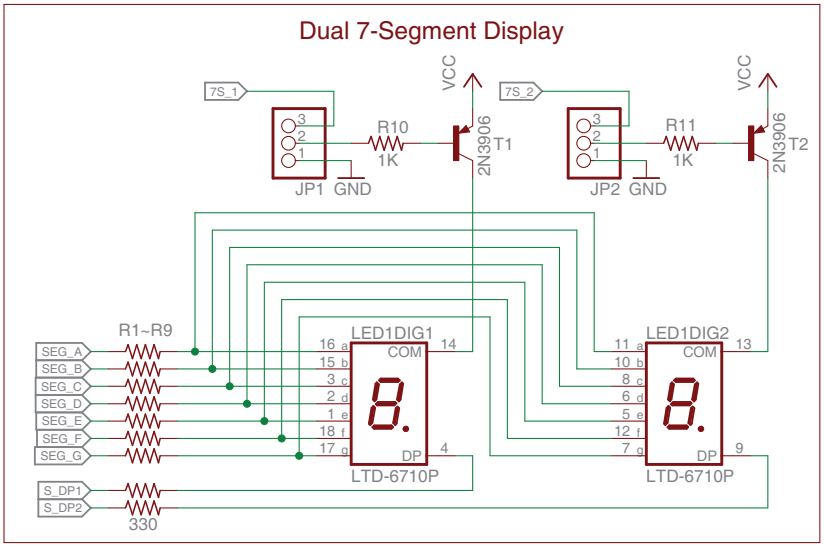


Figure 3.3: Dual 7-Segment Display Schematic

1. To control the displays turn On/Off process, you need to put a jumper between 2 of the three pins in pinheader JP1 for the first display (LED1DIG1) and JP2 for the second display (LED1DIG2). In the case of the LED1DIG1, if you insert the jumper between the middle pin and the GND pin, the display will permanently be On. But, if you insert the jumper between the middle pin and the 7S_1 pin, it will provide you control to the turn On/Off operation. This turn On/Off operation is carried out by a 2N3906 PNP transistors (T1) and a 1 KΩ resistor (R10). This interface allows to turn-On the display with a logic 0 and turn it Off with a logic 1, through the 7S_1 pin. To setup the display LED1DIG2 you need to perform the same procedure used to setup the

LED1DIG1.

2. To activate the segments in both displays you must connect your MCC to the segment pins in the pinheader JP3. The 7-segments displays are common anodes that need a logic 0 to turn On a segment and a logical 1 to turn it Off. The segments between the two displays are connected with the exception of the DP segments pins (S_DP1 and S_DP2).

3.2.3 RGB LED

This block provides access to an RGB (RED-GREEN-BLUE) LED with three limiting current resistors as depicted in Figure **??**.

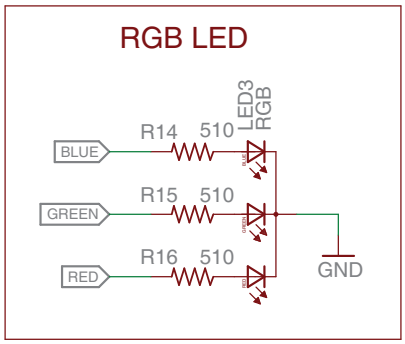


Figure 3.4: RGB Schematic

To use this block you need to connect your MCU to the color control pins BLUE, GREEN, and RED in the pinheader JP3, to drive the RGB LED. The RGB LED is made of common cathode LEDs that needs a logic 1 in one of the three pins to turn-On a particular color and a logic 0 to turn it Off.

3.2.4 OptoSwitch

This block provides access to a standalone RPR-220 optoswitch with a 12KΩ resistor, 330Ω resistor, and a 3mm LED as depicted in Figure 3.5.

This block is used only for demonstrate how the RPR-220 works with the presences of a black or white color object. The RPR-220 has in the transistor collector pin an LED that will turn-On when the sensor detects a white color, otherwise, the LED will turn it Off.

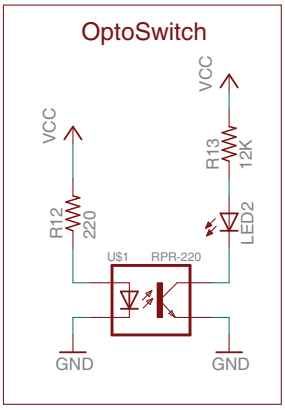


Figure 3.5: Optoswitch Schematic

3.3 Schematic

The Figure 3.6 shows the complete schematic of the module were all the connections are described.

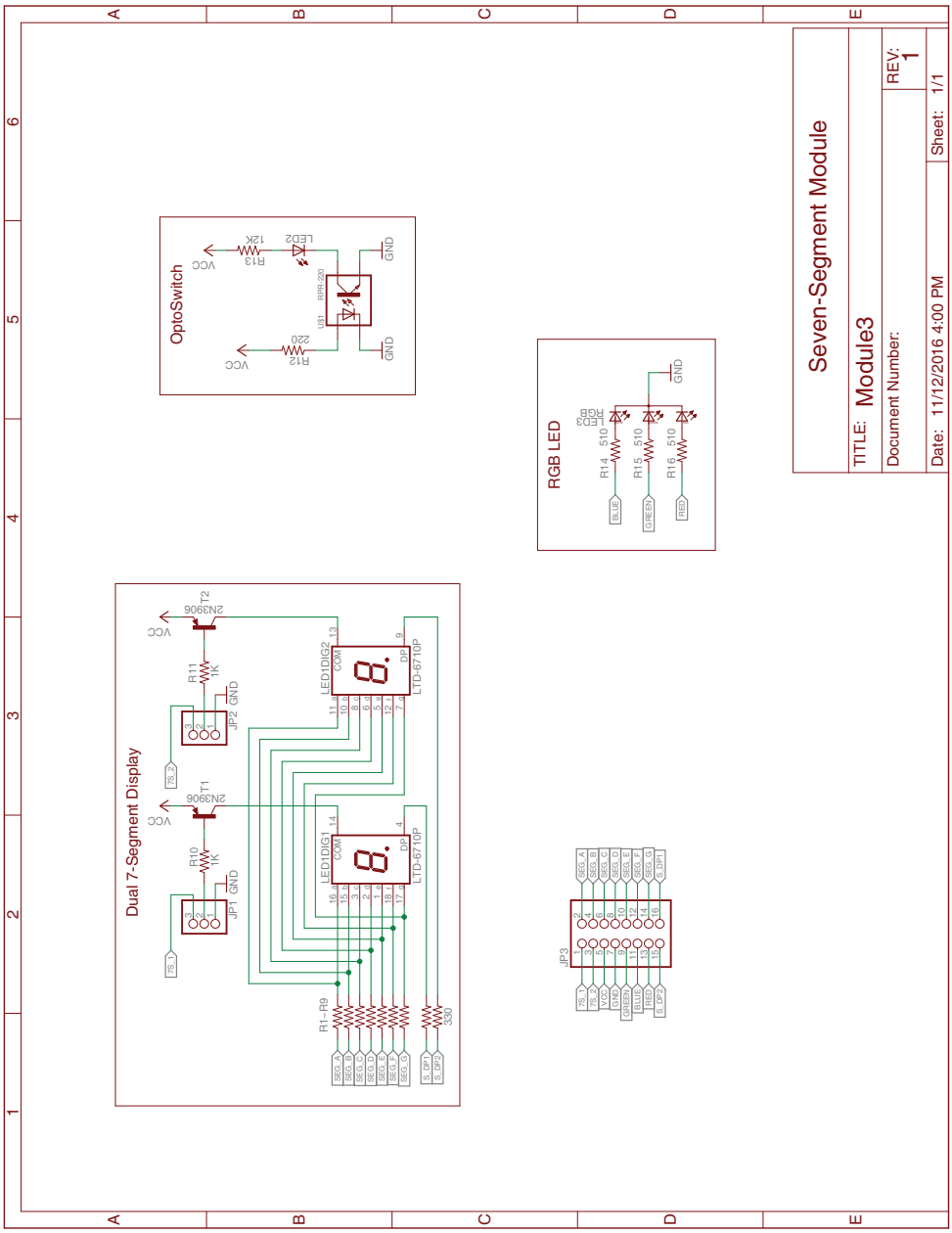


Figure 3.6: Module 3 Schematic

3.4 Board

The Figure 3.7 represent the component layer of the module. This Figure shows how the different elements are arranged on the PCB. The measures are in millimeters.

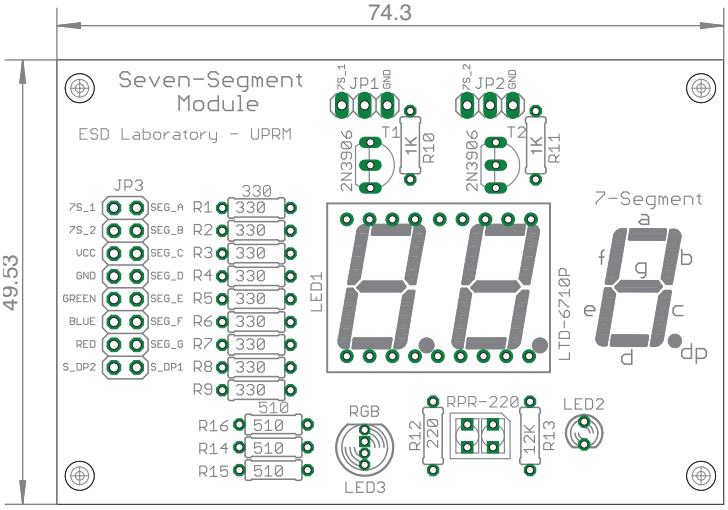


Figure 3.7: Module 3 board - components layer

The Figure 3.8 and Figure 3.9 represents the Top and Bottom layer of the module respectively. These images show the different physical connections between the different components in the module.

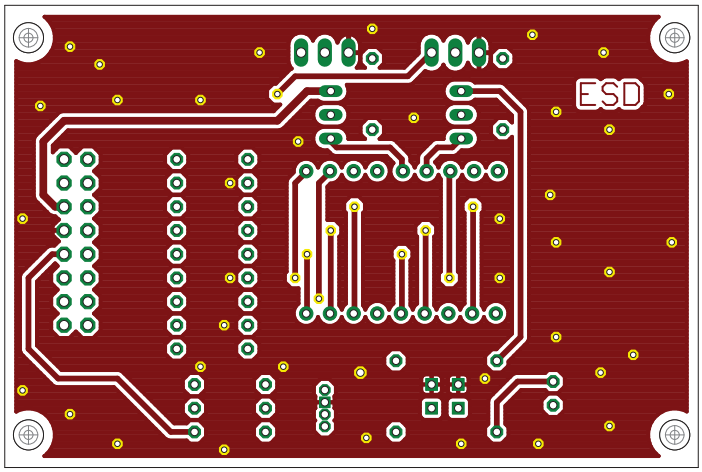


Figure 3.8: Module 3 board - top layer

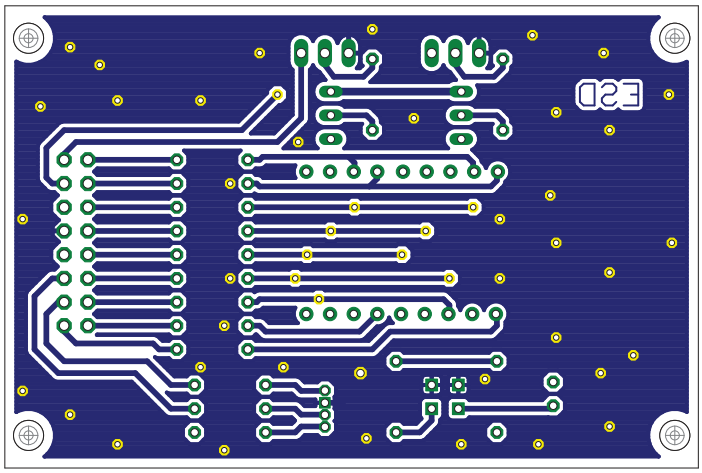


Figure 3.9: Module 3 board - bottom layer

The Figure 3.10 illustrate a real representation of the module's PCB.

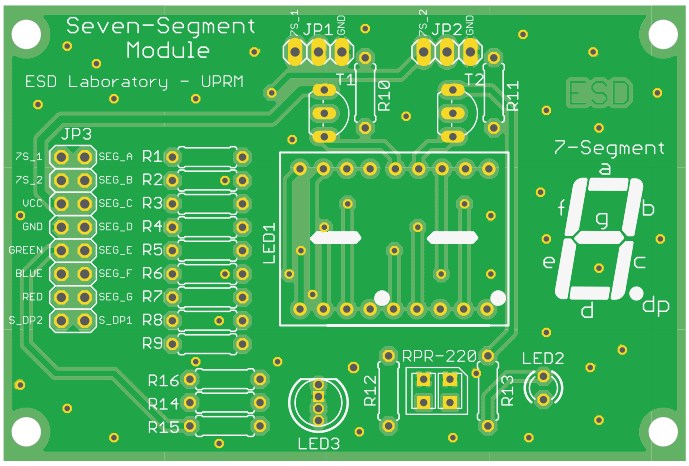


Figure 3.10: Module 3 board

Module 4

Motor Interface Module

4.1 Materials

The materials needed to assembly the Data Converters module is listed in Table 4.1.

Table 4.1: Boom of materials Module 4

Item	Qty	Description	P/N reference	Supplier
1	1	Jumper x5	970	POLOLU
2	1	0.1uF Tantalum Cap.	581-TAP104K035SCS	MOUSER
3	1	2Pins Screw Con.	571-1776493-2	MOUSER
4	1	3Pins Screw Con.	ED1976-ND	DIGI-KEY
5	1	Fuse Holder	693-0031.8201	MOUSER
6	1	1x40 Male Header	965	POLOLU
7	1	Dip16 IC Base	AE9992-ND	DIGI-KEY
8	1	Dip18 IC Base	AE9995-ND	DIGI-KEY
9	1	Dip6 IC Base	AE1485-ND	DIGI-KEY
10	5	Diode 1N4004	512-1N4004	MOUSER
11	1	1.5 Amp Fuse	504-BK/GMC-1.5-R	MOUSER
12	1	L293D Motor Driver	595-L293DNE	MOUSER
13	1	ULN2803 Darlington Array	511-ULN2803A	MOUSER
14	5	RED LED 3mm	754-1604-ND	DIGI-KEY
15	1	DC Motor	1528-1150-ND	DIGI-KEY
16	1	Servomotor	900-00005-ND	DIGI-KEY
17	1	Stepper Motor	1528-1367-ND	DIGI-KEY
18	2	4N25 Optocoupler	78-4N25	MOUSER
19	1	5V Relay	653-G5LE-14-DC5	MOUSER
20	1	1k Ohm Resistor	660-MF1/4LCT52R102J	MOUSER
21	1	4.7K Ohm Resistor	660-MF1/4LCT52R472J	MOUSER
22	2	22 Ohm Resistor	660-MF1/4LCT52R220J	MOUSER
23	2	330 Ohm Resistor	660-MF1/4LCT52R331J	MOUSER
24	5	2.2k Ohm Resistor	660-MF1/4LCT52R222J	MOUSER
25	1	2N3904 NPN Transistor	512-2N3904BU	MOUSER
26	2	MPSA42 NPN Transistor	512-MPSA42	MOUSER
27	2	MPSA92 PNP Transistor	610-MPSA92	MOUSER

4.2 Description

The motor interface module is composed of 4 main blocks. It provides access to standard motor drivers such as H-bridges, relays, and stepper motor controller. Figure 4.1 shows the block diagram of the motor interface module and its four blocks (H-bridge transistor, IC motor driver, relay, and stepper driver). This module is intended to work with either 3.3V and 5V microcontroller and it is also designed to allow direct interfacing to the MCU with no intermediate components.

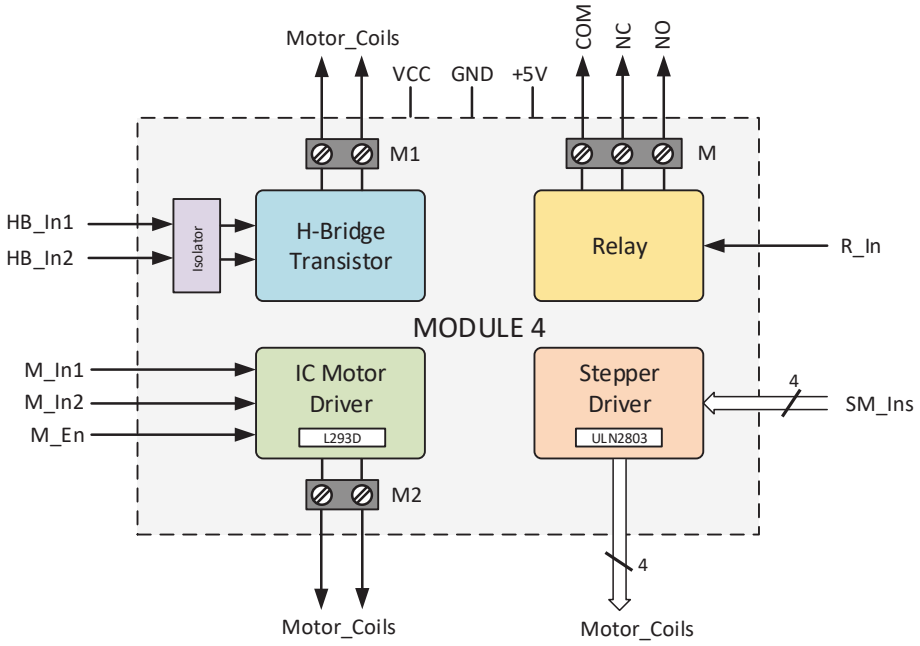


Figure 4.1: Module 4 Block diagram

The schematic of the module is illustrated in Figure 4.7 where all the connection between the different electronics components are depicted. In addition the board layout divided into components layer, top layer, and bottom layer, can be observed in Figure 4.8. Figure 4.9, and Figure 4.10 respectively. Finally, a real board representation of the EM is presented in Figure 4.11. This representation shows the real board to be used in the development of the laboratory experiments.

4.2.1 Power Supply Setup

To setup the different voltages needed for the proper operation of the system, you must have into account the operation voltage of your MCU, motors, and IC drivers.

Although some components in the modules were chosen to work with both 3.3V and 5V microcontrollers, some of them require a 5V power supply. In this module, the most quantity of connections are data signals and motor operating voltages. If your MCU works at 3.3V or 5V, you must connect all the power supply pins labeled on the board: 5V and GND (connect the voltages only when they are required). The power supply pins are highlighted in Figure 4.2. Remember to connect the GND, of your MCU, to the GND in the module to establish the same reference voltage for the entire system.

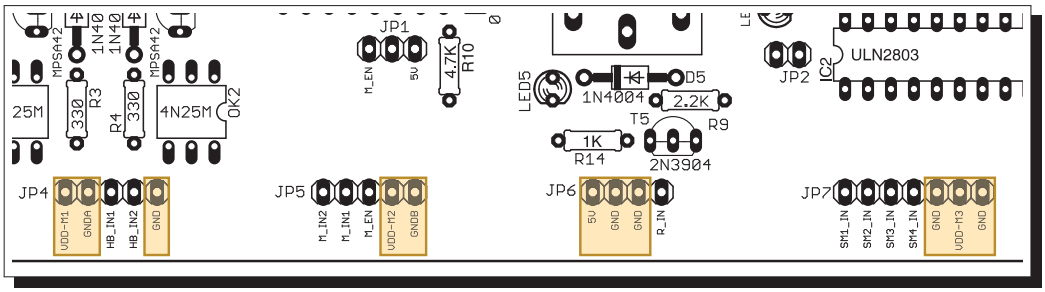


Figure 4.2: Module 4 Power supply pins

4.2.2 H-Bridge Transistor

This block is provide with 2 MPS42 NPN transistors, 2 MPS92 PNP transistors, 4 1N4004 Diodes, 2 4N25 optocouplers, 2 330Ω resistors, 2 22Ω resistors, and a 500mA fuse as depicted in Figure 4.3.

To use this block you need to connect all the voltages requires and insert the fuse into the fuse holder. The motor voltage needed will depend on of the specification of you DC motor, and it will be connected in the pins VDD-M1 and GNDA in pinheader JP4. Remember do not use a dc motor that could exceed the transistor's maximum current. The control signals that comes from your MCU must be connected to the pins HB_IN1 and HB_IN2 in pinheader JP4. Also a ground connection between your MCU and the block need to be carried out through the GND pin in the same pinheader. The control inputs in this block are isolated to prevent noise problems.

4.2.3 IC Motor Driver

This block provides access to a L293D dc motor driver with 4.7 KΩ resistor as depicted in Figure 4.4.

To use this block you need to connect all the controls signals, determine if you want

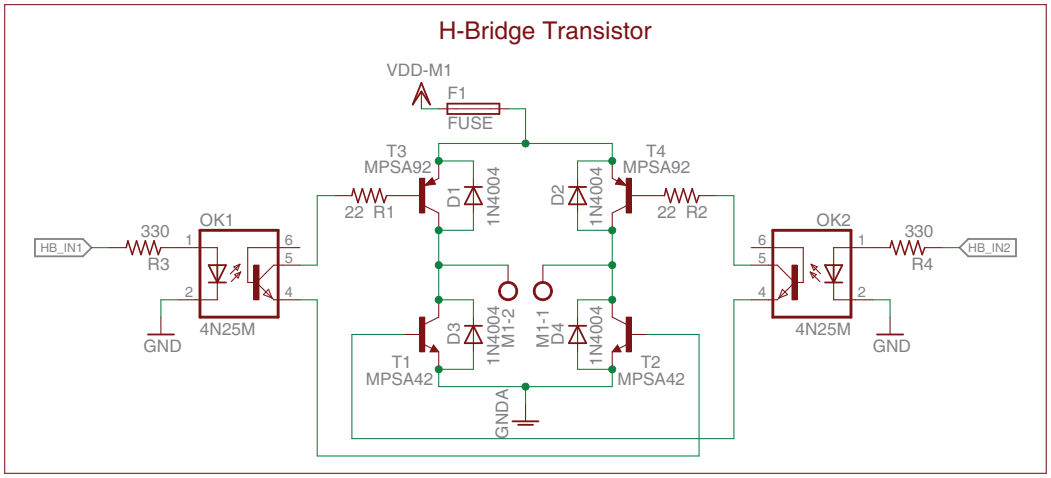


Figure 4.3: H-Bridge Transistor Schematic

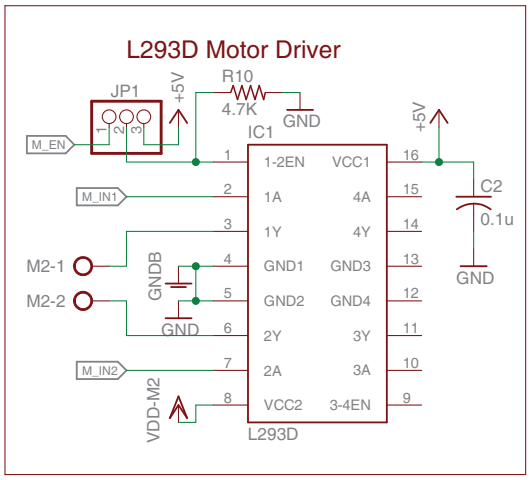


Figure 4.4: IC Motor Driver Schematic

to control the driver enable pin, and connect the power supply to the module. Follow the steps outlined below to setup this block:

1. To setup the power supply, you need to connect the motor voltage in the pins VDD-M2 and GNDB in pinheader JP5, and connect the IC power supply in the pins 5V and GND in pinheader JP6.

2. To define the operation of the enable pin you need to insert a jumper between 2 of the three pins on the pinheader JP1. If you put the jumper between the

middle pin and the 5V pin, the motor driver will permanently be enabled and whatever change presented in the inputs will be reflected on the outputs. But, if you insert the jumper between the middle pin and the M_En pin, it will provide you control to the enable or disable the operation of the motor driver. It will allow you to determine when the inputs will affect the outputs.

3. To control the rotation direction of the DC motor, you need to connect your MCU to the M_IN1 and M_IN2 pins in pinheader JP5 and sends the appropriate sequence of signals.

4.2.4 Relay

This block is provided with a 5V coil relay, 1 1N4004 diode, 1 1KΩ resistor, 1 2.2KΩ resistor, 1 2N3904 NPN transistor, and a 3mm RED LED as depicted in Figure 4.5.

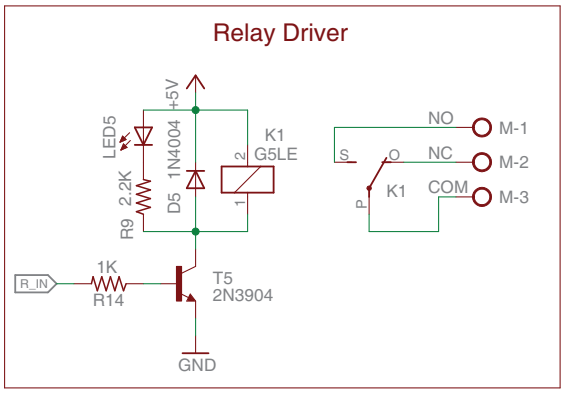


Figure 4.5: Relay Schematic

To use this block, you need to connect the power supply in the pins 5V and GND in the pinheader JP6. Also, you need to connect the control signal that comes from your MCU in the pin R_IN in pinheader JP6. This signal will turn On or Off the relay. The turn On/Off process is carried out by a 2N3904 NPN transistor and a 1KΩ resistor. This interface allows to turn-On the relay with a logic 1 and turn it Off when receives a logic 0. The 3mm LED will be On when the relay is activated.

4.2.5 Stepper Driver

This block is provided with a Darlington transistor array UNL2803, 4 3mm LEDs, and 4 2.2KΩ resistors as depicted in Figure 4.6.

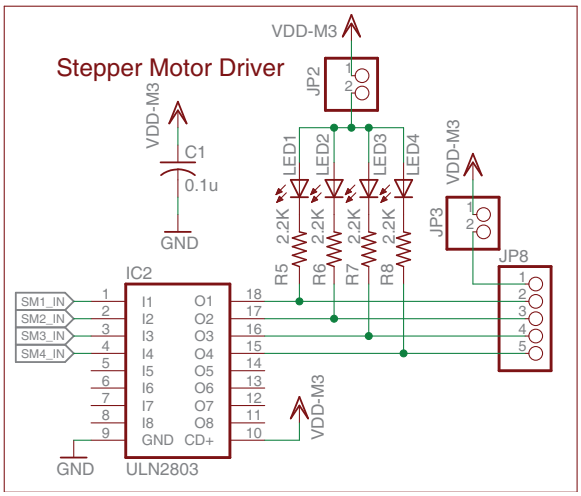


Figure 4.6: Stepper Driver block Schematic

To use this block, you need to connect the motor voltage in the pins VDD-M3 and GND in the pinheader JP7. The motor voltage required will depend on of your stepper motor specifications. To control the stepper motor, you need to connect all the control signals in pins SM1_IN, SM2_IN, SM3_IN, and SM4_IN in pinheader JP7. If you want to enable the turn On/Off process of the LEDs, you need to put a jumper in pinheader JP2. To enable the stepper motor, you need to insert a jumper in pinheader JP3. Each output of the driver will be in high impedance state while the inputs are receiving a logic 0 and they will be at ground when the inputs receive a logic 1.

4.3 Schematic

The Figure 4.7 shows the complete schematic of the module were all the connections are described.

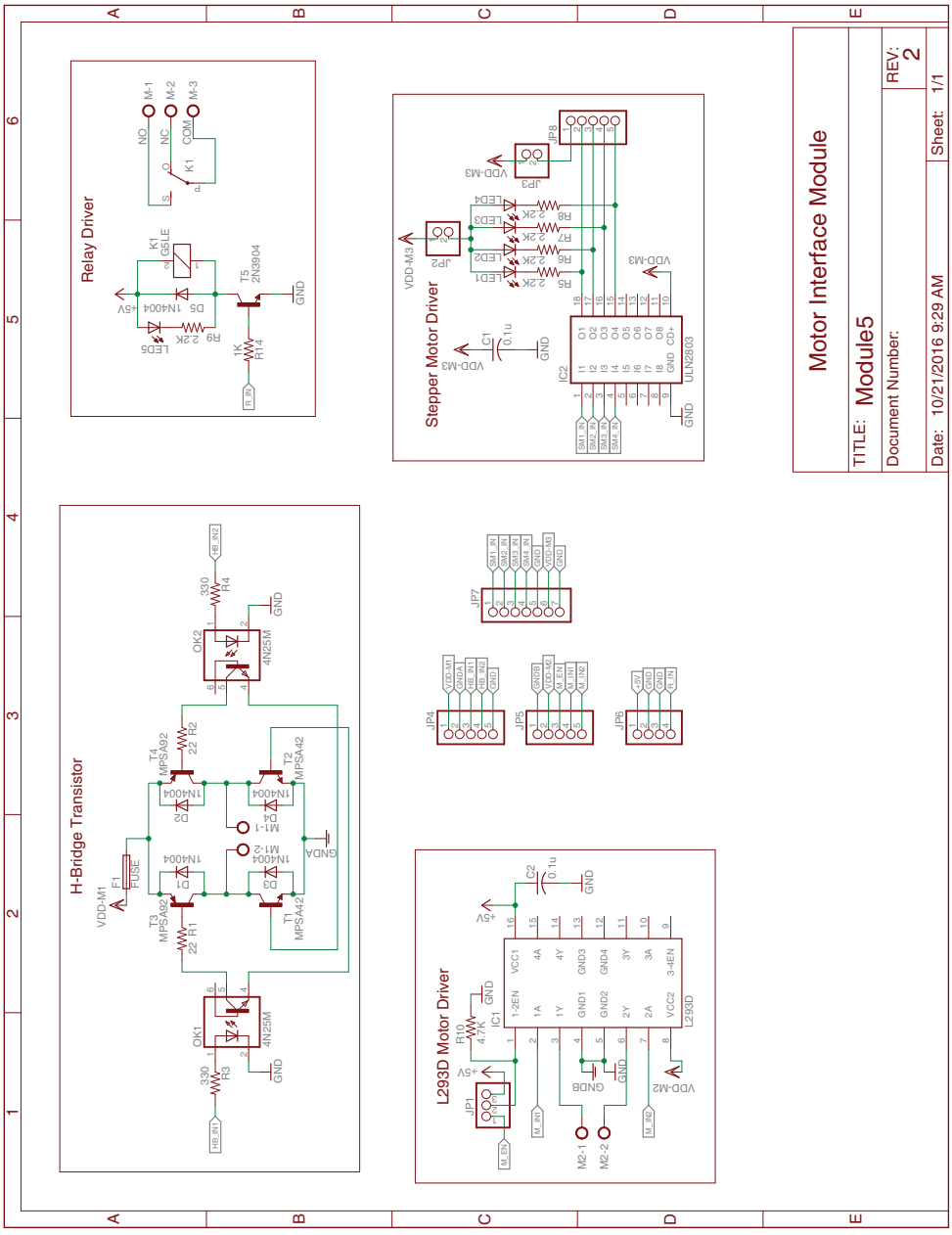


Figure 4.7: Module 4 Schematic

4.4 Board

The Figure 4.8 represent the component layer of the module. This Figure shows how the different elements are arranged on the PCB. The measures are in millimeters.

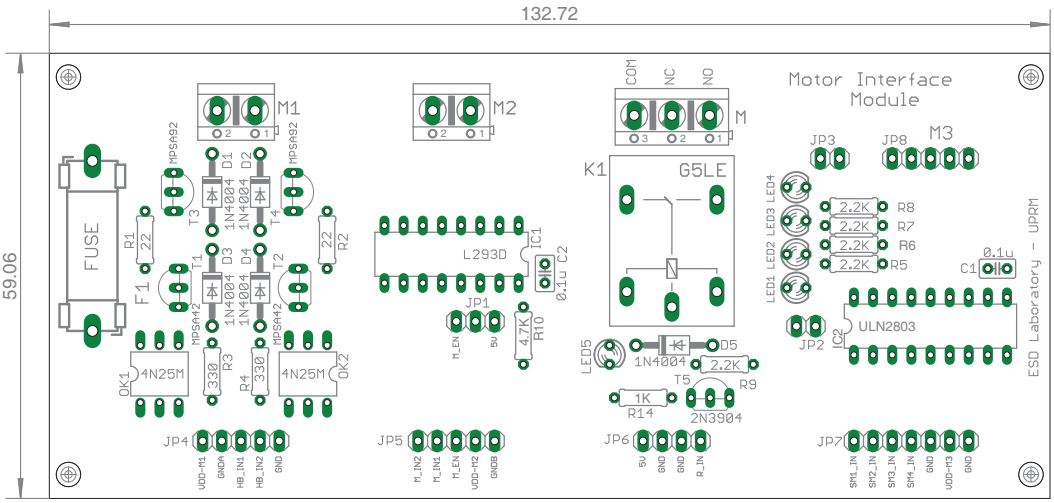


Figure 4.8: Module 4 board - components layer

The Figure 4.9 and Figure 4.10 represents the Top and Bottom layer of the module respectively. These images show the different physical connections between the different components in the module.

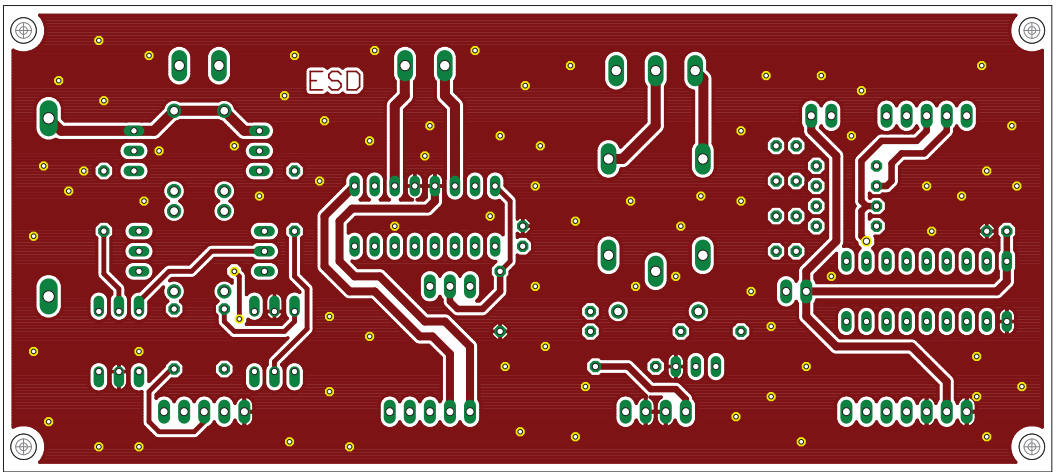


Figure 4.9: Module 4 board - top layer

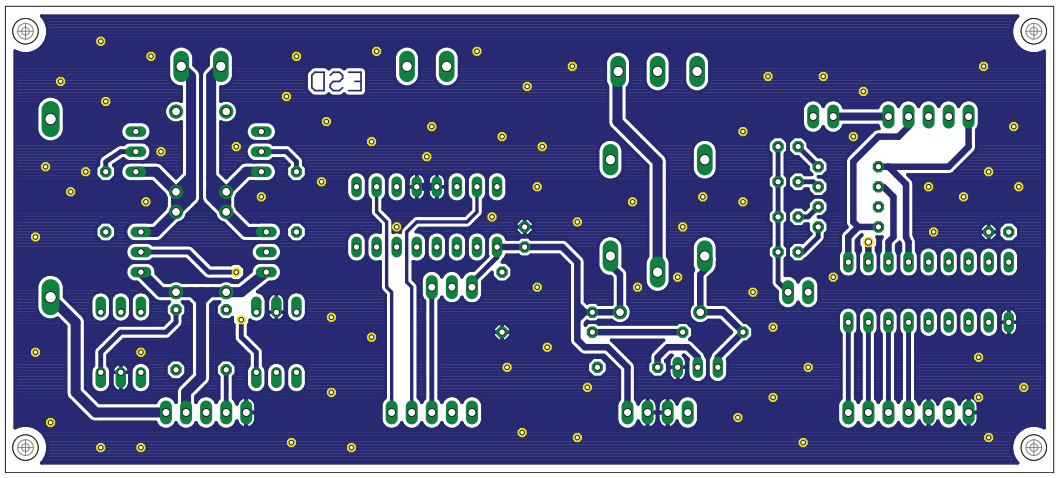


Figure 4.10: Module 4 board - bottom layer

The Figure 4.11 illustrate a real representation of the module's PCB.

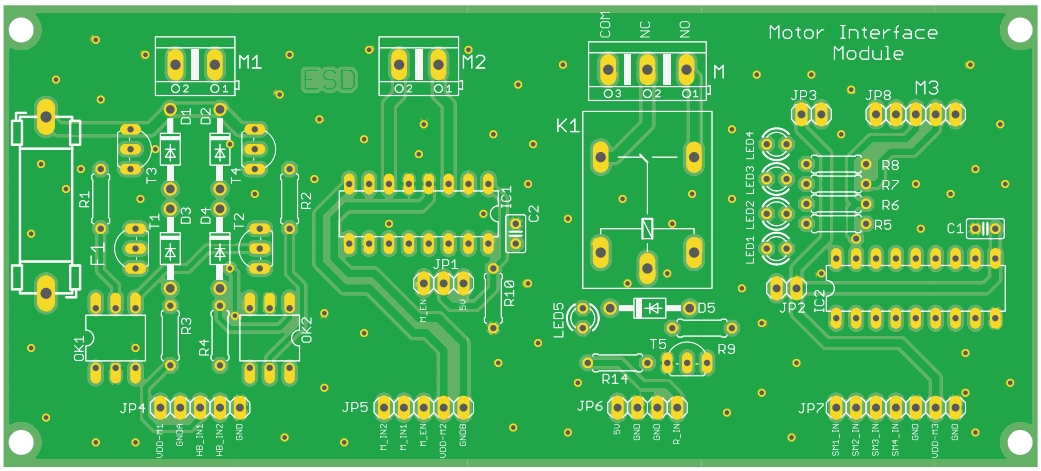


Figure 4.11: Module 4 board

Module 5

Serial Communications Module

5.1 Materials

The materials needed to assembly the Data Converters module is listed in Table 5.1.

Table 5.1: Boom of materials Module 5

Item	Qty	Description	P/N reference	Supplier
1	1	Jumper x5	970	POLOLU
2	1	3V Battery cr2032	P189-ND	DIGI-KEY
3	1	0.1uF Tantalum Cap.	581-TAP104K035SCS	MOUSER
4	3	0.47uF Capacitor	647-UHE1HR47MDD	MOUSER
5	1	0.1uF Capacitor	647-UVR1H0R1MDD	MOUSER
6	1	Female DB9 Connector	AE10921-ND	DIGI-KEY
7	1	1x40 Male Header	965	POLOLU
8	1	Battery Holder	BS-3-ND	DIGI-KEY
9	1	Dip16 IC base	AE9992-ND	DIGI-KEY
10	1	Dip8 IC base	AE9986-ND	DIGI-KEY
11	1	32.768KHz Crystal	732-C004R32.76K-APB	MOUSER
12	1	Real-Time Clock DS1307	700-DS1307	MOUSER
13	1	MAX3232 RS-232 Transceiver	700-MAX3232CPE	MOUSER
14	2	RED LED 3mm	754-1604-ND	DIGI-KEY
15	2	1k Ohm Resistor	660-MF1/4LCT52R102J	MOUSER
16	3	4.7K Ohm Resistor	660-MF1/4LCT52R472J	MOUSER

5.2 Description

The serial communications module is composed of 2 main blocks. It provides access to devices that use serial communication protocols such as RS232 and I^2C. Figure 5.1 shows the block diagram of the serial communication module and its two blocks (Serial-to-RS232 converter and Real-Time clock). This module is intended to work with either 3.3V and 5V microcontrollers and it is also designed to allow direct interfacing to the MCU with no intermediate components.

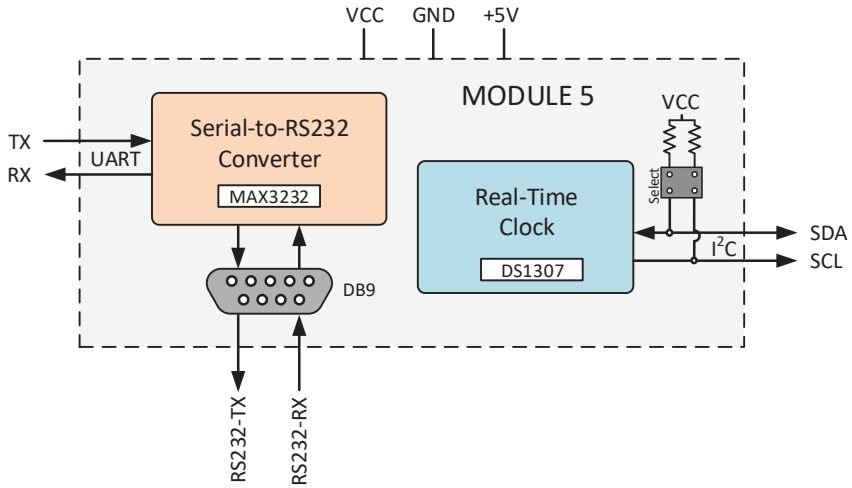


Figure 5.1: Module 5 Block Diagram

The schematic of the module is illustrated in Figure 5.5 where all the connection between the different electronics components are depicted. In addition the board layout divided into components layer, top layer, and bottom layer, can be observed in Figure 5.6. Figure 5.7, and Figure 5.8 respectively. Finally, a real board representation of the EM is presented in Figure 5.9. This representation shows the real board to be used in the development of the laboratory experiments.

5.2.1 Power Supply Setup

To setup the different voltages needed for the proper operation of the system, you must take into account the operation voltage of your MCU. Although some components in the module were chosen to work with both 3.3V and 5V microcontrollers, some of them require a 5V power supply. If your MCU works at 3.3V, you must connect all the power supply pins labeled on the board: 5V, VCC, and GND (using 3.3V for VCC). Also, the jumper in pinheader JP3 must be removed. But, if you are using a 5V microcontroller, you only need to connect one of the two power supply pins (5V or VCC) and put a jumper in pinheader JP3. The power supply pins to be used are highlighted in Figure 5.2. Remember to connect the GND, of your MCU, to the GND in the module to establish the same reference voltage for the entire system.

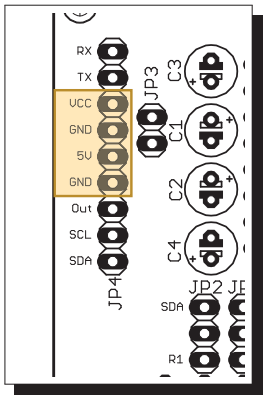


Figure 5.2: Module 5 Power supply pins

5.2.2 Serial-To-RS232 Converter

This block is provided with a MAX3232, 1 0.47uF electrolytic capacitor, 3 0.1uF electrolytic capacitor, 2 1KΩ resistor, 2 3mm LEDs, and a DB9 connector as depicted in Figure 5.3.

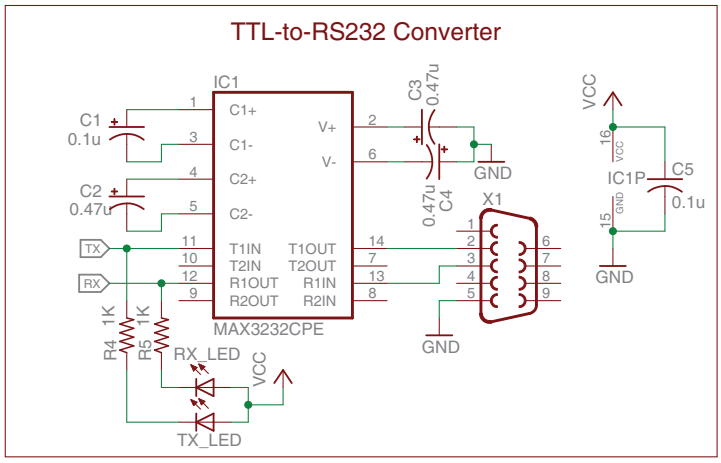


Figure 5.3: Serial-To-RS232 Converter Schematic

To use this block, you need to connect the RX and TX pins of your MCU to the RX and TX pins in pinheader JP4. Also, you need to plug in an RS232-to-USB cable converter in the DB9 connector X1 to start the communication process between your MCU and the PC. Internally, the driver (MAX3232) do the respective conversions between the TTL logic levels and RS232 logic levels.

5.2.3 Real-Time Clock

This block provides access to a real-time clock DS1307 with a 32.768 KHz crystal, three 3KΩ pull-up resistors, and a 3V lithium battery backup as depicted in Figure 5.4.

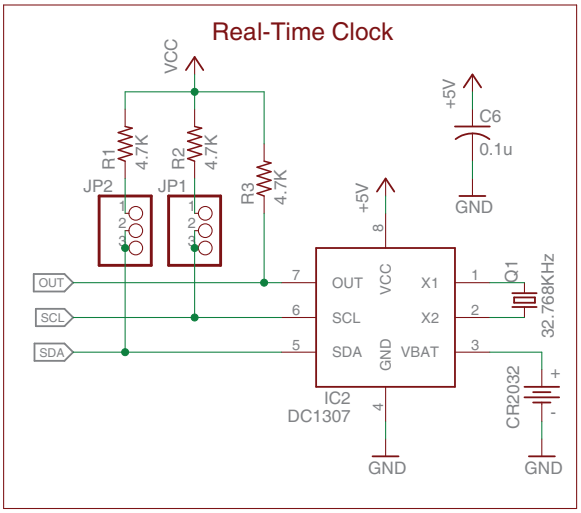


Figure 5.4: Serial-To-RS232 Converter Schematic

To use this block you need to connect the SDA and SCL pins of your MCU to the module pins, determine the use or not of the pull-up resistors, and configure DS1307's OUT signal. Follow the steps outline below to setup this block:

1. The SCL and SDA pins of your MCU needs to be connected to SCL and SDA pins in pinheader JP4 of the module. These pins will be used for establishing the I^2 communication between the devices. The OUT signal in the block is an extra feature that can be activated or not on the DS1307 by configuring its internal registers.

2. To setup the pull-up resistor for the I^2C bus, you need to put jumpers in pinheaders JP1 and JP2. If you want to use the pull-up resistor, for SDA line, you need to insert a jumper between the middle pin and the R1 pin in pinheader JP2. But, if you desired not use the pull-up resistor, you need to insert the jumper between the middle pin and the SDA pin in pinheader JP2 and provide an external pull-up resistor attached to the operation voltage of your MCU. To setup the another pull-up resistor you have to perform the same procedure outlined to setup the SDA line.

5.3 Schematic

The Figure 5.5 shows the complete schematic of the module were all the connections are described.

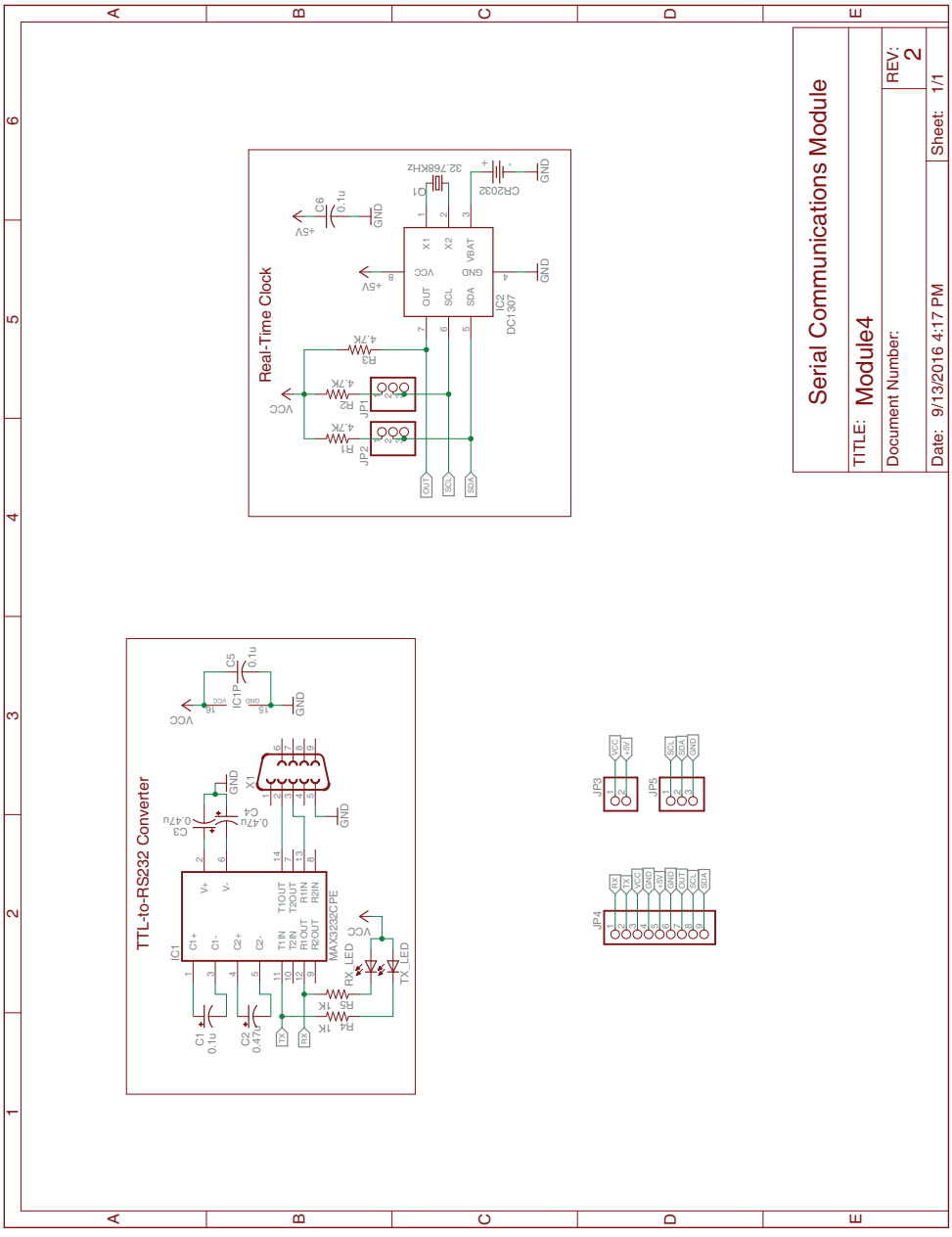


Figure 5.5: Module 5 Schematic

5.4 Board

The Figure 5.6 represent the component layer of the module. This Figure shows how the different elements are arranged on the PCB. The measures are in millimeters.

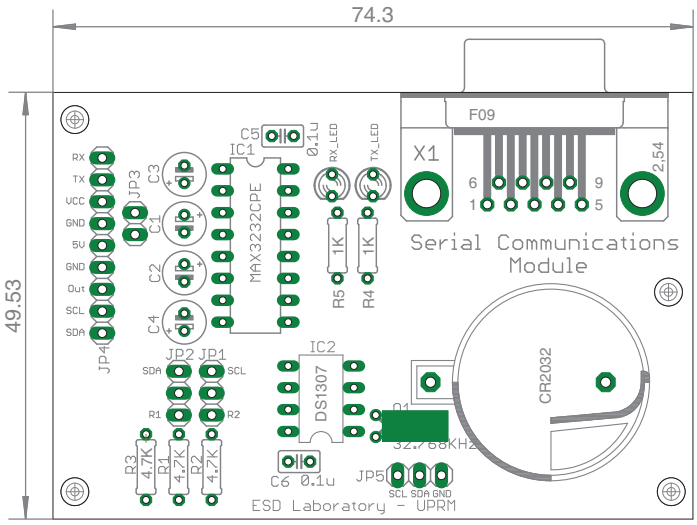


Figure 5.6: Module 5 board - components layer

The Figure 5.7 and Figure 5.8 represents the Top and Bottom layer of the module respectively. These images show the different physical connections between the different components in the module.

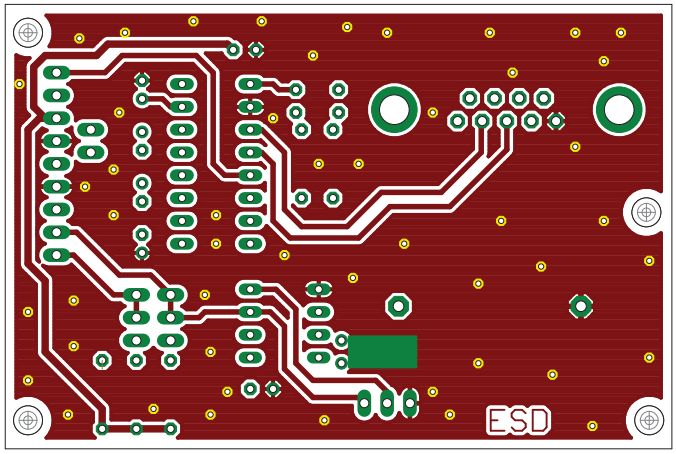


Figure 5.7: Module 5 board - top layer

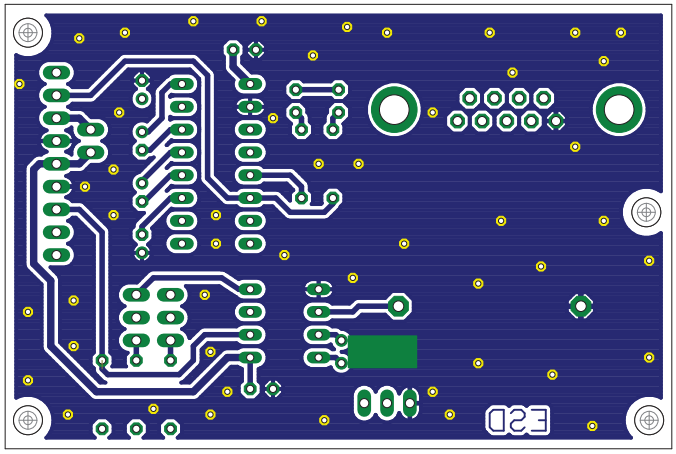


Figure 5.8: Module 5 board - bottom layer

The Figure 5.9 illustrate a real representation of the module's PCB.

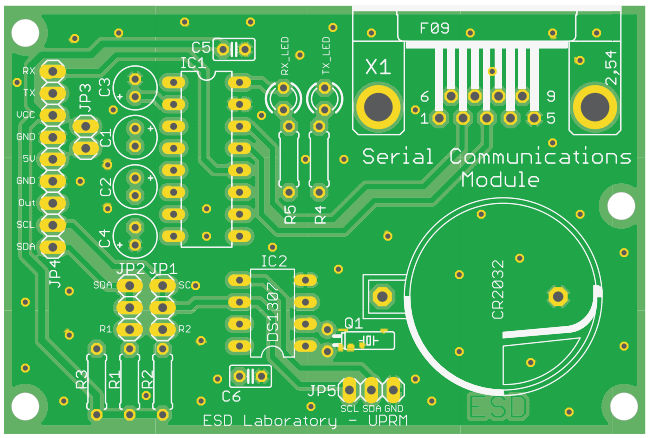


Figure 5.9: Module 5 board

Module 6

Data Converters Module

6.1 Materials

The materials needed to assembly the Data Converters module is listed in Table 6.1.

Table 6.1: Boom of materials Module 6

Item	Qty	Description	P/N reference	Supplier
1	1	0.1uF Tantalum	581-TAP104K035SC	MOUSER
2	1	0.1uF Ceramic	594-D101K20Y5PL63L6R	MOUSER
3	1	1x40 Male Header	965	POLOLU
4	1	2x40 Male Header	966	POLOLU
5	1	Dip16	AE9992-ND	DIGI-KEY
6	1	Dip8	AE9986-ND	DIGI-KEY
7	1	DAC0808	926-DAC0808LCN/NOPB	MOUSER
8	1	LM35	926-LM35DZ/NOPB	MOUSER
9	1	LM358	595-LM358P	MOUSER
10	1	10k Ohm pot	688-RK09K1130AH1	MOUSER
11	1	10k Ohm Prec. Pot	72-T18-10K	MOUSER
12	4	2.4k Ohm	660-MF1/4LCT52R242J	MOUSER

6.2 Description

The data converters module is composed of 3 main blocks. It provides access to common analog devices and digital-to-analog converters. Figure 6.1 shows the block diagram of the data converter module and its three blocks (Digital-To-Analog Converter, Temperature Sensor, and Potentiometers). This module is intended to work with either 3.3V and 5V microcontroller and it is also designed to allow direct interfacing to the MCU with no intermediate components.

The schematic of the module is illustrated in Figure 6.6 where all the connection between the different electronics components are depicted. In addition, the board layout divided into components layer, top layer, and bottom layer, can be observed

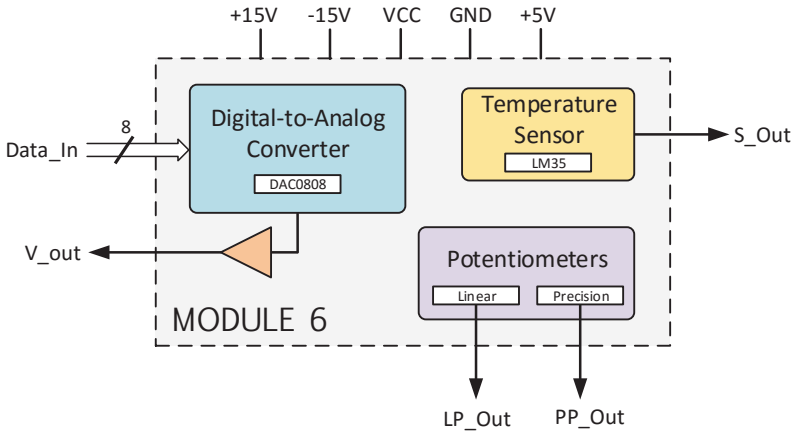


Figure 6.1: Module 6 Block diagram

in Figure 6.7. Figure 6.8, and Figure 6.9 respectively. Finally, a real board representation of the EM is presented in Figure 6.10. This representation shows the real board to be used in the development of the laboratory experiments.

6.2.1 Power Supply Setup

To setup the different voltages needed for the proper operation of the system, you must take into account the operation voltage of your MCU. Although some components in the module were chosen to work with both 3.3V and 5V microcontrollers, some of them require a 5V power supply. If your MCU works at 3.3V, you must connect all the power supply pins labeled on the board: 5V, VCC, and GND (using 3.3V for VCC). Also, the jumper in pinheader JP6 must be removed. But, if you are using a 5V microcontroller, you only need to connect one of the two power supply pins (5V or VCC), and put a jumper in pinheader JP3. The power supply pins to are highlighted in Figure 6.2. Remember to connect the GND, of your MCU, to the GND in the module to establish the same reference voltage for the entire system.

6.2.2 Digital-To-Analog Converter

This block is provided with a DAC0808 digital-to-analog converter, 1 LM358 operational amplifier, 4 2.4KΩ resistors, and a 0.1nF ceramic capacitor as depicted in Figure 6.3.

To use this block you need to connect all the voltages requires by the ICs (+15V, -15V, 5V, and VCC) in pinheader JP2 and JP1. Also, the data signals that comes

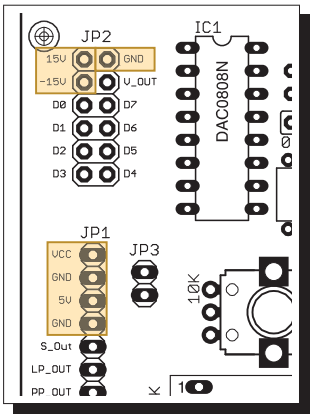


Figure 6.2: Module 6 Power supply pins

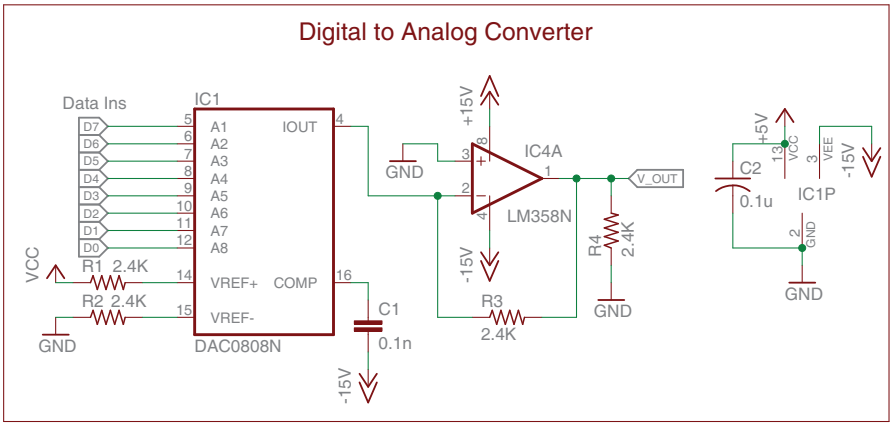


Figure 6.3: Digital-To-Analog Converter Schematic

from your MCU must be connected to D0 to D7 pins in pinheader JP2. The DAC0808 is a data converter of 8 bits that uses the D0 for the LSB and D7 for the MSB. The output voltage could be measured in the V_OUT pin. The reference voltage for the conversion could be the same of your MCU operation voltage but if you need to select another voltage, connect the voltage to the VCC pin and remove the jumper in pinheader JP3. Be sure that the reference voltage needed do not exceed 10V DC.

6.2.3 Temperature Sensor

This block provides access to an LM35 analog temperature sensor as depicted in Figure 6.4.

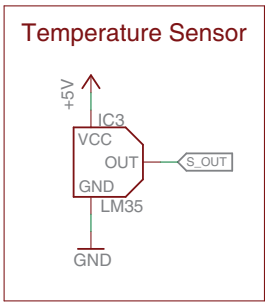


Figure 6.4: Temperature Sensor Schematic

To use this block you only need to connect the power supply to the 5V pin in pinheader JP1 and measure the analog output voltage in the pin S_Out. The sensor has a conversion factor of 10mV for each 1°C of temperature.

6.2.4 Potentiometers

This block provides access to a single turn potentiometer R5 (Linear) and an multi-turn potentiometer R6 (Precision) as depicted in Figure 6.5.

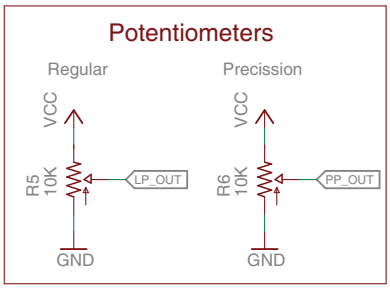


Figure 6.5: Potentiometers Schematic

To use use this block you only need to connect the power supply to the VCC pin in pinheader JP1 and measure the desired potentiometer output in LP_OUT for the R5 potentiometer or PP_OUT for the R6 potentiometer. Be sure to use the same operation voltage of your MCU or a voltage that is acceptable for the ADC module of your MCU. Any other voltage could potentially damage your MCU.

6.3 Schematic

The Figure 6.6 shows the complete schematic of the module were all the connections are described.

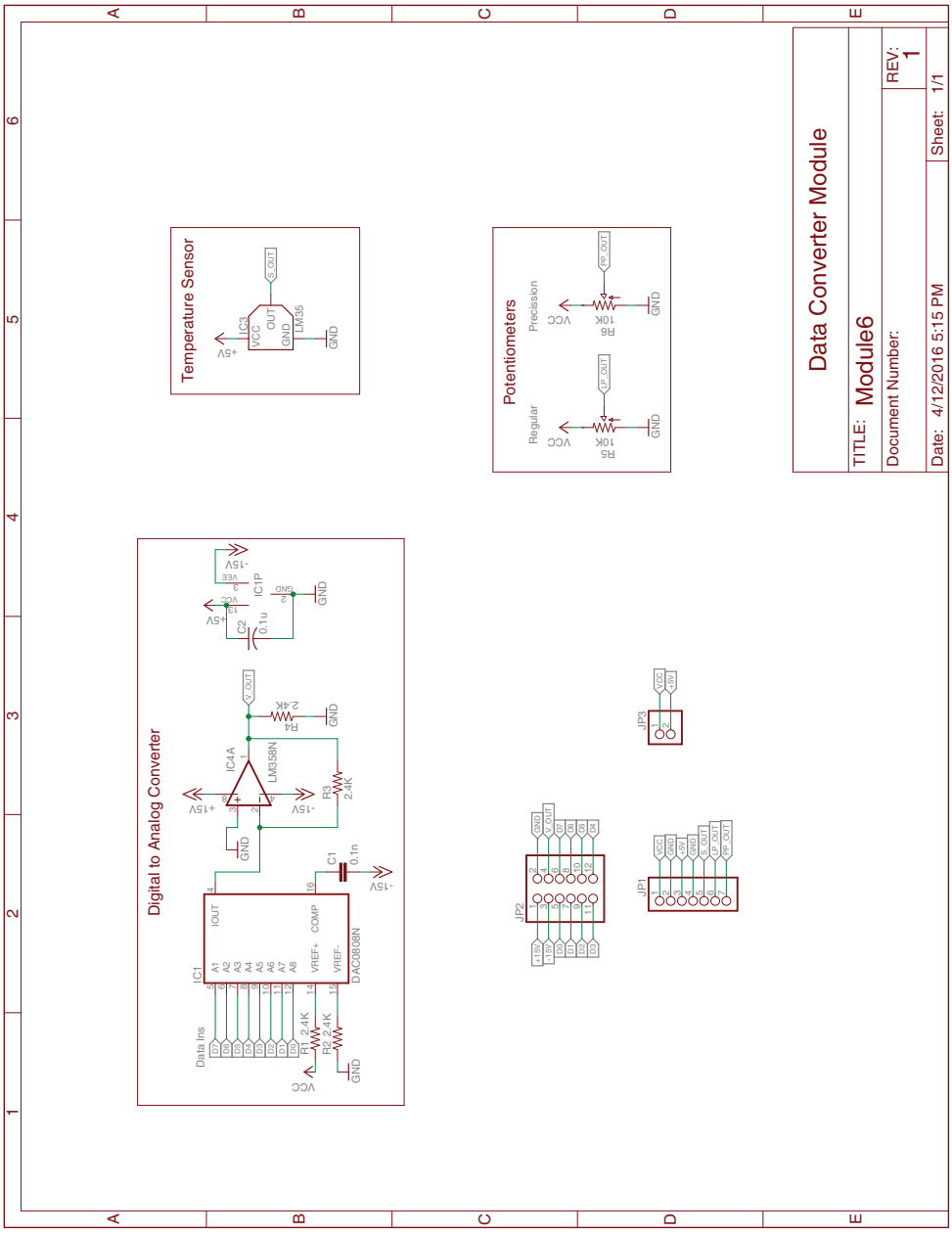


Figure 6.6: Module 6 Schematic

6.4 Board

The Figure 6.7 represent the component layer of the module. This Figure shows how the different elements are arranged on the PCB. The measures are in millimeters.

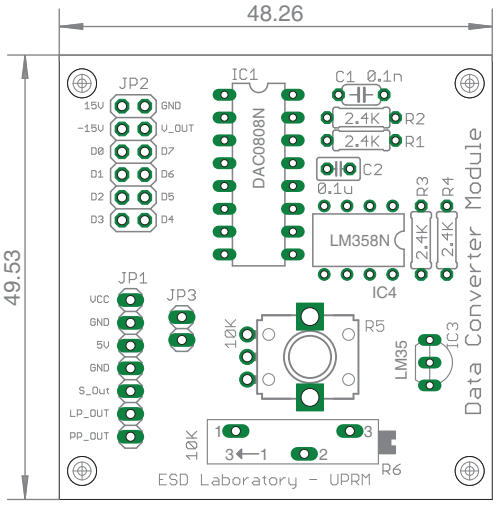


Figure 6.7: Module 6 board - components layer

The Figure 6.8 and Figure 6.9 represents the Top and Bottom layer of the module respectively. These images show the different physical connections between the different components in the module.

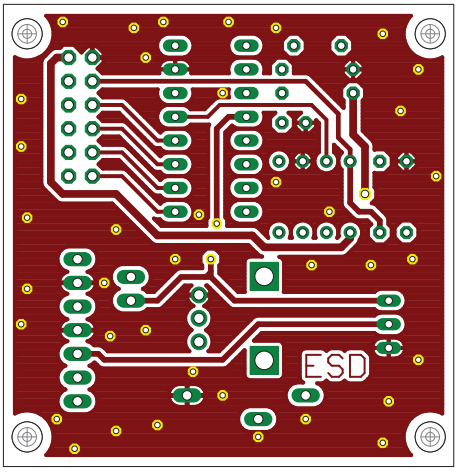


Figure 6.8: Module 6 board - top layer

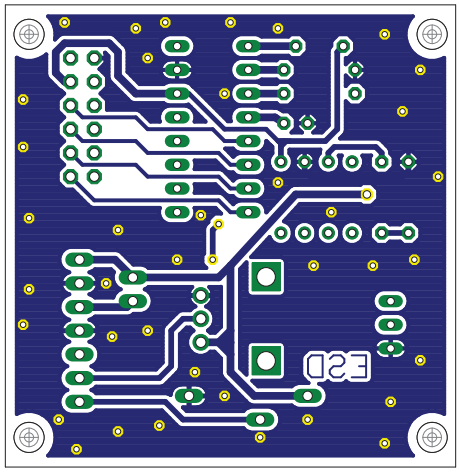


Figure 6.9: Module 6 board - bottom layer

The Figure 6.10 illustrate a real representation of the module's PCB.

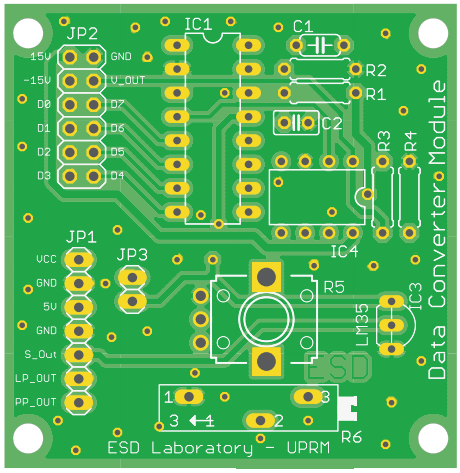


Figure 6.10: Module 6 board

Bibliography

[1] Hd44780u (lcd-ii). HITACHI. https://www.sparkfun.com/datasheets/LCD/HD44780.pdf. Accessed: 2016-03-25.

[2] Lcd 16x2 (wh1602b2-tm1-et#). http://www.mouser.com/ds/2/272/-364177.pdf. Accessed: 2016-03-25.

[3] Ps1240p02ct3. TDK. https://product.tdk.com/info/en/catalog/datasheets/ef532_ps.pdf. Accessed: 2016-03-25.

[4] Dc56-11ewa. KINGBRIGHT. http://www.us.kingbright.com/images/catalog/spec/DC56-11EWA.pdf. Accessed: 2016-03-25.

[5] Wp154a4sureqbfzw. KINGBRIGHT. https://www.kingbrightusa.com/images/catalog/spec/WP154A4SUREQBFZGW.pdf. Accessed: 2016-03-25.

[6] Rpr-220. ROHM. http://rohmfs.rohm.com/en/products/databook/datasheet/opto/optical_sensor/photosensor/rpr-220.pdf. Accessed: 2016-03-25.

[7] Ds1307. MAXIM INTEGRATED. http://datasheets.maximintegrated.com/en/ds/DS1307.pdf. Accessed: 2016-03-25.

[8] Max3232. MAXIM INTEGRATED. http://pdfserv.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf. Accessed: 2016-03-25.

[9] Parallax standard servo (#900-00005). PARALLAX. https://www.parallax.com/sites/default/files/downloads/900-00005-Standard-Servo-Product-Documentation-v2.2.pdf. Accessed: 2016-03-25.

[10] L293d. TEXAS INSTRUMENTS. http://www.mouser.pr/ProductDetail/Texas-Instruments/L293DNE/?qs=sGAEpiMZZMtYFXwiBRPs0wSafWlCmJbc. Accessed: 2016-03-25.

[11] Dac0808. TEXAS INSTRUMENTS. http://www.ti.com/lit/ds/symlink/dac0808.pdf. Accessed: 2016-03-25.

[12] Lm35. TEXAS INSTRUMENTS. http://www.ti.com/lit/ds/symlink/lm35.pdf. Accessed: 2016-03-25.

Appendix E
Students Grades

E.1 Control Group Student Grades

Table E.1 : Control group pre-tests grades

	Pre-Test						
Student	**1**	**2**	**3**	**4**	**5**	**6**	**μ Student**
N1	15.0	11.5	12.5	5.0	27.5	15.0	14.4
N2	12.5	30.5	20.0	30.0	7.5	0.0	16.8
N3	5.0	12.5	15.0	15.0	15.0	2.5	10.8
N4	12.5	10.0	15.0	24.0	10.0	5.0	12.8
N5	12.5	15.0	17.5	15.0	10.0	7.5	12.9
N6	5.0	12.0	15.0	12.0	17.5	15.0	12.8
N7	15.0	17.5	12.5	20.0	22.5	27.5	19.2
N8	12.5	12.5	0.0	17.5	10.0	5.0	9.6
N9	10.0	5.0	25.0	22.5	10.0	15.0	14.6
N10	32.0	32.5	32.0	17.5	20.0	15.0	24.8
N11	15.0	33.0	25.0	20.0	27.5	12.5	22.2
N12	0.0	33.5	20.0	20.0	10.0	17.5	16.8
N13	10.0	38.0	17.5	32.5	22.5	5.0	20.9
N14	0.0	35.0	17.0	10.0	10.0	5.0	12.8
N15	0.0	28.0	14.5	5.0	25.0	7.5	13.3
N16	20.0	37.5	25.0	37.5	17.5	5.0	23.8
μ Test	11.1	22.8	17.7	19.0	16.4	10.0	16.2
σ	8.2	11.6	7.2	9.1	7.0	7.1	

Note: The maximum grade achievable is 40 pts

Table E.2 : Control group post-tests grades

	Post-Test						
Student	**1**	**2**	**3**	**4**	**5**	**6**	**μ Student**
N1	32.5	30.0	22.5	10.0	10.0	7.5	18.8
N2	30.0	37.5	17.5	22.5	0.0	30.0	22.9
N3	40.0	20.0	22.5	15.0	22.5	5.0	20.8
N4	40.0	30.5	25.0	32.5	17.5	30.0	29.3
N5	35.0	23.5	20.0	35.0	20.0	7.5	23.5
N6	22.5	18.0	15.0	7.5	17.5	12.5	15.5
N7	35.0	28.0	15.0	25.0	20.0	32.5	25.9
N8	25.0	22.5	7.5	32.5	12.5	22.5	20.4
N9	37.5	2.5	32.5	30.0	15.0	30.0	24.6
N10	31.0	40.0	35.0	10.0	25.0	22.5	27.3
N11	5.0	40.0	20.0	25.0	30.0	20.0	23.3
N12	27.0	30.0	22.5	22.5	12.5	17.5	22.0
N13	11.0	31.0	27.5	35.0	20.0	15.0	23.3
N14	3.0	33.0	10.0	17.5	15.0	17.5	16.0
N15	11.0	23.5	25.0	12.5	15.0	27.5	19.1
N16	9.0	38.0	25.0	27.5	30.0	40.0	28.3
μ Test	24.7	28.0	21.4	22.5	17.7	21.1	22.6
σ	12.8	9.7	7.4	9.4	7.5	10.1	

Note: The maximum grade achievable is 40 pts

Table E.3 : Control group pre-tests normalized grades

	Pre-Test (%)						
Student	**1**	**2**	**3**	**4**	**5**	**6**	**μ Student**
N1	37.50	28.75	31.25	12.50	68.75	37.50	36.0
N2	31.25	76.25	50.00	75.00	18.75	0.00	41.9
N3	12.50	31.25	37.50	37.50	37.50	6.25	27.1
N4	31.25	25.00	37.50	60.00	25.00	12.50	31.9
N5	31.25	37.50	43.75	37.50	25.00	18.75	32.3
N6	12.50	30.00	37.50	30.00	43.75	37.50	31.9
N7	37.50	43.75	31.25	50.00	56.25	68.75	47.9
N8	31.25	31.25	0.00	43.75	25.00	12.50	24.0
N9	25.00	12.50	62.50	56.25	25.00	37.50	36.5
N10	80.00	81.25	80.00	43.75	50.00	37.50	62.1
N11	37.50	82.50	62.50	50.00	68.75	31.25	55.4
N12	0.00	83.75	50.00	50.00	25.00	43.75	42.1
N13	25.00	95.00	43.75	81.25	56.25	12.50	52.3
N14	0.00	87.50	42.50	25.00	25.00	12.50	32.1
N15	0.00	70.00	36.25	12.50	62.50	18.75	33.3
N16	50.00	93.75	62.50	93.75	43.75	12.50	59.4
μ Test	27.7	56.9	44.3	47.4	41.0	25.0	40.4
σ	20.57	29.02	17.97	22.77	17.53	17.82	

Note: The student grades were normalized to be presented in percentages

Table E.4 : Control group normalized post-tests grades

	Post-Test (%)						
Student	**1**	**2**	**3**	**4**	**5**	**6**	**μ Student**
N1	81.25	75.00	56.25	25.00	25.00	18.75	46.9
N2	75.00	93.75	43.75	56.25	0.00	75.00	57.3
N3	100.00	50.00	56.25	37.50	56.25	12.50	52.1
N4	100.00	76.25	62.50	81.25	43.75	75.00	73.1
N5	87.50	58.75	50.00	87.50	50.00	18.75	58.8
N6	56.25	45.00	37.50	18.75	43.75	31.25	38.8
N7	87.50	70.00	37.50	62.50	50.00	81.25	64.8
N8	62.50	56.25	18.75	81.25	31.25	56.25	51.0
N9	93.75	6.25	81.25	75.00	37.50	75.00	61.5
N10	77.50	100.00	87.50	25.00	62.50	56.25	68.1
N11	12.50	100.00	50.00	62.50	75.00	50.00	58.3
N12	67.50	75.00	56.25	56.25	31.25	43.75	55.0
N13	27.50	77.50	68.75	87.50	50.00	37.50	58.1
N14	7.50	82.50	25.00	43.75	37.50	43.75	40.0
N15	27.50	58.75	62.50	31.25	37.50	68.75	47.7
N16	22.50	95.00	62.50	68.75	75.00	100.00	70.6
μ Test	61.6	70.0	53.5	56.3	44.1	52.7	56.4
σ	32.04	24.25	18.40	23.50	18.75	25.31	

Note: The student grades were normalized to be presented in percentages

		Student (%)															
		N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**	**N9**	**N10**	**N11**	**N12**	**N13**	**N14**	**N15**	**N16**
Test 1	Pre	37.5	31.3	12.5	31.3	31.3	12.5	37.5	31.3	25.0	80.0	37.5	0.0	25.0	0.0	0.0	50.0
	Post	81.3	75.0	100.0	100.0	87.5	56.3	87.5	62.5	93.8	77.5	12.5	67.5	27.5	7.5	27.5	22.5
	G_i	43.8	43.8	87.5	68.8	56.3	43.8	50.0	31.3	68.8	-2.5	-25.0	67.5	2.5	7.5	27.5	-27.5
	$<g_i>$	70.0	63.6	100.0	100.0	81.8	50.0	80.0	45.5	91.7	0.0	0.0	67.5	3.3	7.5	27.5	0.0
Test 2	Pre	28.8	76.3	31.3	25.0	37.5	30.0	43.8	31.3	12.5	81.3	82.5	83.8	95.0	87.5	70.0	93.8
	Post	75.0	93.8	50.0	76.3	58.8	45.0	70.0	56.3	6.3	100.0	100.0	75.0	77.5	82.5	58.8	95.0
	G_i	46.3	17.5	18.8	51.3	21.3	15.0	26.3	25.0	-6.3	18.8	17.5	-8.8	-17.5	-5.0	-11.3	1.3
	$<g_i>$	64.9	73.7	27.3	68.3	34.0	21.4	46.7	36.4	0.0	100.0	100.0	0.0	0.0	0.0	0.0	20.0
Test 3	Pre	31.3	50.0	37.5	37.5	43.8	37.5	31.3	0.0	62.5	80.0	62.5	50.0	43.8	42.5	36.3	62.5
	Post	56.3	43.8	56.3	62.5	50.0	37.5	37.5	18.8	81.3	87.5	50.0	56.3	68.8	25.0	62.5	62.5
	G_i	25.0	-6.3	18.8	25.0	6.3	0.0	6.3	18.8	18.8	7.5	-12.5	6.3	25.0	-17.5	26.3	0.0
	$<g_i>$	36.4	0.0	30.0	40.0	11.1	0.0	9.1	18.8	50.0	37.5	0.0	12.5	44.4	0.0	41.2	0.0
Test 4	Pre	12.5	75.0	37.5	60.0	37.5	30.0	50.0	43.8	56.3	43.8	50.0	50.0	81.3	25.0	12.5	93.8
	Post	25.0	56.3	37.5	81.3	87.5	18.8	62.5	81.3	75.0	25.0	62.5	56.3	87.5	43.8	31.3	68.8
	G_i	12.5	-18.8	0.0	21.3	50.0	-11.3	12.5	37.5	18.8	-18.8	12.5	6.3	6.3	18.8	18.8	-25.0
	$<g_i>$	14.3	0.0	0.0	53.1	80.0	0.0	25.0	66.7	42.9	0.0	25.0	12.5	33.3	25.0	21.4	0.0
Test 5	Pre	68.8	18.8	37.5	25.0	25.0	43.8	56.3	25.0	25.0	50.0	68.8	25.0	56.3	25.0	62.5	43.8
	Post	25.0	0.0	56.3	43.8	50.0	43.8	50.0	31.3	37.5	62.5	75.0	31.3	50.0	37.5	37.5	75.0
	G_i	-43.8	-18.8	18.8	18.8	25.0	0.0	-6.3	6.3	12.5	12.5	6.3	6.3	-6.3	12.5	-25.0	31.3
	$<g_i>$	0.0	0.0	30.0	25.0	33.3	0.0	0.0	8.3	16.7	25.0	20.0	8.3	0.0	16.7	0.0	55.6
Test 6	Pre	37.5	0.0	6.3	12.5	18.8	37.5	68.8	12.5	37.5	37.5	31.3	43.8	12.5	12.5	18.8	12.5
	Post	18.8	75.0	12.5	75.0	18.8	31.3	81.3	56.3	75.0	56.3	50.0	43.8	37.5	43.8	68.8	100.0
	G_i	-18.8	75.0	6.3	62.5	0.0	-6.3	12.5	43.8	37.5	18.8	18.8	0.0	25.0	31.3	50.0	87.5
	$<g_i>$	0.0	75.0	6.7	71.4	0.0	0.0	40.0	50.0	60.0	30.0	27.3	0.0	28.6	35.7	61.5	100.0

Note: The pre- and post-test results are normalized in percentages

Table E.5 : Control group individual gain factors

Table E.6 : Control group Test 1 (High-Voltage Safety) discretized Grades

TEST 1 - High-Voltage Safety					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N11	0	0	0	0	0
N14	0	0	0	0	0
N13	1	0	0	0	1
N15	0	1	0	0	1
N16	1	0	0	0	1
N6	1	1	0	0	2
N8	1	1	0	0	2
N12	1	1	0	0	2
N1	1	1	1	0	3
N2	1	1	0	1	3
N7	0	1	1	1	3
N10	1	1	0	1	3
N3	1	1	1	1	4
N4	1	1	1	1	4
N5	1	1	1	1	4
N9	1	1	1	1	4
$\boldsymbol{\mu}$	0.75	0.75	0.375	0.438	2.313
$\boldsymbol{\sigma}$	0.447	0.447	0.5	0.512	
$\boldsymbol{I_{Diff}}$	75%	75%	38%	44%	
$\boldsymbol{I_{Disc}}$	0.60	0.80	0.80	1.00	
$\boldsymbol{S^2}$	1.963				
q	0.250	0.250	0.625	0.563	
p*q	0.188	0.188	0.234	0.246	
KR-20	0.752				

Note: The student responses were discretized in 1s and 0s

Table E.7 : Control group Test 2 (IDE, ASM/C Programming & IO) discretized Grades

TEST 2 - IDE, ASM/C Programming & IO					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N9	0	0	0	0	0
N5	0	0	1	0	1
N15	0	0	1	0	1
N1	0	1	0	1	2
N3	1	0	1	0	2
N6	0	0	1	1	2
N8	0	0	1	1	2
N12	0	0	1	1	2
N13	0	1	1	0	2
N4	0	1	1	1	3
N7	1	0	1	1	3
N14	0	1	1	1	3
N2	1	1	1	1	4
N10	1	1	1	1	4
N11	1	1	1	1	4
N16	1	1	1	1	4
$\boldsymbol{\mu}$	0.375	0.5	0.875	0.688	2.438
$\boldsymbol{\sigma}$	0.5	0.516	0.342	0.479	
$\boldsymbol{I_{Diff}}$	38%	50%	88%	69%	
$\boldsymbol{I_{Disc}}$	0.60	0.80	0.40	0.80	
$\boldsymbol{S^2}$	1.463				
q	0.625	0.500	0.125	0.313	
p*q	0.234	0.250	0.109	0.215	
KR-20	0.596				

Note: The student responses were discretized in 1s and 0s

Table E.8 : Control group Test 3 (Interrupt & Switch Debouncing) discretized Grades

TEST 3 - Interrupt & Switch Debouncing					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N2	0	0	0	0	0
N7	0	0	0	0	0
N8	0	0	0	0	0
N3	1	0	0	0	1
N5	1	0	0	0	1
N6	1	0	0	0	1
N11	0	0	1	0	1
N14	1	0	0	0	1
N1	1	0	1	0	2
N4	1	0	0	1	2
N12	1	0	0	1	2
N13	1	0	1	0	2
N15	1	0	1	0	2
N16	1	0	1	0	2
N9	1	1	0	1	3
N10	1	0	1	1	3
$\boldsymbol{\mu}$	0.75	0.063	0.375	0.25	1.438
$\boldsymbol{\sigma}$	0.447	0.25	0.5	0.447	
$\boldsymbol{I_{Diff}}$	75%	6%	38%	25%	
$\boldsymbol{I_{Disc}}$	0.60	0.20	0.80	0.40	
$\boldsymbol{S^2}$	0.929				
q	0.250	0.938	0.625	0.750	
p*q	0.188	0.059	0.234	0.188	
KR-20	0.375				

Note: The student responses were discretized in 1s and 0s

Table E.9 : Control group Test 4 (Timers and Applications) discretized Grades

TEST 4 - Timers and Applications					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N10	0	0	0	0	0
N1	0	0	0	1	1
N3	0	0	0	1	1
N6	0	0	0	1	1
N7	0	1	0	0	1
N12	0	0	0	1	1
N14	1	0	0	0	1
N15	0	1	0	0	1
N2	0	0	1	1	2
N9	0	1	0	1	2
N11	1	0	0	1	2
N4	0	1	1	1	3
N8	1	0	1	1	3
N13	1	0	1	1	3
N16	1	0	1	1	3
N5	1	1	1	1	4
$\boldsymbol{\mu}$	0.375	0.313	0.375	0.75	1.813
$\boldsymbol{\sigma}$	0.5	0.479	0.5	0.447	
$\boldsymbol{I_{Diff}}$	38%	31%	38%	75%	
$\boldsymbol{I_{Disc}}$	0.80	0.20	1.00	0.40	
$\boldsymbol{S^2}$	1.229				
q	0.625	0.688	0.625	0.250	
p*q	0.234	0.215	0.234	0.188	
KR-20	0.388				

Note: The student responses were discretized in 1s and 0s

Table E.10 : Control group Test 5 (Low-Power Modes, LED Display Techniques & keypads) discretized Grades

TEST 5 - Low-Power Modes					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N1	0	0	0	0	0
N2	0	0	0	0	0
N4	0	1	0	0	1
N5	1	0	0	0	1
N6	1	0	0	0	1
N7	0	1	0	0	1
N8	1	0	0	0	1
N9	0	0	1	0	1
N12	0	0	1	0	1
N14	0	0	1	0	1
N15	0	0	1	0	1
N13	1	0	1	0	2
N3	1	1	1	0	3
N10	1	0	1	1	3
N11	1	1	1	0	3
N16	1	0	1	1	3
$\boldsymbol{\mu}$	0.375	0.313	0.375	0.75	1.813
$\boldsymbol{\sigma}$	0.5	0.479	0.5	0.447	
$\boldsymbol{I_{Diff}}$	38%	31%	38%	75%	
$\boldsymbol{I_{Disc}}$	0.80	0.20	1.00	0.40	
$\boldsymbol{S^2}$	1.229				
q	0.625	0.688	0.625	0.250	
p*q	0.234	0.215	0.234	0.188	
KR-20	0.388				

Note: The student responses were discretized in 1s and 0s

Table E.11 : Control group Test 6 (Introduction to Serial Communications) discretized Grades

TEST 6 - Serial Communications					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N1	0	0	0	0	0
N3	0	0	0	0	0
N5	0	0	0	0	0
N6	0	0	0	1	1
N12	0	0	0	1	1
N13	0	0	0	1	1
N14	0	0	0	1	1
N4	1	0	0	1	2
N8	1	0	0	1	2
N10	1	0	0	1	2
N11	1	0	0	1	2
N15	1	0	0	1	2
N2	0	1	1	1	3
N7	1	0	1	1	3
N9	1	1	0	1	3
N16	1	1	1	1	4
$\boldsymbol{\mu}$	0.5	0.188	0.188	0.813	1.688
$\boldsymbol{\sigma}$	0.516	0.403	0.403	0.403	
$\boldsymbol{I_Diff}$	50%	19%	19%	81%	
$\boldsymbol{I_Disc}$	0.80	0.60	0.60	0.60	
$\boldsymbol{S^2}$	1.429				
q	0.500	0.813	0.813	0.188	
p*q	0.250	0.152	0.152	0.152	
KR-20	0.674				

Note: The student responses were discretized in 1s and 0s

E.2 Experimental Group Student Grades

Table E.12 : Experimental group pre-tests grades

	Pre-Test								
Student	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**μ Student**
N1	0.0	17.0	2.5	26.0	12.3	18.0	8.0	15.0	12.4
N2	5.0	22.5	30.0	22.0	10.8	5.0	20.0	7.5	15.4
N3	15.0	15.0	7.5	23.0	8.5	10.0	7.5	15.0	12.7
N4	17.0	15.0	15.5	5.0	2.5	8.5	0.0	6.0	8.7
N5	20.0	5.0	8.0	16.5	12.6	9.0	13.0	15.0	12.4
N6	0.0	15.0	7.0	17.5	7.5	5.0	4.5	7.5	8.0
N7	10.0	20.0	11.0	9.5	22.1	7.0	5.0	11.0	12.0
N8	12.0	15.0	21.0	24.5	34.0	10.0	7.5	12.5	17.1
N9	8.5	27.0	18.0	33.0	20.1	19.0	12.0	1.0	17.3
N10	0.0	6.0	11.0	12.0	11.5	0.0	10.0	0.0	6.3
N11	12.5	19.0	13.5	16.7	27.3	8.3	10.5	18.0	15.7
N12	5.0	15.5	15.5	28.0	15.3	7.0	7.5	10.5	13.0
N13	7.5	24.0	17.5	29.0	11.6	17.0	12.0	5.0	15.5
N14	15.0	20.0	22.0	29.0	12.8	5.0	2.5	15.0	15.2
N15	15.0	23.0	10.0	27.0	24.6	4.0	11.0	19.0	16.7
N16	2.5	34.0	10.0	31.0	30.0	10.3	20.0	12.5	18.8
N17	9.5	34.0	26.0	10.0	11.5	5.3	16.5	10.0	15.3
N18	18.0	27.0	24.5	14.0	16.6	8.5	2.5	9.0	15.0
N19	0.0	35.0	26.5	22.0	29.5	20.8	12.5	8.0	19.3
N20	10.0	24.0	16.5	18.0	15.8	15.3	0.0	21.0	15.1
N21	8.5	23.0	17.5	20.0	13.3	5.0	9.5	14.0	13.9
N22	2.5	24.0	15.0	14.0	13.3	0.0	0.0	11.5	10.0
N23	15.0	7.0	14.5	39.0	25.5	8.5	10.0	8.5	16.0
N24	7.5	15.0	9.0	13.0	12.5	0.0	14.5	5.0	9.6
N25	5.0	20.0	16.5	16.0	2.5	14.0	2.0	15.0	11.4
μ Test	8.8	20.1	15.4	20.6	16.2	8.8	8.7	10.9	13.7
σ	6.1	8.0	6.8	8.3	8.4	5.8	5.8	5.3	

Note: The maximum grade achievable is 40 pts

Table E.13 : Experimental group post-tests grades

	Post-Test								
Student	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**μ Student**
N1	32.5	36.0	24.0	31.5	24.3	25.0	29.0	27.0	28.7
N2	37.5	28.0	30.5	38.0	30.6	29.5	26.0	14.5	29.3
N3	40.0	29.0	35.0	38.0	35.5	33.0	30.0	17.0	32.2
N4	30.0	33.0	33.0	38.5	34.1	35.0	30.5	25.5	32.5
N5	30.0	31.0	25.0	20.0	26.5	27.0	30.0	15.5	25.6
N6	40.0	40.0	32.0	34.5	35.0	38.0	37.0	20.0	34.6
N7	30.0	38.0	32.5	35.0	25.0	24.0	36.0	32.5	31.6
N8	37.5	36.0	32.5	33.5	17.5	35.0	16.5	28.5	29.6
N9	40.0	34.0	30.0	35.0	38.5	35.0	20.5	26.5	32.4
N10	37.5	28.0	36.0	33.5	30.0	35.0	23.0	13.5	29.6
N11	37.5	24.0	25.0	36.5	31.6	29.3	12.0	13.5	26.2
N12	35.0	28.0	39.0	39.5	34.1	37.5	40.0	17.5	33.8
N13	40.0	38.0	32.0	33.0	26.1	32.8	20.5	12.0	29.3
N14	35.0	31.0	35.0	33.5	33.5	25.0	40.0	32.5	33.2
N15	22.5	34.0	33.0	37.5	37.5	28.5	28.0	38.0	32.4
N16	38.0	38.0	34.0	34.0	37.5	26.0	26.0	31.5	33.1
N17	35.0	38.0	36.0	31.5	33.5	35.5	40.0	32.0	35.2
N18	32.5	26.0	31.0	30.0	33.0	33.0	30.0	29.0	30.6
N19	35.0	35.0	34.5	32.0	33.5	31.8	40.0	33.0	34.4
N20	40.0	33.0	26.5	33.0	30.6	22.5	35.5	38.5	32.5
N21	35.0	34.0	25.0	30.0	33.0	27.5	38.0	27.0	31.2
N22	37.5	32.0	23.5	31.0	37.0	33.5	10.0	34.0	29.8
N23	40.0	25.0	29.0	39.0	36.0	28.6	31.0	37.0	33.2
N24	30.0	40.0	13.5	19.0	27.6	24.0	32.0	17.5	25.5
N25	37.5	36.0	35.0	36.0	35.5	26.5	28.0	27.0	32.7
μ Test	35.4	33.0	30.5	33.3	31.9	30.3	29.2	25.6	31.2
σ	4.4	4.7	5.5	5.0	5.0	4.6	8.5	8.4	

Note: The maximum grade achievable is 40 pts

Table E.14 : Experimental group normalized pre-tests grades

	Pre-Test (%)								
Student	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**μ Student**
N1	0.0	42.5	6.3	65.0	30.8	45.0	20.0	37.5	30.9
N2	12.5	56.3	75.0	55.0	27.0	12.5	50.0	18.8	38.4
N3	37.5	37.5	18.8	57.5	21.3	25.0	18.8	37.5	31.7
N4	42.5	37.5	38.8	12.5	6.3	21.3	0.0	15.0	21.7
N5	50.0	12.5	20.0	41.3	31.5	22.5	32.5	37.5	31.0
N6	0.0	37.5	17.5	43.8	18.8	12.5	11.3	18.8	20.0
N7	25.0	50.0	27.5	23.8	55.3	17.5	12.5	27.5	29.9
N8	30.0	37.5	52.5	61.3	85.0	25.0	18.8	31.3	42.7
N9	21.3	67.5	45.0	82.5	50.3	47.5	30.0	2.5	43.3
N10	0.0	15.0	27.5	30.0	28.8	0.0	25.0	0.0	15.8
N11	31.3	47.5	33.8	41.8	68.3	20.6	26.3	45.0	39.3
N12	12.5	38.8	38.8	70.0	38.3	17.5	18.8	26.3	32.6
N13	18.8	60.0	43.8	72.5	29.0	42.5	30.0	12.5	38.6
N14	37.5	50.0	55.0	72.5	32.0	12.5	6.3	37.5	37.9
N15	37.5	57.5	25.0	67.5	61.5	10.0	27.5	47.5	41.8
N16	6.3	85.0	25.0	77.5	75.0	25.6	50.0	31.3	47.0
N17	23.8	85.0	65.0	25.0	28.8	13.1	41.3	25.0	38.4
N18	45.0	67.5	61.3	35.0	41.5	21.3	6.3	22.5	37.5
N19	0.0	87.5	66.3	55.0	73.8	51.9	31.3	20.0	48.2
N20	25.0	60.0	41.3	45.0	39.5	38.1	0.0	52.5	37.7
N21	21.3	57.5	43.8	50.0	33.3	12.5	23.8	35.0	34.6
N22	6.3	60.0	37.5	35.0	33.3	0.0	0.0	28.8	25.1
N23	37.5	17.5	36.3	97.5	63.8	21.3	25.0	21.3	40.0
N24	18.8	37.5	22.5	32.5	31.3	0.0	36.3	12.5	23.9
N25	12.5	50.0	41.3	40.0	6.3	35.0	5.0	37.5	28.4
μ Test	22.1	50.2	38.6	51.6	40.4	22.0	21.9	27.3	34.2
σ	15.0	19.6	16.7	20.4	20.6	14.2	14.2	12.9	

Note: The student grades were normalized to be presented in percentages

Table E.15 : Experimental group normalized post-tests grades

	Post-Test (%)								
Student	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**μ Student**
N1	81.3	90.0	60.0	78.8	60.8	62.5	72.5	67.5	71.7
N2	93.8	70.0	76.3	95.0	76.5	73.8	65.0	73.8	73.3
N3	100.0	72.5	87.5	95.0	88.8	82.5	75.0	82.5	80.5
N4	75.0	82.5	82.5	96.3	85.3	87.5	76.3	87.5	81.1
N5	75.0	77.5	62.5	50.0	66.3	67.5	75.0	67.5	64.1
N6	100.0	100.0	80.0	86.3	87.5	95.0	92.5	95.0	86.4
N7	75.0	95.0	81.3	87.5	62.5	60.0	90.0	60.0	79.1
N8	93.8	90.0	81.3	83.8	43.8	87.5	41.3	87.5	74.1
N9	100.0	85.0	75.0	87.5	96.3	87.5	51.3	87.5	81.1
N10	93.8	70.0	90.0	83.8	75.0	87.5	57.5	87.5	73.9
N11	93.8	60.0	62.5	91.3	79.0	73.3	30.0	73.3	65.4
N12	87.5	70.0	97.5	98.8	85.3	93.8	100.0	93.8	84.6
N13	100.0	95.0	80.0	82.5	65.3	81.9	51.3	81.9	73.2
N14	87.5	77.5	87.5	83.8	83.8	62.5	100.0	62.5	83.0
N15	56.3	85.0	82.5	93.8	93.8	71.3	70.0	71.3	80.9
N16	95.0	95.0	85.0	85.0	93.8	65.0	65.0	65.0	82.8
N17	87.5	95.0	90.0	78.8	83.8	88.8	100.0	88.8	88.0
N18	81.3	65.0	77.5	75.0	82.5	82.5	75.0	82.5	76.4
N19	87.5	87.5	86.3	80.0	83.8	79.5	100.0	79.5	85.9
N20	100.0	82.5	66.3	82.5	76.5	56.3	88.8	56.3	81.1
N21	87.5	85.0	62.5	75.0	82.5	68.8	95.0	68.8	78.0
N22	93.8	80.0	58.8	77.5	92.5	83.8	25.0	83.8	74.5
N23	100.0	62.5	72.5	97.5	90.0	71.5	77.5	71.5	83.0
N24	75.0	100.0	33.8	47.5	69.0	60.0	80.0	60.0	63.6
N25	93.8	90.0	87.5	90.0	88.8	66.3	70.0	66.3	81.7
μ Test	88.6	82.5	76.3	83.3	79.7	75.8	73.0	64.1	77.9
σ	10.7	11.5	13.6	12.3	12.3	11.4	20.9	20.7	

Note: The student grades were normalized to be presented in percentages

Table E.16 : Experimental group individual gain factors

		Student (%)															
		N1	**N2**	**N3**	**N4**	**N5**	**N6**	**N7**	**N8**	**N9**	**N10**	**N11**	**N12**	**N13**	**N14**	**N15**	**N16**
Test 1	Pre	0.0	12.5	37.5	42.5	50.0	0.0	25.0	30.0	21.3	0.0	31.3	12.5	18.8	37.5	37.5	6.3
	Post	81.3	93.8	100.0	75.0	75.0	100.0	75.0	93.8	100.0	93.8	93.8	87.5	100.0	87.5	56.3	95.0
	G_i	81.3	81.3	62.5	32.5	25.0	100.0	50.0	63.8	78.8	93.8	62.5	75.0	81.3	50.0	18.8	88.8
	$<g_i>$	81.3	92.9	100.0	56.5	50.0	100.0	66.7	91.1	100.0	93.8	90.9	85.7	100.0	80.0	30.0	94.7
Test 2	Pre	42.5	56.3	37.5	37.5	12.5	37.5	50.0	37.5	67.5	15.0	47.5	38.8	60.0	50.0	57.5	85.0
	Post	90.0	70.0	72.5	82.5	77.5	100.0	95.0	90.0	85.0	70.0	60.0	70.0	95.0	77.5	85.0	95.0
	G_i	47.5	13.8	35.0	45.0	65.0	62.5	45.0	52.5	17.5	55.0	12.5	31.3	35.0	27.5	27.5	10.0
	$<g_i>$	82.6	31.4	56.0	72.0	74.3	100.0	90.0	84.0	53.8	64.7	23.8	51.0	87.5	55.0	64.7	66.7
Test 3	6.3	75.0	18.8	38.8	20.0	17.5	27.5	52.5	45.0	27.5	33.8	38.8	43.8	55.0	25.0	25.0	
	Post	60.0	76.3	87.5	82.5	62.5	80.0	81.3	81.3	75.0	90.0	62.5	97.5	80.0	87.5	82.5	85.0
	G_i	53.8	1.3	68.8	43.8	42.5	62.5	53.8	28.8	30.0	62.5	28.8	58.8	36.3	32.5	57.5	60.0
	$<g_i>$	57.3	5.0	84.6	71.4	53.1	75.8	74.1	60.5	54.5	86.2	43.4	95.9	64.4	72.2	76.7	80.0
Test 4	Pre	65.0	55.0	57.5	12.5	41.3	43.8	23.8	61.3	82.5	30.0	41.8	70.0	72.5	72.5	67.5	77.5
	Post	78.8	95.0	95.0	96.3	50.0	86.3	87.5	83.8	87.5	83.8	91.3	98.8	82.5	83.8	93.8	85.0
	G_i	13.8	40.0	37.5	83.8	8.8	42.5	63.8	22.5	5.0	53.8	49.5	28.8	10.0	11.3	26.3	7.5
	$<g_i>$	39.3	88.9	88.2	95.7	14.9	75.6	83.6	58.1	28.6	76.8	85.0	95.8	36.4	40.9	80.8	33.3
Test 5	Pre	30.8	27.0	21.3	6.3	31.5	18.8	55.3	85.0	50.3	28.8	68.3	38.3	29.0	32.0	61.5	75.0
	Post	60.8	76.5	88.8	85.3	66.3	87.5	62.5	43.8	96.3	75.0	79.0	85.3	65.3	83.8	93.8	93.8
	G_i	30.0	49.5	67.5	79.0	34.8	68.8	7.3	-41.3	46.0	46.3	10.8	47.0	36.3	51.8	32.3	18.8
	$<g_i>$	43.3	67.8	85.7	84.3	50.7	84.6	16.2	0.0	92.5	64.9	33.9	76.1	51.1	76.1	83.8	75.0
Test 6	Pre	45.0	12.5	25.0	21.3	22.5	12.5	17.5	25.0	47.5	0.0	20.6	17.5	42.5	12.5	10.0	25.6
	Post	62.5	73.8	82.5	87.5	67.5	95.0	60.0	87.5	87.5	87.5	73.3	93.8	81.9	62.5	71.3	65.0
	G_i	17.5	61.3	57.5	66.3	45.0	82.5	42.5	62.5	40.0	87.5	52.6	76.3	39.4	50.0	61.3	39.4
	$<g_i>$	31.8	70.0	76.7	84.1	58.1	94.3	51.5	83.3	76.2	87.5	66.3	92.4	68.5	57.1	68.1	52.9
Test 7	Pre	20.0	50.0	18.8	0.0	32.5	11.3	12.5	18.8	30.0	25.0	26.3	18.8	30.0	6.3	27.5	50.0
	Post	72.5	65.0	75.0	76.3	75.0	92.5	90.0	41.3	51.3	57.5	30.0	100.0	51.3	100.0	70.0	65.0
	G_i	52.5	15.0	56.3	76.3	42.5	81.3	77.5	22.5	21.3	32.5	3.8	81.3	21.3	93.8	42.5	15.0
	$<g_i>$	65.6	30.0	69.2	76.3	63.0	91.5	88.6	27.7	30.4	43.3	5.1	100.0	30.4	100.0	58.6	30.0
Test 8	Pre	37.5	18.8	37.5	15.0	37.5	18.8	27.5	31.3	2.5	0.0	45.0	26.3	0.0	37.5	47.5	31.3
	Post	67.5	36.3	42.5	63.8	38.8	50.0	81.3	71.3	66.3	33.8	33.8	43.8	30.0	81.3	95.0	78.8
	G_i	30.0	17.5	5.0	48.8	1.3	31.3	53.8	40.0	63.8	33.8	-11.3	17.5	17.5	43.8	47.5	47.5
	$<g_i>$	48.0	21.5	8.0	57.4	2.0	38.5	74.1	58.2	65.4	33.8	0.0	23.7	20.0	70.0	90.5	69.1

Note: The pre- and post-test results are normalized in percentages

Table E.17 : Experimental group individual gain factors (continuation)

		Student (%)								
		N17	**N18**	**N19**	**N20**	**N21**	**N22**	**N23**	**N24**	**N25**
Test 1	Pre	23.8	45.0	0.0	25.0	21.3	6.3	37.5	18.8	12.5
	Post	87.5	81.3	87.5	100.0	87.5	93.8	100.0	75.0	93.8
	G_i	63.8	36.3	87.5	75.0	66.3	87.5	62.5	56.3	81.3
	$<g_i>$	83.6	65.9	87.5	100.0	84.1	93.3	100.0	69.2	92.9
Test 2	Pre	85.0	67.5	87.5	60.0	57.5	60.0	17.5	37.5	50.0
	Post	95.0	65.0	87.5	82.5	85.0	80.0	62.5	100.0	90.0
	G_i	10.0	-2.5	0.0	22.5	27.5	20.0	45.0	62.5	40.0
	$<g_i>$	66.7	0.0	0.0	56.3	64.7	50.0	54.5	100.0	80.0
Test 3	Pre	65.0	61.3	66.3	41.3	43.8	37.5	36.3	22.5	41.3
	Post	90.0	77.5	86.3	66.3	62.5	58.8	72.5	33.8	87.5
	G_i	25.0	16.3	20.0	25.0	18.8	21.3	36.3	11.3	46.3
	$<g_i>$	71.4	41.9	59.3	42.6	33.3	34.0	56.9	14.5	78.7
Test 4	Pre	25.0	35.0	55.0	45.0	50.0	35.0	97.5	32.5	40.0
	Post	78.8	75.0	80.0	82.5	75.0	77.5	97.5	47.5	90.0
	G_i	53.8	40.0	25.0	37.5	25.0	42.5	0.0	15.0	50.0
	$<g_i>$	71.7	61.5	55.6	68.2	50.0	65.4	0.0	22.2	83.3
Test 5	Pre	28.8	41.5	73.8	39.5	33.3	33.3	63.8	31.3	6.3
	Post	83.8	82.5	83.8	76.5	82.5	92.5	90.0	69.0	88.8
	G_i	55.0	41.0	10.0	37.0	49.3	59.3	26.3	37.8	82.5
	$<g_i>$	77.2	70.1	38.1	61.2	73.8	88.8	72.4	54.9	88.0
Test 6	Pre	13.1	21.3	51.9	38.1	12.5	0.0	21.3	0.0	35.0
	Post	88.8	82.5	79.5	56.3	68.8	83.8	71.5	60.0	66.3
	G_i	75.6	61.3	27.6	18.1	56.3	83.8	50.3	60.0	31.3
	$<g_i>$	87.1	77.8	57.4	29.3	64.3	83.8	63.8	60.0	48.1
Test 7	Pre	41.3	6.3	31.3	0.0	23.8	0.0	25.0	36.3	5.0
	Post	100.0	75.0	100.0	88.8	95.0	25.0	77.5	80.0	70.0
	G_i	58.8	68.8	68.8	88.8	71.3	25.0	52.5	43.8	65.0
	$<g_i>$	100.0	73.3	100.0	88.8	93.4	25.0	70.0	68.6	68.4
Test 8	Pre	25.0	22.5	20.0	52.5	35.0	28.8	21.3	12.5	37.5
	Post	80.0	72.5	82.5	96.3	67.5	85.0	92.5	43.8	67.5
	G_i	55.0	50.0	62.5	43.8	32.5	56.3	71.3	31.3	30.0
	$<g_i>$	73.3	64.5	78.1	92.1	50.0	78.9	90.5	35.7	48.0

Note: The pre- and post-test results are normalized in percentages

Table E.18 : Experimental group Test 1 (High-Voltage Safety) discretized Grades

TEST 1 - High-Voltage Safety					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N5	1	0	0	1	2
N7	1	0	0	1	2
N15	1	0	0	1	2
N24	1	1	0	0	2
N1	1	1	0	1	3
N4	0	1	1	1	3
N12	1	0	1	1	3
N14	1	1	1	0	3
N17	1	1	1	0	3
N18	1	0	1	1	3
N19	1	1	1	0	3
N21	1	1	0	1	3
N2	1	1	1	1	4
N3	1	1	1	1	4
N6	1	1	1	1	4
N8	1	1	1	1	4
N9	1	1	1	1	4
N10	1	1	1	1	4
N11	1	1	1	1	4
N13	1	1	1	1	4
N16	1	1	1	1	4
N20	1	1	1	1	4
N22	1	1	1	1	4
N23	1	1	1	1	4
N25	1	1	1	1	4
$\boldsymbol{\mu}$	0.960	0.800	0.760	0.840	3.360
$\boldsymbol{\sigma}$	0.200	0.408	0.436	0.374	
$\boldsymbol{I_{Diff}}$	96%	80%	76%	84%	
$\boldsymbol{I_{Disc}}$	0.167	0.500	0.833	0.167	
$\boldsymbol{S^2}$	0.573				
q	0.040	0.200	0.240	0.160	
p*q	0.038	0.160	0.182	0.134	
KR-20	0.135				

Note: The student responses were discretized in 1s and 0s

Table E.19 : Experimental group Test 2 (IDE, GPIOs, and LCD) discretized Grades

TEST 2 - IDE, GPIOs, and LCD					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N11	1	0	0	0	1
N22	0	0	1	1	2
N2	1	0	1	1	3
N3	1	0	1	1	3
N4	1	1	0	1	3
N5	1	0	1	1	3
N10	1	0	1	1	3
N12	1	0	1	1	3
N14	0	1	1	1	3
N18	1	0	1	1	3
N19	1	0	1	1	3
N20	1	0	1	1	3
N23	1	0	1	1	3
N1	1	1	1	1	4
N6	1	1	1	1	4
N7	1	1	1	1	4
N8	1	1	1	1	4
N9	1	1	1	1	4
N13	1	1	1	1	4
N15	1	1	1	1	4
N16	1	1	1	1	4
N17	1	1	1	1	4
N21	1	1	1	1	4
N24	1	1	1	1	4
N25	1	1	1	1	4
$\boldsymbol{\mu}$	0.920	0.560	0.920	0.960	3.360
$\boldsymbol{\sigma}$	0.277	0.507	0.277	0.200	
$\boldsymbol{I_{Diff}}$	92%	56%	92%	96%	
$\boldsymbol{I_{Disc}}$	0.167	0.833	0.333	0.167	
$\boldsymbol{S^2}$	0.573				
q	0.080	0.440	0.080	0.040	
p*q	0.074	0.246	0.074	0.038	
KR-20	0.329				

Note: The student responses were discretized in 1s and 0s

Table E.20 : Experimental group Test 3 (Interrupts, Switch Debouncing, and Keypad) discretized Grades

TEST 3 - Interrupts, Switch Debouncing, and Keypad					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N24	0	0	0	0	0
N5	1	0	0	0	1
N11	0	0	0	1	1
N1	1	0	0	1	2
N20	0	1	0	1	2
N21	0	1	0	1	2
N22	1	1	0	0	2
N23	1	1	0	0	2
N2	1	1	0	1	3
N4	1	1	0	1	3
N7	1	1	0	1	3
N9	1	1	0	1	3
N13	1	1	0	1	3
N14	1	1	0	1	3
N15	1	1	0	1	3
N16	1	1	0	1	3
N27	1	1	0	1	3
N18	1	1	0	1	3
N25	1	1	0	1	3
N3	1	1	1	1	4
N6	1	1	1	1	4
N8	1	1	1	1	4
N10	1	1	1	1	4
N12	1	1	1	1	4
N19	1	1	1	1	4
$\boldsymbol{\mu}$	0.840	0.840	0.240	0.840	2.760
$\boldsymbol{\sigma}$	0.374	0.374	0.436	0.374	
$\boldsymbol{I_{Diff}}$	84%	84%	24%	84%	
$\boldsymbol{I_{Disc}}$	0.667	0.667	1.000	0.333	
$\boldsymbol{S^2}$	1.107				
q	0.160	0.160	0.760	0.160	
p*q	0.134	0.134	0.182	0.134	
KR-20	0.628				

Note: The student responses were discretized in 1s and 0s

Table E.21 : Experimental group Test 4 (Timers and LEDs) discretized Grades

TEST 4 - Timers and LEDs					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N5	0	0	0	0	0
N24	0	0	0	1	1
N17	0	1	1	0	2
N18	1	0	1	0	2
N1	0	1	1	1	3
N8	0	1	1	1	3
N10	0	1	1	1	3
N13	1	1	0	1	3
N14	1	1	0	1	3
N19	0	1	1	1	3
N20	0	1	1	1	3
N21	0	1	1	1	3
N22	0	1	1	1	3
N2	1	1	1	1	4
N3	1	1	1	1	4
N4	1	1	1	1	4
N6	1	1	1	1	4
N7	1	1	1	1	4
N9	1	1	1	1	4
N11	1	1	1	1	4
N12	1	1	1	1	4
N15	1	1	1	1	4
N16	1	1	1	1	4
N23	1	1	1	1	4
N25	1	1	1	1	4
$\boldsymbol{\mu}$	0.600	0.880	0.840	0.880	3.200
$\boldsymbol{\sigma}$	0.500	0.332	0.374	0.332	
$\boldsymbol{I_{Diff}}$	60%	88%	84%	88%	
$\boldsymbol{I_{Disc}}$	0.833	0.500	0.333	0.500	
$\boldsymbol{S^2}$	1.083				
q	0.400	0.120	0.160	0.120	
p*q	0.240	0.106	0.134	0.106	
KR-20	0.613				

Note: The student responses were discretized in 1s and 0s

Table E.22 : Experimental group Test 5 (Low-Power Modes and PWM) discretized Grades

TEST 5 - Low-Power Modes and PWM					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N7	0	0	0	1	1
N9	0	0	0	1	1
N1	1	1	0	0	2
N2	1	0	1	0	2
N5	0	0	1	1	2
N11	0	0	1	1	2
N12	0	1	1	0	2
N14	1	1	0	0	2
N27	1	1	0	0	2
N4	1	1	1	0	3
N13	1	1	1	0	3
N17	0	1	1	1	3
N20	0	1	1	1	3
N21	1	0	1	1	3
N22	1	1	0	1	3
N23	1	1	1	0	3
N3	1	1	1	1	4
N6	1	1	1	1	4
N10	1	1	1	1	4
N18	1	1	1	1	4
N19	1	1	1	1	4
N24	1	1	1	1	4
N25	1	1	1	1	4
N26	1	1	1	1	4
N28	1	1	1	1	4
$\boldsymbol{\mu}$	0.720	0.760	0.760	0.680	2.920
$\boldsymbol{\sigma}$	0.458	0.436	0.436	0.476	
$\boldsymbol{I_{Diff}}$	72%	76%	76%	68%	
$\boldsymbol{I_{Disc}}$	0.667	0.833	0.500	0.333	
$\boldsymbol{S^2}$	0.993				
q	0.280	0.240	0.240	0.320	
p*q	0.202	0.182	0.182	0.218	
KR-20	0.281				

Note: The student responses were discretized in 1s and 0s

Table E.23 : Experimental group Test 6 (Motor Interfacing) discretized Grades

TEST 6 - Motor Interfacing					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N7	0	1	0	0	1
N23	0	1	0	0	1
N27	0	0	1	0	1
N28	1	0	0	0	1
N1	0	1	0	1	2
N5	1	1	0	0	2
N17	1	1	0	0	2
N18	1	0	1	0	2
N19	1	0	1	0	2
N21	1	0	1	0	2
N24	1	0	1	0	2
N26	0	1	1	0	2
N14	1	0	1	1	3
N2	1	1	1	0	3
N3	1	1	1	0	3
N4	1	1	1	0	3
N10	0	1	1	1	3
N11	1	1	1	0	3
N12	1	1	1	0	3
N22	1	1	1	0	3
N25	1	0	1	1	3
N6	1	1	1	1	4
N9	1	1	1	1	4
N13	1	1	1	1	4
N20	1	1	1	1	4
$\boldsymbol{\mu}$	0.760	0.680	0.760	0.320	2.520
$\boldsymbol{\sigma}$	0.436	0.476	0.436	0.476	
$\boldsymbol{I_{Diff}}$	76%	68%	76%	32%	
$\boldsymbol{I_{Disc}}$	0.667	0.167	0.833	0.667	
$\boldsymbol{S^2}$	0.927				
q	0.240	0.320	0.240	0.680	
p*q	0.182	0.218	0.182	0.218	
KR-20	0.182				

Note: The student responses were discretized in 1s and 0s

Table E.24 : Experimental group Test 7 (Serial Communications) discretized Grades

TEST 7 - Serial Communications					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N9	0	1	0	0	1
N12	0	1	0	0	1
N25	1	0	0	0	1
N5	0	1	1	0	2
N10	1	0	0	1	2
N11	0	1	0	1	2
N14	1	1	0	0	2
N19	0	1	0	1	2
N1	1	1	0	1	3
N2	1	1	1	0	3
N3	1	1	0	1	3
N4	1	1	0	1	3
N18	1	1	0	1	3
N21	0	1	1	1	3
N26	1	1	1	0	3
N27	1	1	1	0	3
N28	1	1	0	1	3
N6	1	1	1	1	4
N7	1	1	1	1	4
N13	1	1	1	1	4
N17	1	1	1	1	4
N20	1	1	1	1	4
N22	1	1	1	1	4
N23	1	1	1	1	4
N24	1	1	1	1	4
$\boldsymbol{\mu}$	0.760	0.920	0.520	0.680	2.880
$\boldsymbol{\sigma}$	0.436	0.277	0.510	0.476	
$\boldsymbol{I_{Diff}}$	76%	92%	52%	68%	
$\boldsymbol{I_{Disc}}$	0.667	0.333	0.833	0.667	
$\boldsymbol{S^2}$	1.027				
q	0.240	0.080	0.480	0.320	
p*q	0.182	0.074	0.250	0.218	
KR-20	0.394				

Note: The student responses were discretized in 1s and 0s

Table E.25 : Experimental group Test 8 (Data Converters (DAC & ADC)) discretized Grades

TEST 8 - Data Converters (DAC & ADC)					
Student	**Q1**	**Q2**	**Q3**	**Q4**	**TOTAL**
N2	0	0	0	0	0
N11	0	0	0	0	0
N13	0	0	0	0	0
N3	1	0	0	0	1
N5	1	0	0	0	1
N10	1	0	0	0	1
N24	0	1	0	0	1
N4	0	1	1	0	2
N6	1	1	0	0	2
N8	1	0	1	0	2
N9	1	1	0	0	2
N12	1	1	0	0	2
N16	1	0	1	0	2
N25	0	0	1	1	2
N1	1	1	1	0	3
N14	1	0	1	1	3
N17	1	0	1	1	3
N18	1	1	1	0	3
N19	1	1	1	0	3
N21	1	1	1	0	3
N22	1	1	1	0	3
N7	1	1	1	1	4
N15	1	1	1	1	4
N20	1	1	1	1	4
N23	1	1	1	1	4
$\boldsymbol{\mu}$	0.760	0.560	0.600	0.280	2.200
$\boldsymbol{\sigma}$	0.436	0.507	0.500	0.458	
$\boldsymbol{I_{Diff}}$	76%	68%	76%	32%	
$\boldsymbol{I_{Disc}}$	0.500	1.000	1.000	0.667	
$\boldsymbol{S^2}$	1.583				
q	0.240	0.440	0.400	0.720	
p*q	0.182	0.246	0.240	0.202	
KR-20	0.600				

Note: The student responses were discretized in 1s and 0s

Appendix F
IRB Acceptance Letters

Comité para la Protección de los Seres Humanos en la Investigación
CPSHI/IRB 00002053
Universidad de Puerto Rico – Recinto Universitario de Mayagüez
Decanato de Asuntos Académicos
Call Box 9000
Mayagüez, PR 00681-9000

26 de agosto de 2015

Danilo Rojas
Ing. Eléctrica y de Computadoras
RUM

Estimado estudiante:

El Comité para la Protección de los Seres Humanos en la Investigación (CPSHI) he considerado su proyecto titulado Modular and *Design Approach for Improving the Learning Process of Undergraduate Students in the Embedded System Course* (# Protocolo 20150811). Luego de evaluar el mismo hemos certificado que este cumple con todos los requisitos para ser aprobado como exento bajo la Categoría 1 del 45 CFR 46.101(b). La determinación de exención implica que su proyecto no requiere ser re-evaluado ni re-autorizado por nuestro comité. Le recordamos que la aprobación emitida por nuestro comité no lo exime de cumplir con cualquier otro requisito institucional o gubernamental relacionado al tema o fuente de financiamiento de su proyecto.

Cualquier cambio al protocolo o a la metodología que altere los criterios de exención deberá ser revisado y aprobado por el CPSHI ANTES de su implantación, excepto en casos en que el cambio sea necesario para eliminar algún riesgo inmediato para los/as participantes. El CPSHI deberá ser notificado de dichos cambios tan pronto le sea posible al/ a la investigador/a. Igualmente, el CPSHI deberá ser informado de inmediato de cualquier efecto adverso o problema inesperado que surgiera con relación al riesgo de los seres humanos, de cualquier queja sobre la conducción de esta investigación y de cualquier violación a la confidencialidad de los participantes.

Atentamente,

Dr. Rafael A. Boglio Martínez
Presidente
CPSHI/IRB
UPR - RUM

Comité para la Protección de los Seres Humanos en la Investigación
CPSHI/IRB 00002053
Universidad de Puerto Rico – Recinto Universitario de Mayagüez
Decanato de Asuntos Académicos
Call Box 9000
Mayagüez, PR 00681-9000

23 de mayo de 2016

Danilo Rojas Méndez
Ing. Eléctrica y de Computadoras
RUM

Estimado estudiante:

Como presidente del Comité para la Protección de los Seres Humanos en la Investigación (CPSHI) he revisado las modificaciones realizadas a su proyecto de investigación titulado **Modular and Design Approach for Improving the Learning Process of Undergraduate Students in the Embedded System Course** (# Protocolo 20150811). Luego de evaluar los cambios, he determinado que los mismos no alteran los criterios de aprobación de su proyecto. Por tanto, el mismo sigue exento bajo la Categoría 1 del 45 CFR 46.101(b).

Cualquier cambio al protocolo o a la metodóloga deberá ser revisado y aprobado por el CPSHI antes de su implantación. El CPSHI deberá ser informado de inmediato de cualquier efecto adverso o problema inesperado que surgiera con relación al riesgo de los seres humanos, de cualquier queja sobre la conducción de esta investigación y de cualquier violación a la confidencialidad de los participantes.

Atentamente,

Dr. Rafael A. Boglio Martínez
Presidente
CPSHI/IRB
UPR - RUM