A METHODOLOGY TO INTEGRATE EMBEDDED SYSTEMS IMPLEMENTATION IN A DIGITAL CONTROL SYSTEMS COURSE

By

Juan Felipe Patarroyo Montenegro

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

May 2015

Approved by:

Gerson Beauchamp-Báez, Ph.D. Chairman, Graduate Committee	Date
Manuel Jiménez-Cedeño, Ph.D. Member, Graduate Committee	Date
Eduardo J. Juan-García, Ph.D. Member, Graduate Committee	Date
Aidsa I. Santiago-Román, Ph.D. Member, Graduate Committee	Date
Pedro Vasquez-Urbano, Ph.D. Graduate Studies Representative	Date
Raúl E. Torres-Muñiz, Ph.D. Department Chairperson	Date

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

A METHODOLOGY TO INTEGRATE EMBEDDED SYSTEMS IMPLEMENTATION IN A DIGITAL CONTROL SYSTEMS COURSE

By

Juan Felipe Patarroyo Montenegro

May 2015

Chair: Dr. Gerson Beauchamp-Báez
Department: Electrical and Computer Engineering Department

This thesis presents the development of a teaching methodology for improving student's understanding on how digital controllers are implemented in the real world. Specifically, the proposed methodology includes a set of workshops to train students in the use of microcontrollers for digital control purposes and in the use of a microcontroller based workstation for implementing the digital controllers. The effectiveness of this approach, compared to other methods (Simulink, LabVIEW, among others) was evaluated with a pedagogical experiment that followed a backward design approach. The workshop assessment methodology is based on the use of a pre and post-test designed for each workshop to measure student outcomes. Finally, the professor evaluated student's performance through oral exams and demonstrations using a rubric that has been designed to measure student outcomes as established by ABET criteria. To assess the contribution of this work, students' performance in the term project was compared to results of previous years. Our analysis revealed that the proposed learning objectives were accomplished.

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

METODOLOGÍA PARA INTEGRAR LA IMPLEMENTACIÓN DE SISTEMAS EMBEBIDOS EN UN CURSO DE CONTROL DIGITAL

Por

Juan Felipe Patarroyo Montenegro

Mayo 2015

Consejero: Dr. Gerson Beauchamp Báez
Departmento: Ingeniería Eléctrica y Computadoras

Esta tesis presenta el desarrollo de una metodología educativa para mejorar el entendimiento de los estudiantes de cómo se implementan los controladores digitales en el mundo real. Específicamente, la metodología propuesta incluye una serie de talleres para adiestrar a los estudiantes en el uso de microcontroladores para control digital y en el uso de una estación basada en microcontrolador para implementar los controladores digitales. La efectividad de este enfoque, comparada con otros métodos (Simulink, LabVIEW, entre otros) fue evaluada mediante un experimento pedagógico que siguió un enfoque de diseño en reversa. La metodología de evaluación de los talleres está basada en el uso de pre y pos-pruebas diseñadas para cada taller para medir los resultados en los estudiantes. Finalmente, el profesor evaluó el desempeño de los estudiantes a través de exámenes orales y demostraciones utilizando una rúbrica que fue diseñada para medir los resultados de los estudiantes de acuerdo a lo establecido por los criterios de ABET. Para evaluar la contribución de este trabajo, el desempeño de los estudiantes en el proyecto final fue comparado con los resultados de años previos. Nuestro análisis reveló que los objetivos educativos propuestos fueron alcanzados.

Dedicated to my family, especially my mother Luz, my father Hector, and my sister Angela, who have always believed in me and gave me their support and love. To my girlfriend Tatiana who has always been by my side giving me the strength to keep me going every day.

Acknowledgements

I would like to express my thanks to my advisor and friend Professor Gerson Beauchamp. Thank you for encouraging my research and helping me to grow as a professional and human being. Also, special thanks to Professor Rogelio Palomera for economically supporting part of this work and also giving me moral support. Additionally, thanks to my committee members, Professor Dr. Aidsa Santiago, Professor Dr. Manuel Jiménez and Professor Dr. Eduardo Juan, for serving as my committee and for reviewing my work. Special thanks to my academic counselor and friend Sandra Montalvo, who gave me her guidance during the experience of being new in a country and a university. I would like to thank all my Puerto Rican friends and professors, who always made me feel at home and showed me the amazing things of this beautiful island. Finally, I would like to express my gratitude to my Colombian friends, who became a second family and supported me during good or bad moments.

Table of Content

List of Tables

List of Figures

Abbreviations and Acronyms

ADC	Analog –to-Digital Converter
B&B	Ball & Beam
DCS	Digital Control Systems
eQEP	Enhanced Quadrature Encoder Peripheral
GUI	Graphical User Interface
IA	Instrumentation Amplifier
LSA	Linear Systems Analysis
MCS	Microcontroller Based System
MCU	Microcontroller Unit
NG	Normalized Average Gain
OBE	Outcome Based Education
PCB	Printed Circuit Board
PICL	Process Instrumentation and Control Laboratory
PWM	Pulse Width Modulation
SG	Score Gain
SRV-02	Quanser's Rotary Servo Base Unit
TI	Texas Instruments®
UPRM	University of Puerto Rico at Mayaguez

Chapter 1: INTRODUCTION

The use of microcontrollers is playing an increasingly important role in a wide range of engineering applications. Specifically, the area of control systems depends significantly on the use of microcontrollers for control systems implementation. Most of the course projects in the Control Systems Area in our institution are implemented using high level simulation tools and data acquisition boards. These methods have yielded excellent results because they offer a block based environment which makes all the internal details transparent for the user. However, they have the disadvantage of not exposing students to implementing digital controllers directly with microprocessors. Furthermore, programming skills are becoming more important nowadays and a drag-and-drop block based environment does not provide enough understanding on how a digital controller is implemented in real world using computer programs.

The *Digital Control System*s (DCS) is a highly demanded course in our undergraduate electrical engineering curriculum. The *Introduction to Control Systems* course as well as basic knowledge of electronics and signals and systems are prerequisites for the DCS course. The topics covered include modeling of discrete-time control systems, using the Z-Transform to analyze discrete-time systems, stability criteria, and root locus design in the Z-domain. These topics appear commonly in digital control systems textbooks [1, 2, 3, 4].

There are two main objectives of the DCS course. First, to teach the fundamentals of digital control theory and second, to teach how to design a discrete-time controller and implement it in the laboratory using embedded systems and high level tools (Simulink, LabVIEW among others). The first objective is usually accomplished through lectures, homework assignments, and simulations. Lectures provide the theoretical background of digital control systems, while homework assignments and simulations are used to reinforce the theoretical knowledge learned by students.

The second objective is traditionally accomplished by following a project-based learning inductive methodology referred to as a Term Project [5]. For this project, students are organized in teams of two or three students and the instructor specifies an assignment to carry out one or more tasks that lead to the development of a final product. Teams are required to design, simulate, and implement a digital controller for the Quanser's Ball and Beam (B&B) system [6] available in the Process Instrumentation and Control Laboratory (PICL) [7]. The B&B system is a classical challenging mechanical system that uses a servomechanism (SRV-02, [8]) to balance a rolling ball on a tilting beam. Figure 1 is a photo of the B&B system. This system is used in our DCS course because it is complicated enough to be challenging and simple enough to be easy to control with a single-input single-output control architecture. The control architecture used is discussed in Chapter 5. The controllers should be implemented with Simulink through Quanser's Quarc® tool and with microcontrollers. Finally, a comparison between both implementations should be made.

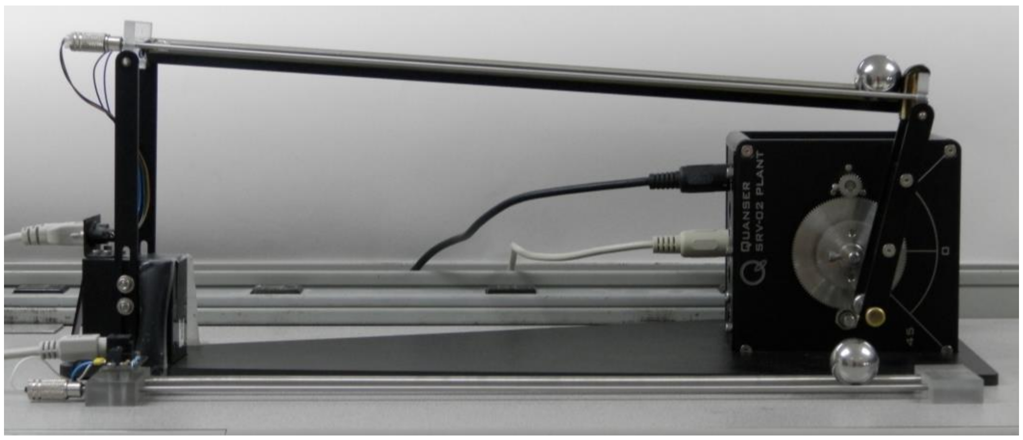


Figure 1. Photo of the Ball & Beam System

To document the term project, students deliver a work plan, two progress reports, and a final report. The work plan specifies how each team will complete the tasks assigned and includes a time schedule with the activities needed to complete the project. Progress reports are used to assess progress and allows for the instructor to provide guidance to the teams accordingly. Teams present a demonstration and an oral test to validate these reports. In the final report, students should use the theory acquired in lectures and assignments to demonstrate that they can design and implement a digital controller for a simple system.

This work was focused on the improvement of the DCS course project. The objective was to design and implement a methodology to teach students in the DCS course to implement digital controllers using a microcontroller based control system (MCS). The research question that guided this study was: **What is the effectiveness of using an MCS to teach students to implement digital controllers in the DCS course?** A backward design approach based on Streveler's et al., [9] was implemented to re-design the DCS course, in which the content, assessment, and pedagogy used to implement the proposed system were aligned. As part of this design, a set of workshops were developed and taught through the course. At the end of the course, students were assessed on how they integrated microcontrollers in the DCS implementation for their projects.

The results of this work are expected to help students to get a better understanding on how digital controllers are implemented in real world using microcontrollers. Additionally, this work could be used in the future for the development of research projects or even another course in our department.

The rest of the thesis is organized as follows. Chapter 2 presents the theoretical foundations and literature review for this work. Chapters 3 include the Problem Statement and Objectives respectively. Chapter 4 contains the technical and educational methodology used to complete this research. The outcomes and analysis of our work are presented in Chapter 5. Conclusions, implications, contributions, and future work are presented in Chapter 6. A set of appendices are provided to enhance this thesis and to provide the reader the basic documents required to implement the methodology presented.

Chapter 2: LITERATURE REVIEW & PREVIOUS WORK

This Chapter reviews the approaches followed in engineering education found in recent literature. First, we briefly introduce the theoretical aspects that should be taken into account in Engineering Education and the importance of course projects in students' learning process. Then we discuss several educational approaches related to Control Systems, Embedded Systems, and Embedded Control Systems.

In the last Section, we present the work done during 2014 at the University of Puerto Rico at Mayaguez regarding the design and implementation of a microcontroller based system presented in this thesis that was used in the DCS course projects.

2.1 Education in Engineering

One of the most common teaching methods in Engineering is the deductive lecture–based learning. Felder, et al., explain that in this approach, the instructor presents a topic, based on general principles, which will be used to derive the mathematical models. Then, some examples on how to use these models are presented to students, and finally, exams will test how students solves similar problems [10]. This methodology continues to have widespread application because it is easy and cheap to implement. That is, a professor can cover a large amount of material in a short amount of time simply by lecturing and presenting derivations. However, Milner-Bolotin et al. show that this approach tends to leave students with an incomplete understanding of the concepts taught in the course [11]. Also, this could leave students thinking that they could never come up with the complete derivation by themselves.

Prince and Felder declare in their study that the inductive method is a natural way of learning where students observe a particular experiment and then infer the governing principles from them [12]. Children learn from examples, first they get familiar with a specific phenomenon and their natural curiosity leads them to ask themselves for an explanation [13]. Felder concludes that engineers prefer the motivational inductive learning style; they need to see the phenomena and acquire curiosity before they can understand the underlying theory [14]. In fact, one of the principles of educational psychology is that people are most strongly motivated to learn things they can clearly perceive and need to know [15].

Some of the most common inductive methods include inquiry learning [16, 17], problem-based learning (PBL) [18, 19, 20], project-based learning [21, 22], case based teaching [23], discovery learning [24] and just-in-time teaching [25]. These methods usually involve students' participation by discussing questions and solving problems by themselves (*Active Learning)*. Also, students are intended to solve problems working in groups (*Collaborative Learning*). Prince and Felder [12] developed Table 1 to compare and summarize inductive methodologies. Each inductive methodology is qualified according to a set of features that most of them share. For example, problem-based learning and case-based learning share the fact that learning depends on the use of complex, ill-structured, open-ended real world problems. Whereas, Project-based learning is based on major projects to provide context for learning and usually is implemented in collaborative team-based learning.

Table 1 - Features of Common inductive instructional Methods. (1) By definition, (2) always, (3) usually, (4) possibly.

Feature\Method	**Inquiry**	**Problem-Based**	**Project-Based.**	**Case-**	**Discovery**	**JiTT**
Questions or problems provide context for learning	1	2	2	2	2	2
Complex, ill-structured, open-ended real-world problems provide context for learning	4	1	3	2	4	4
Major projects provide context for learning	4	4	1	3	4	4
Case studies provide context for learning	4	4	4	1	4	4
Students discover course content for themselves	2	2	2	3	1	2
Students complete & submit conceptual exercises electronically; instructor adjusts lessons according to their responses.	4	4	4	4	4	1
Primarily self-directed learning	4	3	3	3	2	4
Active Learning	2	2	2	2	2	2
Collaborative/cooperative (team-based) learning	4	3	3	4	4	4

Source: Prince and Felder [12]

Pellegrino [26] proposes that inductive methodologies should be designed in such a way that the course objectives, pedagogy, and assessment are aligned. This will guarantee outcomes fulfillment and concrete results in the learning process. Streveler et al., [9] present]] an approach based on Outcome Based Education (OBE) and engineering design methods to align these three elements. That is, start with the requirements or specifications, emphasize metrics, and then prepare prototypes that meet the requirements. This work follows a *Backward Design* approach based on Wiggins and McTighe's book [27] and the *How People Learn* framework provided by Bransford et al. [28].

Gavin [29] used a hybrid problem-project-based learning to teach design skills to civil engineering students. Learning and assessment methods followed a *Backward Design* to meet key learning outcomes. A survey demonstrated student satisfaction with the PBL process and recognized that skills required by industry, such as teamwork, time management and technical competence, were enhanced. The drawback of this approach is that it requires too much time by both students and laboratory staff for coordination of assessment submission dates.

Crespo, et al., [30] proposed a unified model for outcome based education. This approach determines appropriate assessment methods by capturing the influence of learning outcomes in the learning assessment process. The author proposes an application scenario that shows a teacher preparing an Assembly Programming course. First the teacher prepares the desired outcomes based on the student's desired knowledge, skills and competence. Then, the author suggests the use of a database called ICOPER where the teacher is able to search for different assessment resources by only typing the desired learning outcome. Finally, the teacher implements the method and the database is updated with the results. This represents a unified assessment method.

Wang [31] implemented an inductive method to teach nuclear energy using internet resources instead of textbooks. The author organized the course resources so as to facilitate inductive learning and also included enough documents to foster deductive learners as well. In the course, the instructor makes a reference to the most famous nuclear accidents to explain the theoretical principles of a nuclear reactor; thus allowing students to learn inductively. Assessment showed students being more interested in the course and understanding the theoretical background of nuclear reactors. Unfortunately, this approach did not provided any assessment on how students' outcomes were fulfilled.

2.2. Education in Control Systems

Kroumov, et al., developed a set of tools for interactive learning which are aimed at improving the understanding and skills for analysis and design of control systems [32]. The author developed a Graphical User Interface (GUI) to simulate the most typical control problems. For this, the author established a set of requirements that every learning tool should have to be considered user friendly. Students were asked to comment on the problems they faced while using the interactive tools. Results were positive and students declared that the software made it easier to complete the design process. The system only worked as a simulation tool. Unfortunately, the software is very rigid since it only covers some specific topics in a high level environment. Also, there is no evidence of any quantitative assessment method.

Bayrakceken, et al., developed a deductive-inductive outcome-based methodology to teach control systems to undergraduate students using an Unmanned Aerial Vehicle (UAV) and a set of debugging tools [33]. The author designed a quadcopter and used it to develop students' demonstrations and laboratory experiments. In some cases, previous theory was explained to generate a background and show some principles of control theory. In other cases, the demonstrations were shown before presenting the theory to generate curiosity on students. Both demonstrations and experiments were designed in such a way that the desired outcomes were accomplished. To measure the effectiveness of this approach, a student feedback analysis was conducted and instructor observations were considered. Results indicated that students had better performance on the desired learning outcomes when compared to previous years. However, there are no quantitative assessment tools to measure the effectiveness of this approach.

2.3. Education in Embedded Systems

Edson, et al., [34] presented a learning methodology adopted by their university in the Microprocessors Course. It focused on the improvement of the learning process in the microprocessors course by enhancing the development of practical problems. Students developed a series of laboratory experiments specifically designed for the application of the theoretical knowledge learned in class. The authors explain that the learning process involves four main pillars: To learn to know, to learn to make, to learn to live together, and to learn to be. The effectiveness of this approach was verified using a standard evaluation AVIN (from the Portuguese, AValiação INtegradora) [35]; they concluded that students obtained good solutions in the questions that involve the knowledge of Microprocessors.

Subbian et al., [36] presented experiences from teaching and redesigning an advanced embedded systems course that integrates various hardware and software concepts to accommodate the needs of interdisciplinary engineering education. This allowed prerequisites and knowledge requirements for this course to be reduced to a certain minimum level and restrictions could be placed at the program-level. Learning objectives were set according to the cognitive domain of Bloom's Taxonomy [37]. The author suggests a project-oriented method, an Expertise and Project Team Planning survey was given on the first day of class and students were asked to rate their expertise or experience in various aspects. A student survey showed that the revised version of the course was mostly successful in accommodating students from various disciplines.

Kumar et al., [38] proposed a project-oriented methodology for embedded systems education using an FPGA platform. The authors used a generic architecture, containing multiple processors that allowed for easy integration of custom and/or predefined peripherals. The advantage of this approach was that it could be used in multidisciplinary projects due to the architecture flexibility. Each lab has a tutorial where step-by-step guidance was given and an assignment to apply concepts learned in this tutorial. Student feedback analysis showed that students prefer simple tools. However, they agreed that using an FPGA fosters long-term learning. One of the limitations of the project was that a significant number of FPGA boards were necessary since the project requires extensive use of hardware, this represents high expenses.

2.4. Education in Embedded Control Systems

Beauchamp, Jimenez, Mulero, et al., [39] analyzed the performance of two implementation methods for the controller of a Ball-and-Beam system, using a Microcontroller Unit and using Simulink. They used the bilinear transformation for converting a continuous time controller and the direct z-domain design process to implement the controllers. The authors showed that the implemented controller yielded similar but not identical results in both implementations (Simulink, MCU). This work followed a hybrid problem-based methodology which was implemented with a group of students of the University of Puerto Rico at Mayaguez (UPRM). A limitation of this work is that there were no formal methods to validate assessment and the effectiveness of this approach in an educational environment.

Following, Jimenez, Beauchamp, Mulero, et al. [39], described an integrated experience in two different engineering areas: Linear Systems Analysis (LSA) and Microprocessor Interfacing Course (MI) [40]. In this project an embedded state feedback controller for a 3 Degree of Freedom Helicopter was designed and implemented using a TI C2000 multi-core microcontroller. The integration of these two courses showed excellent results and the project is a platform for future implementation of digital controllers using embedded systems. Unfortunately, these projects did not used quantitative assessment methods to validate students understanding of the learned concepts.

In the fall of 2013 Patarroyo and Bolívar implemented a digital controller for the Ball & Beam system (B&B) as part of the UPRM Digital Control Systems course project [41]. This work was based on previous implementations and the TI C2000 F28069 Microcontroller [42], the TI DRV8833 motor driver [43], and a homemade analog signal conditioning board. Although this system had excellent performance, it lacked of an organized wiring structure and an intuitive operation mode.

During the summer of 2014, Patarroyo, developed a formal version of the system mentioned above with educational purposes for the Electrical and Computer Engineering Department at UPRM [44]. The system was enclosed in a box with all the ports needed to connect to the PICL Experiments. A custom made analog signal conditioning board read the sensors and provided the required voltage range (0-3.3V) for the microcontroller Analog to Digital Converter (ADC). A Graphical User Interface was implemented using Code Composer Studio V5.5 and the Direct Memory Access (DMA) Tool. A set of C Code Libraries was created and Discrete Time Controllers for the Quanser's Ball & Beam System and the Single Inverted Pendulum System were developed.

Krauss et al., [45] performed real time feedback control experiments with educational purposes. They used a low-cost microcontroller system connected to a PC in 3 different modes: *Prototyping, Hybrid,* and *Fully Embedded*. The system was tested with undergraduate students by providing them some code examples and a sample project. A total of five quizzes with the same questions and a perception test were given. Results showed that students improved their understanding about how control systems work. None of the operating modes allowed student to develop directly in the microcontroller and debugging at the same time.

Choi [46] described the embedded control courseware and its benefits in the linear control systems course. The embedded control courseware consists of a set of lab experiments to teach students how to implement proportional, integral, and derivative controllers as C programs running on microcontrollers. Student evaluation proved that this approach was effective in teaching students to design and build embedded control solutions for solving feedback control problems. However, students lost significant time dealing with the wiring structure, connecting ports and dealing with several hardware issues. Also, the analog signal conditioning circuit used by the author lacks of precision components and the system uses a dual supply, which implies that it lacks of flexibility and portability. Also, there is no formal methodology to teach students how to use the microcontroller.

Roshan et al., [47] developed laboratory testbeds for a Mechatronics course entitled "Embedded and Real-Time Control Systems". The course was divided in a lecture component and a laboratory component. In the lecture component, the main topics of embedded systems were introduced. The laboratory component of the course was project oriented, involving several low-cost mechatronic testbeds. Students went through the design of an embedded computer system using open-architecture mechatronic testbeds and integrated development environments. Also, students developed a testbed to control the position of a 1 DOF magnetic manipulator using high level code generation tools. This approach lacked of formal procedures, the wiring structure was not organized and students had to develop a testbed each semester. The use of a high level Code Generation tool was convenient for basic projects. However, when a student had to deal with real problems he/she would have to make a custom C Code program in a low level environment, which represented a big limitation of this work. Also, this approach lacked of formal assessment instruments to assess its effectiveness.

Finally, Martí, et al., [48] presented a prototype of a laboratory experiment that was integrated in the education of embedded control systems engineers. In his paper, the authors describe how the experiment could be tailored to the needs and diverse background of both undergraduate and graduate students' education. The authors tested the performance of the system using a Real-Time Operating System (RTOS). Many experiments regarding to the controller performance using multi-tasking software were presented. The system showed excellent results and it proved to be very flexible. This approach used state-space linear systems and did not require analog signal conditioning circuits or power amplifier modules. Unfortunately, it focused only on the microcontroller problem. Also, no assessment methods were used to validate the accomplished objectives.

In conclusion, only a few of the reviewed approaches teach students how to develop digital controllers directly with microcontrollers using C code language. However, there is no formal course design to implement these tools. Finally, it is not common to find formal assessment methods to measure the effectiveness of the reviewed m approaches. Our approach is intended to align assessment methods with course content, and delivery.

Chapter 3:
PROBLEM STATEMENT & OBJECTIVES

3.1. Problem Statement

The use of microcontrollers is playing an increasingly important role in a wide range of engineering applications [49]. Specifically, the area of control systems depends significantly on the use of microcontrollers for control system implementation.

The problem addressed in this thesis is that of developing a methodology to improve students' skills for implementing digital control systems in the real world using microcontrollers. To the best of our knowledge, although there have been approaches in the area of control systems that use many high level tools to help students in the learning process, there is no formal methodology to implement the use of embedded controllers in an educational environment. Furthermore, most of these approaches do not present formal methods to assess established student learning outcomes in their proposed solution.

Our research question is: **What is the effectiveness of using an MCS to teach students to implement digital controllers in a Digital Control Systems (DCS*)* course?.**

The proposed hypothesis is that a teaching methodology using microcontrollers in the DCS course, compared to the classical high-level tools, fosters creativity and problem solving skills while providing a better knowledge on how a digital control system works in a real environment.

3.2. Objectives

This section describes the objectives that have been formulated for this work. First, a general objective for this research project is presented. According to this, the specific objectives were divided in two sections: Educational Objectives and Technical Objectives.

3.2.1 General Objective

To design and implement a teaching methodology to teach students in the DCS course to implement digital controllers using a microcontroller based control system (MCS).

3.2.2. Educational Objectives

1. Developing a set of workshops to explain students the main topics in the Embedded Control Systems implementation. These workshops will follow a pedagogy capable of aligning student outcomes, learning objectives, and assessment.
2. Developing a set of data collection instruments to measure student's outcomes of these workshops according to ABET criteria.
3. Determining if the Microprocessors course should be taken as a pre-requisite of the DCS course.

3.2.3. Technical Objectives

1. Designing and implementing a microcontroller-based system capable of reading different types of sensors, activate actuators, and reduce the risk of damage from overvoltage or wrong connections from the user.
2. Developing a set of C Code Libraries with the basic configurations for the microcontroller peripherals.
3. Providing a documentation package with user manuals, student handouts, troubleshooting guides, and tutorials. This documentation will be used for future developments and will ease student's learning process.

Chapter 4: METHODOLOGY

This Chapter presents the methodology used to develop this research. For this project, we have identified two sets of objectives: technical and educational. The methodology to achieve the technical objectives is presented mainly because it resulted in the development of an instrument that was used to accomplish the educational objectives.

4.1. Methodology to Achieve Technical Objectives

To achieve technical objectives, we developed a microcontroller based control workstation (MCS). This MCS integrates the Texas Instruments (TI) C2000 TMS320F28069 Microcontroller [42], the DRV8833 motor driver [43], and a custom made Analog Signal Conditioning Board. The MCS is enclosed in a box with all the ports needed to connect to the PICL experiments. A set of C language libraries with solutions to the most common control problems was developed. Also, this TI microcontroller uses a powerful tool called GUI Composer™ that helps students to create a Graphical User Interface (GUI) in a drag-and-drop environment. This GUI allows for watching the variables of the controller in real-time and change the controller parameters on-the-fly, if needed. The development process of the MCS was presented in the SACNAS 2014 Annual Conference [50] and is to appear in the IEEE 2015 Frontiers in Education Conference [51] and the ASEE Annual Conference [52].

4.1.1 Microcontroller Based Control System Design

A general system overview of the MCS is shown in Figure 2. The Analog Signal Conditioner is required to convert the voltage signal levels from the Quanser® analog sensors to a range that could be read by the microcontroller (0-3.3V). A DC motor driver (DRV8833) [43] was configured and used to drive the B&B system motor (SRV-02, [8]) from the PWM signals generated by the microcontroller. The Optical Encoder reads the angular position of the motor. Finally, a Timer Interrupt Unit sends a signal to the CPU indicating when the control action should be performed.

The MCS uses a C2000 TMS320F28069 microcontroller. It operates at 80MHz and is equipped with 16 Analog-to-Digital Conversion (ADC) channels with a resolution of 12 bits, 2 Quadrature Encoder read-modules (eQEP), 16 independent 32 bits PWM channels, Floating Point Unit (FPU), JTAG emulation tool, and other features that make this microcontroller ideal for high capacity digital control purposes.

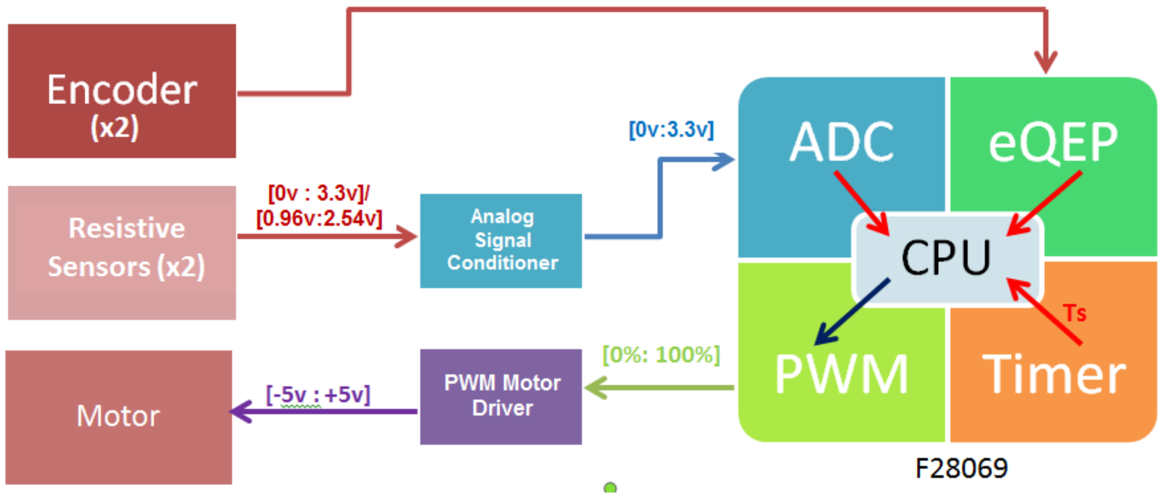


Figure 2. MCS System General Overview

4.1.2 Analog Signal Conditioning Board

The MCS was designed to work with two Quanser® analog sensors, these resistive sensors were originally designed to have a dual power supply of -12V to +12V. Using two 7kΩ bias resistors, they convert this voltage level from -5V to +5V. For this reason, a Signal Conditioner Circuit is used to convert these voltage levels to a range of 0V to 3.3V which is the operative range of the microcontroller ADC unit.

The use of negative polarization voltages represents a disadvantage because the system should use a dual voltage power supply, making it less portable and incrementing costs of production and flexibility. The proposed solution is to use a single power supply of 3.3V on the resistive sensor; this yields an output voltage range between 0.96V and 2.33V. See Figure 3.

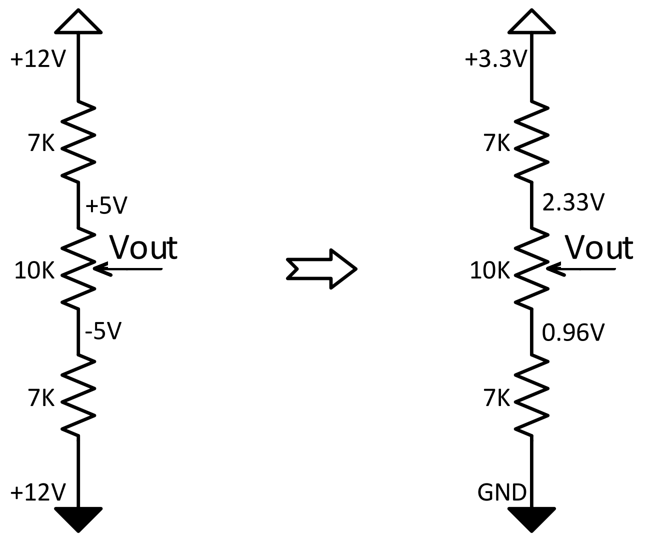


Figure 3. Quanser® Resistive Sensors Voltage Levels. Left: Quanser® Power Supply Voltage Levels. Right: Proposed Power Supply Voltage Levels.

Thus, it was necessary to amplify and add an offset to this signal to read it adequately with the microprocessor. The AD623AN instrumentation amplifier (IA) was used for this task. This amplifier may work using a single power supply and its gain is set using only one resistor R_G. We used $R_G = R_{AG} + 49.9k\Omega$. Figure 4 shows the proposed analog signal conditioning circuit design. The output voltage is given by

$$Vout = \left(1 + \frac{100k}{R_G}\right)\left(Vin - \frac{3.3R_{OFF}}{R_{OFF} + 4.87k}\right) \tag{1}$$

The desired amplifier gain and DC-offset is obtained by adjusting R_G and R_{OFF}, following the instructions provided in Workshop 2 (Refer to Appendix D, Workshop 2, Slide 8). The output equation that converts the input signal range of 0.96 to 2.33V to an output signal range of 0 to 3.3V is given by

$$Vout = 2.4(Vin - 0.9625) \tag{2}$$

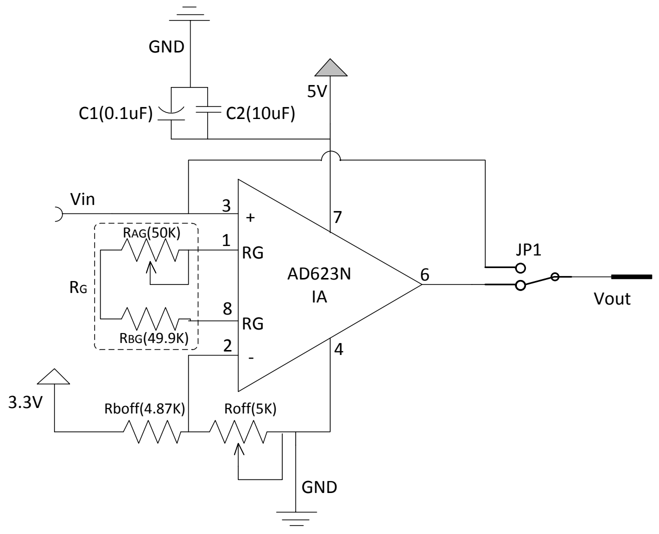


Figure 4. Analog Signal Conditioning Circuit

A voltage divider was used in the analog signal conditioning circuit to provide an offset voltage of 0.96V on the inverting input of the IA, pin 2 of the AD623N. The variable resistor R_{OFF} is used for calibration to minimize tolerance errors. The two capacitors C1 and C2 are used to isolate noise from the microcontroller and other digital circuits. Furthermore, a 3 pin jumper JP1 was used at the output to bypass the amplifier in the case when the sensors already work at the same voltage range of the microcontroller ADC and their signals do not need to be conditioned. This circuit was simulated using ORCAD Capture™ and it showed adequate performance. Figure 5 shows that the input signal (purple line) in the range of 0.96-2.54V is amplified and offset to be in a range of 0-3.3V (red line).

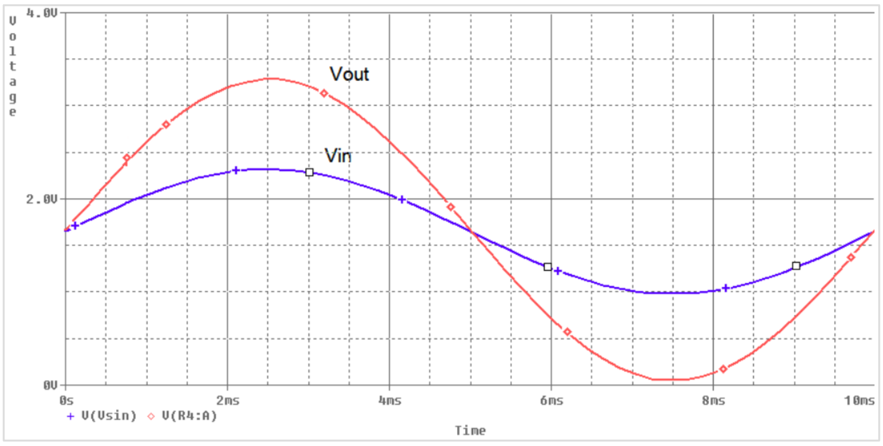


Figure 5. Signal Conditioner Simulated Results

The Vout pin is connected to the ADC pins of the microcontroller development board. This development board has all the protection circuitry required to prevent from input over-voltage or input inverse-voltage. This is accomplished using diode voltage clippers at the input of the ADC pins of the microcontroller [42].

A PCB layout was developed using Eagle®. Several requirements had to be taken into account for the PCB design, for example, conditioned signals should have a testing point for troubleshooting and path widths should be wide enough to prevent ground loop noise. A photo of the PCB prototype is shown in Figure 6.

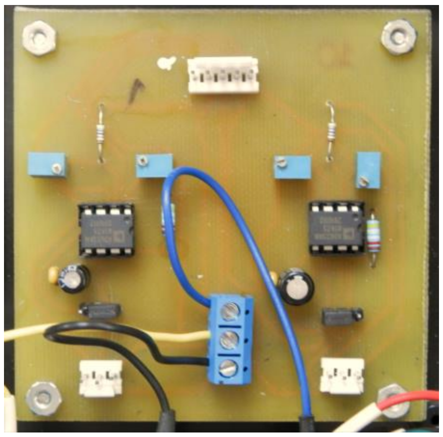


Figure 6. Photo of the Analog Signal Conditioning Board

4.1.3 System Assembly

Each of the twelve MCS stations was packed in an enclosure with a power supply port, sensor ports for the connections with the Quanser® Experiments, and test points for analog signals. This system is able to handle different situations that may appear in practice such as overvoltage and bad connections among others. Figure 7 shows photos of the final box assembly. The MCS is organized so that the student may visually identify its main modules and use them with little risk of damage.

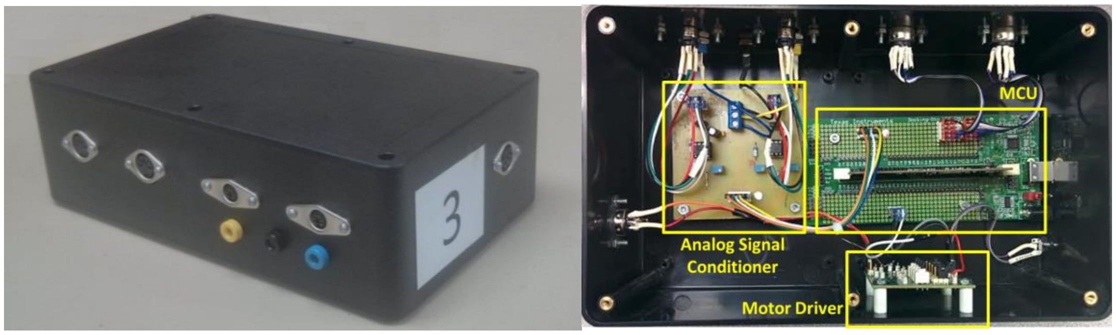


Figure 7. Photo of the final box assembly of the MCS stations. Left: External view. Right: Internal View.

4.1.4 Peripheral Configuration Libraries

Peripheral configuration procedures are presented below. The peripherals most used in the development of digital controllers include: (1) Analog-to-Digital Converter (ADC), (2) Enhanced Quadrature Encoder Unit (eQEP), (3) Enhanced Pulse Width Modulation Unit (ePWM), and (4) CPU Timer Interrupt Unit. These configuration libraries are based on the TI Control Suite™ package. This package contains some sample projects for configuring the main peripherals of C2000 microcontrollers [53]. Refer to the C2000 TMS320F28069 User's Manual for a deeper understanding of these routines [42].

4.1.4.1 ADC Unit

The TMS320F28069 has 16 independent ADC Channels with a resolution of 12 bits, each of these channels is activated by an event called *Start of Conversion* (SOC). Each SOC has its own configuration on three different items: Channel of Conversion, Trigger Source, and Window Size.

The MCS only uses 2 ADC channels (A6, A7) triggered in continuous mode. This means that every time the ADC produces a conversion, an *End of Conversion* (EOC) flag is set; this flag initiates the next conversion and so on. The window size is set to 7 clock cycles, this is the minimum amount of time that the ADC needs to produce a reliable conversion. A general scheme is shown on Figure 8.

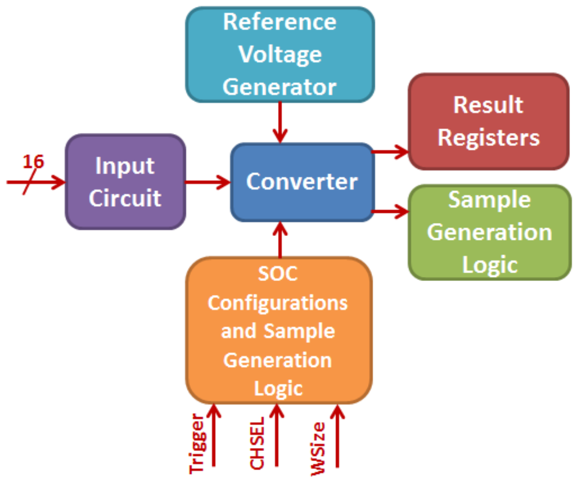


Figure 8. ADC Unit General Scheme

ADC Initialization, labeled as Code Fragment 1, is provided in Appendix A. To avoid unwanted ADC values, it is very important to wait until the End-of-Conversion (EOC) flag is set or at least wait 7 clock cycles before reading a new converted value. Once this value is acquired, it must be multiplied by the sensor sensitivity to get the B&B ball position in meters. This procedure is expressed as

$$Ball\ Position\ (m) = \ (ADCvalue - 2047)\left(\frac{0.4m}{4096}\right) \qquad (3)$$

The position sensor sensitivity for the ADC is $\frac{0.4m}{4096}$. This sensitivity is obtained by dividing the beam length (0.4m) by the number of discrete levels in the ADC unit (4096). It is required to subtract 2047 in (3) to set the zero value of the ball position at the center of the beam.

4.1.4.2 Enhanced Quadrature Encoder Unit (eQEP)

Most of the Quanser® rotational sensors use optical encoders for angle measuring. An optical encoder is a sensor that generates 2 square wave signals indicating angle and rotating direction. Figure 9 shows both signals and their meaning.

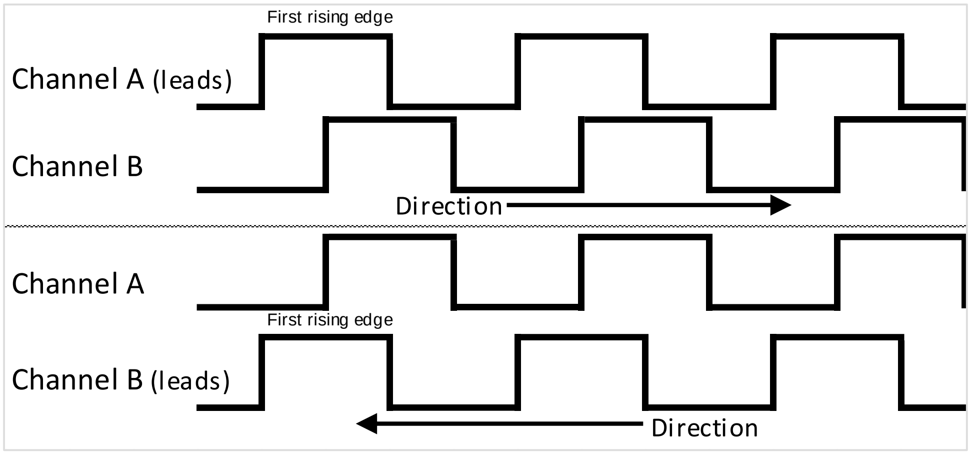


Figure 9. Quadrature Encoder Signals

Each signal has a 90 degrees phase shift from the other depending on the rotating direction. The TMS320F28069 microcontroller has 2 eQEP units capable of decoding these signals and counting the rotating steps of the sensor. Encoder input pins are located for Channel A in GPIO pins 20 and 21, and for Channel B in GPIO pins 24 and 25. The eQEP configuration procedure is provided in Appendix A, Code Fragment 2.

The encoder register value is multiplied by the encoder sensitivity to obtain the angle measure in radians. The sensitivity is obtained by dividing one revolution ($2\pi\ rad$) by the number of counts in the encoder (4096). The angle of the motor is given by

$$Motor\ Angle = \left(\frac{2\pi\ rad}{4096\ counts}\right)\ eQEP\_Count \tag{4}$$

4.1.4.3 Enhanced Pulse Width Modulation Module (ePWM)

The DRV8433 [43] motor driver was used to generate the voltage needed to move the SRV-02 motor. Changing duty cycle affects the output voltage of the motor driver, see Figure 10. Two independent PWM Channels were used for this task. To move the motor clockwise (CW), duty cycle in PWMA channel must be fixed to 100% and the duty cycle in PWMB channel determines the CW angular velocity. To move the motor counterclockwise (CCW), duty cycle in PWMB channel must be fixed to 100% and the duty cycle in PWMA channel determines the CCW angular velocity.

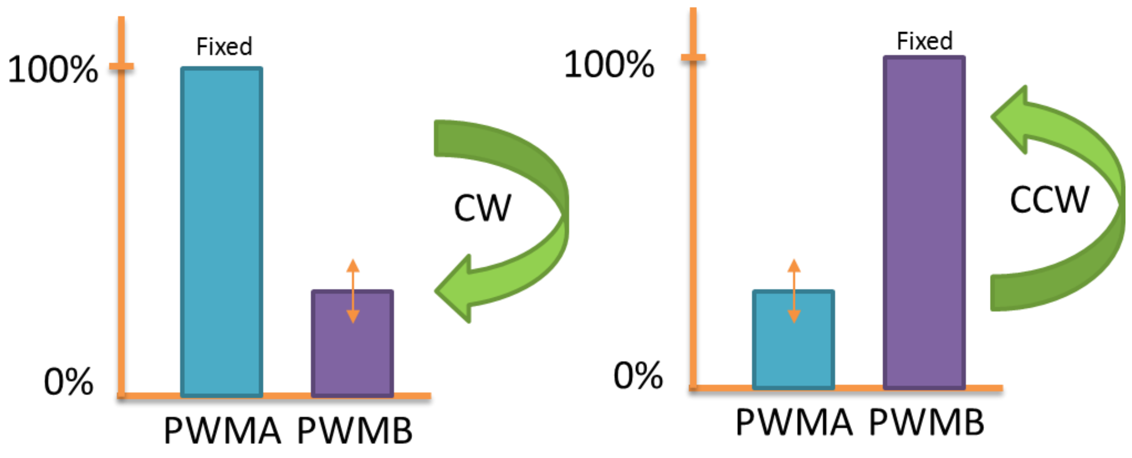


Figure 10. Effect of changing duty cycles difference on the motor speed.

The motor driver manufacturer recommends a working PWM frequency of 10 kHz. This guarantees an effective power transfer. As seen in Figure 11, each time the timer counter reaches the compare register value (OCRnA, OCRnB), a toggle on the output signal is made (OCnA, OCnB). The pins used were GPIO00 and GPIO01, corresponding to EPWM1A and EPWM1B. These pins are connected to the motor driver channels that control the H Bridge. PWM initialization function is provided in Appendix A, Code Fragment 3. The function that transforms Volts to PWM signals is shown in Appendix A, Code Fragment 4.

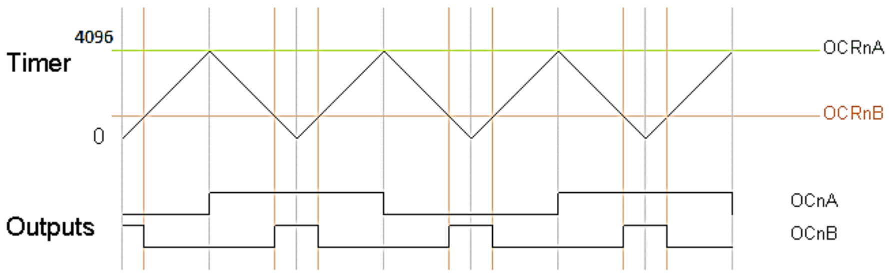


Figure 11. PWM Up-Down Count Mode Waveform.

4.1.4.4 CPU Timer Interrupt Unit

The CPU Timer Interrupt Unit is used to generate interrupts at a fixed time interval; this allows using the microcontroller in between control actions, taking advantage of all the features that it offers. When these interrupts occur, the processor executes the difference equations that realize the control action. Since the MCS has to read the sensors and update the control action on each sampling period, it is necessary to compute the control action before the next sampling (timer interrupt) occurs. In this sense, the MCS operates in real-time. This real-time operation guarantees the proper operation of the digital control system. A Timer Interrupt Unit overview is shown on Figure 12.

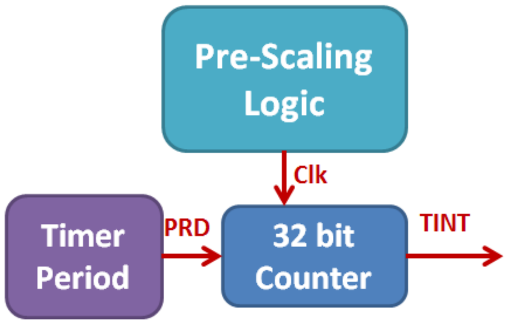


Figure 12. Timer Interrupt Unit Overview

The Pre-Scaling Logic divides the CPU frequency to generate a new clock signal that will indicate when a count event occurs. The Timer Period block establishes the maximum counting value, when the counter reaches this value, a TINT flag is set and the CPU will generate an interrupt executing a user defined subroutine. The Timer initialization procedure is provided in Appendix A, Code Fragment 5.

4.2. Methodology to Achieve Educational Objectives

This section describes the methodology that was followed to achieve the educational objectives formulated in Section 3.2.2 First, we present the participants that were the subjects of the experiments developed in this research. Then, we present the workshop design which followed Streveler's OBE framework [9] to formulate the workshops' content, assessment, and delivery. Finally, a methodology used to assess the performance of the participants in the DCS project is presented.

4.2.1 Participants

The undergraduate Electrical Engineering Program in the Electrical and Computer Engineering (ECE) Department consist of 165 credits that should be completed in a period of five years. The curriculum provides students with a general education background in mathematics, science, and humanities. The program has five areas of emphasis: Applied Electromagnetics, Communications and Signal Processing, Control Systems, Electronics, and Power. Most students in the DCS course are in their fourth or fifth year of their course plan and are specializing in the Control Systems Area.

Some of the students in this course have already taken the microcontrollers course, which is a core course in the curriculum. At this point, students have basic knowledge on the use of microcontrollers, but they do not have the required skills to create an embedded controller.

4.2.2 Workshop Design

Streveler, et al. [9], presented an approach combining Outcome Based Education (OBE) and engineering design methods to align course content (curriculum), assessment (evaluation methods), and delivery (teaching strategy) . In that approach, it is recommended to begin with the course requirements or specifications, emphasize metrics, and then prepare prototypes that meet the requirements. This work follows a *Backward Design* approach as presented by Wiggins and McTighe's [27] [54] and the *How People Learn* framework presented by Bransford, et al. [28]. In the following sections we describe the activities of the backward design approach: content, assessment, and delivery of the workshops that allowed the successful implementation of the proposed methodology.

4.2.2.1 Workshop Content

In this study, the workshop content was designed to assist participants in accomplishing ABET student Outcome E of the DCS course [55, 56]. This outcome is stated as: *Implement a digital controller using a digital computer and software*. According to the OBE framework [9], the content of each workshop was designed in four stages: (1) *Desired Outcomes*, (2) *Curricular Priorities*, and (3) *Learning Objectives*. The first two stages provided a baseline to determine the expected student outcomes and define student's *Learning Objectives* that must be accomplished.

Desired Outcomes: According to Streveler's OBE methodology [9], the first step is to identify the desired outcomes by answering three guiding questions: What do we want students to know?, What do we want students to be able to do?, and Who do we want students to be?. In other words, we must focus on student's *minds, hands, and heart* [57]. The answers to these questions provided: (1) a clear understanding of what our "ideal student should be like" after completing the course and (2) a guideline to specify our learning objectives, teaching strategy, and assessment methods.

The instructor identified the desired student outcomes for the workshops according to the laboratory experiments characteristics, the course general objectives, and project requirements. Specifically, student outcomes for the workshops were focused on the knowledge of microcontroller characteristics and knowledge about C code language. Additionally, student outcomes should modify students' behavior by enhancing self-learning skills and curiosity about how digital controllers work in a real environment. Table 2 shows the student outcome analysis that resulted from answering the guiding questions.

Table 2 - Student Outcomes Analysis

What do we want students to know?	Students should know the advantages of implementing a controller using a Microcontroller based System against versus high-level implementation tools such as Simulink or Labview.
	Students should identify the basic components in a Microcontroller based control system.
	Students should know how to use the software developing tools to program the Microcontroller.
	Students should identify the basic structures of the C Code programming language.
What do we want students to be able to do?	Students must be able to configure the main peripherals of the Microcontroller based control system.
	Students must be able to use C Code sample libraries to implement a digital controller using difference equations.
	Students must be able to use an instrumentation amplifier for conditioning an analog signal.
Who do we want students to be?	Students should be self-learners.
	Students must be courious about understanding how a microcontroller based control system works.

Curricular Priorities: Once we determined the expected student outcomes of the course workshops, the next step was to translate these characteristics into curricular priorities [9]. Following [9], we organized student outcomes in three levels of content: *Enduring Understanding*, *Important Elements to Know & Do,* and *Worth Being Familiar With*. We proposed that the *Enduring Understanding* for students was to have a clearer perspective on what the main hardware and software characteristics of a microcontroller based control system are. *Important elements to know and do* refer to the concepts and skills learned in the implementation of the controller using an embedded system. Finally, Students at the end of the course must be *Familiar With* the how microcontrollers are used in the implementation of control systems. Table 3 shows the curricular priorities for the course workshops.

Table 3- Curricular Priorities of the Digital Control Systems Course Workshops

Enduring Understanding	Implementing a Digital Controller using Microcontrollers (MCU).
	Configure the main peripherals of the Microcontroller based control system.
	Use C Code sample libraries to implement a digital controller from difference equations.
Important Elements to Know & Do	Identify the advantages of implementing a controller using a Microcontroller based System versus using high level implementing tools such as Simulink or LabVIEW.
	How to design a computer program algorithm.
	Employ a user manual to solve a particular problem of any technology.
	Understand the working principle of the different sensors in the laboratory experiments.
Worth being Familiar With	The set of tools that ease the programming and debugging process of the Microcontroller.
	Microcontroller programming.
	Main characteristics of the TI C2000 microprocessor.
	C code main operations and programming structure.

Learning Objectives: Learning objectives were established according to the outcome analysis and curricular priorities determined in the previous sub-sections. The learning objectives were written following the cognitive domain of Bloom's Taxonomy which involves knowledge and the development of intellectual skills. This includes the recognition of specific procedural, patterns, facts and concepts that serve in the development of intellectual abilities and skills [37]. In the DCS course workshops, students were expected to successfully complete the following course learning objectives:

1. <u>Identifying</u> the main differences between implementing a digital controller using MCU's and high level implementing tools (Matlab, Simulink, LabVIEW, among others).
2. <u>Identifying</u> the main components in a MCU based control system.
3. <u>Identifying</u> the main components in a MCU C Code Program.
4. <u>Explaining</u> the working principle of the different sensors in the laboratory experiments.

5. Developing a C Code program to read and write in the main peripherals of the C2000 microcontroller.
6. Developing a Graphical User Interface to read from and write to the memory of the microcontroller on-the-fly.
7. Synthetizing relevant information to develop a C Code program to implement a digital controller.
8. Testing an MCU based digital controller with the Ball & Beam experiment.

4.2.2.2 Workshop Delivery

Workshops were delivered in the PICL and were two hours long each. Students were organized in pairs and were assigned a computer workstation and an MCS workstation. The instructor delivered a tutorial where a step-by-step guidance was given according to the learning objectives of the session. The workshops presentations appear in Appendix D.

Workshops were delivered alongside the development of the course project. According to this, students were evaluated in their progress reports on how they integrated the use of the MCS in their projects. Three different workshops were designed to achieve the learning objectives:

- The first workshop targeted learning objectives 1, 2, and 3 (the fundamental elements of embedded control systems). Students recognized the main differences between implementing a digital controller using an MCS and using high level tools. Also, in this workshop students learned basic operations in a C code programming through the use of the Code Composer Studio Development Software. Finally, students implemented a basic C Code blinking led program which allowed them to familiarize with the Code Composer Studio Environment.
- The second workshop targeted learning objectives 4, 5, and 6 that address the configuration and provide a hands-on experience with the microcontroller stations. Students learned how to use the microcontroller peripherals to read signals from sensors and to control the motor driver. Students were expected to learn about the working principle of each sensor of a Ball & Beam system and to read them using the microcontroller. Then, students learned to translate the signals coming from the

peripherals to the international system units (meters, radians, Volts). Finally, students created a GUI to view the variables measured by the sensors.

- The third workshop targeted the last two learning objectives by implementing a real controller using difference equations and the Texas Instruments GUI Composer. This workshop synthetized the concepts learned in the two previous workshops by implementing a digital controller using C Code. Specifically, students were expected to learn how to transform a z-domain transfer function to a difference equations and how to convert this difference equation into a C Code program to control a servomechanism.

4.2.2.3 Workshop Assessment

Assessment activities were designed according to the learning goals established for each workshop. To achieve these, we used Bloom's Taxonomy [37] to define course objective, evaluate how each item was satisfied and then determine assessment activities for each course objective. This guaranteed an alignment between course content and objectives [9]. Among the proposed assessment activities were: diagnostic examinations, pre and post-tests, and oral exams. A description of each assessment activity is presented below.

Diagnostic Test: The objective of this test was to explore students' background on microcontrollers, programming languages, and control systems. It was given before the first workshop. A sample of the diagnostic test is provided in Appendix B. Results obtained from the diagnostic test are presented in Section 5.2.1.1.

Pre and Post-Test: Students received a pre-test (before the workshop) and one post-test (after the workshop) with identical questions. These tests measured student outcomes' fulfillment on each workshop. The results obtained in pre and post-tests indicate that the workshops were effective. Examples of the pre and post-tests are included in Appendix B. Results obtained from the pre and post-test are presented in Section 5.2.1.2.

Oral Exam: At the end of the course project, students took an oral exam to assess how effectively they implemented the controller using both methods (Simulink and Microcontroller). A rubric was designed and used to assess this oral exam. A sample of the rubric is provided in Appendix C. Results obtained from the oral exam are presented in Section 5.2.1.3.

4.2.3 Methodology Assessment

To assess the contribution of this work, we evaluated students' performance in their term project and compared to those students who did not received the workshops in previous years. For this, we developed a set of rubrics written according to the cognitive domain of Bloom's Taxonomy and ABET criteria [37] [55] (See Appendix C). Final project reports provided enough information about students' fulfillment of the project outcomes for those students that took the course in previous years. Specifically, we assessed the ability of students to implement and validate a digital controller using Simulink and Microcontrollers. Participants were divided in three categories:

(1) Students from 2013 fall semester who implemented and validated digital controllers using only Simulink.

(2) Students from 2013 fall semester who implemented and validated digital controllers using Simulink and additionally implemented their controller on a microcontroller voluntarily. These students were not trained in the use of microcontrollers for digital control implementation.

(3) Students from the 2014 fall semester who were required to implement and validate digital controllers using both Simulink and MCS. This group was trained with the workshops developed in this work.

All students in these categories were assessed through their project reports. A second evaluator assessed a representative sample from each group to guarantee inter-rater reliability and that no arbitrary evaluations were made. Then, a correlating factor between the samples assessed by us and samples assessed by the second evaluator was computed [58]. The correlating factor is given by

$$Correl(X,Y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}} \qquad (5)$$

where x and y are the scores given by each evaluator and $\bar{x}$ and $\bar{y}$ are the average scores given by each evaluator. A correlating factor greater than 0.8 will guarantee inter-rater reliability and will indicate that there is no arbitrary preferences in the validation process [59]. Sample size n_0 for large size populations is defined according to Yamane's simplified formula [60]

$$n_0 = \frac{N}{1 + Ne^2} \tag{6}$$

where N represents the total population and e is the level of precision. However, for small size populations, the sample size n can be adjusted as

$$n = \frac{n_0}{1 + \left(\frac{n_0 - 1}{N}\right)} \tag{7}$$

which, in general, will be smaller than n_0.

Equations (6) and (7) were used to determine the sample sizes for the inter-rater reliability evaluations. Results obtained from the methodology assessment are presented in Section 5.2.2.

Chapter 5:
RESULTS AND ANALYSIS

This Chapter presents the results obtained from this work. First, we present the results obtained in the implementation of a digital controller using the MCS stations. Then, we analyze the pedagogical results obtained from the workshop design process and assessment activities.

5.1 Digital Controller Implementation

This section describes the results obtained in the implementation of a digital controller for the B&B system using the MCS. First, we present the design of a Lead-Lag digital controller and explain how to transform it to a computer program using difference equations. Then, we describe the digital controller implementation using the MCS and a graphical user interface called GUI Composer™. Finally, system documentation and user manuals are presented.

5.1.1 Controller Design

A Lead-Lag Controller for the B&B system was implemented using the MCS. Figure 13 shows a block diagram of general DCS for this controller. The blocks inside the dotted rectangle represent the difference equations performed by the microcontroller, while the blocks outside the dotted rectangle represent the dynamical system of the B&B system. The variables and transfer functions shown in the block diagram are defined by:

$X(s)$-Ball Position
$\theta(s)$-Motor angle
$Vm(s)$-Voltage applied to motor
$R(s)$-Reference signal (Master Ball)
$H_1(s), H_2(s), H_3(s)$ - B&B sensors transfer functions
$A(s), B(s), E(s)$-Auxiliary signals
$G_{c1}(z)$-Ball position controller (Outer Loop)
$G_{c2}(z)$-Motor angle controller (Inner Loop)
$G_m(s)$-Motor transfer function$\left(\frac{\theta(s)}{Vm(s)}\right)$
$G_p(s)$-Slave ball transfer function$\left(\frac{X(s)}{\theta(s)}\right)$.

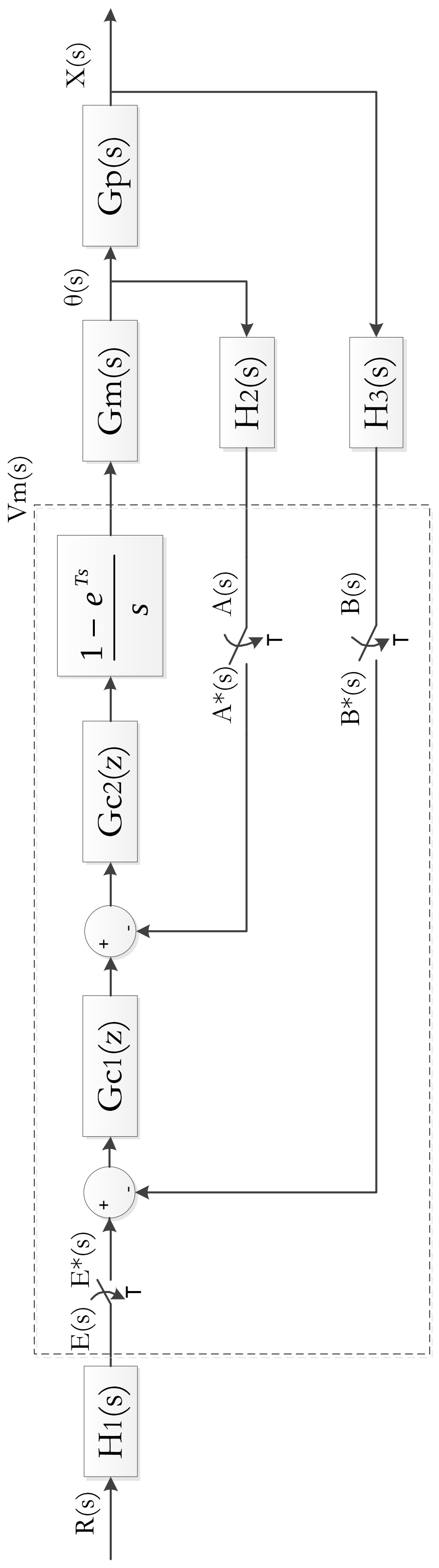


Figure 13. Ball & Beam controller System Block Diagram.

System Transfer Functions were derived in *Quanser's Student Handout* [61]. Controller C code program appears in Appendix A, Code Fragment 6. A sampling period of 30ms was selected according to Shannon's sampling theorem [4]. Performance specifications were established according to experience in previous courses. The Ball Position Controller was required to have an overshoot under 30% and a settling time under 5 seconds. According to these specifications, a Lead-Lag compensator was designed using discrete-time root-locus methods. The Lead-Lag controller has two poles and two zeros. A pole-zero pair corresponds to the Lead compensator and the other pole-zero pair corresponds to the Lag compensator. The Lead compensator has its zero to the right of its pole in the z-plane while the Lag compensator has its pole to the right of its zero; but very close to each other. The designed Lead-Lag Controller is given by

$$G_{c1}(z) = \frac{\theta_{ref}(z)}{X_{error}(z)} = 82.26\frac{(z-0.986)(z-0.995)}{(z-0.6287)(z-0.999)} \quad \textbf{(8)}$$

Where $\theta_{ref}(z)$ is the Z-transform of the output of the controller and $X_{error}(z)$ is the Z-transform of the error between the reference position and the actual position of the ball. To obtain the difference equation, it is necessary to apply a Z-transform property which states that $Z^{-1}\{z^{-n}\theta_{ref}(z)\} = \theta_{ref}[k-n]$. This is the *Time-Shift* property and states that whenever a z^{-n} factor multiplies a Z-transform, it is equivalent to an *n-samples* shift in the time domain. This may be interpreted as *n* register shifts in a microcontroller program. The difference equation performed by the controller is

$$\begin{aligned}\theta_{ref}[k] = 82.86x_{error}[k] - 164.1457x_{error}[k-1] + 81.2915x_{error}[k-2] \ldots \quad \textbf{(9)}\\ +1.6277\theta_{ref}[k-1] - 0.6281\theta_{ref}[k-2]\end{aligned}$$

The angular Position Controller was chosen to be a proportional controller with *K*=4.7. A reference angle $\theta_{ref}[k]$ is obtained by performing (9). The output voltage applied to the motor is

$$V_m[k] = 4.7\left(\theta_{ref}[k] - \theta[k]\right) \quad \textbf{(10)}$$

Once the difference equations were obtained and all the peripherals were configured, a control action should be performed each time the CPU generates a timer interrupt. This allows the digital controller program to run in real-time. The values of all the registers in the controller are updated every time the difference equations are executed. The controller implementation via a C Code program is provided in Appendix A, Code Fragment 6.

5.1.2 GUI Implementation

One of the most important features of the Texas Instruments® C2000 microcontrollers and Code Composer Studio™ is the debugging tool. GUI Composer™ is an extension of this software. This tool allows developers to create Graphical User Interfaces (GUI) to access directly to the microcontroller registers in real-time. Once the student develops the software, he or she can create a Graphic Panel where the defined variables can be watched in real time and parameters may be modified on-the-fly. The student is able to change zeroes, poles, and all the controller parameters. Three sample GUI's were created to be used as templates for other projects: SRV-02 Proportional Controller, SRV-02 Lead-Lag Controller, B&B Lead-Lag Controller. The B&B Lead-Lag controller performance using a GUI Composer Interface is shown in the left side of Figure 14. The blue line is the reference signal, the orange line is the actual ball position in response to the reference signal. Additionally, the controller performance using the microcontroller was compared against the performance obtained by implementing the same controller using Simulink (right side of Figure 14). Results show that both controllers have similar performance.

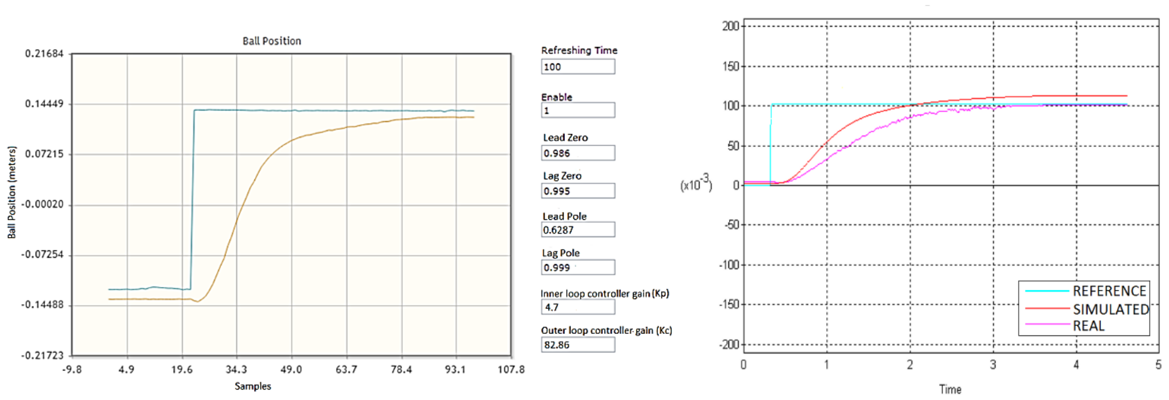

Refreshing Time: 100
Enable: 1
Lead Zero: 0.986
Lag Zero: 0.995
Lead Pole: 0.6287
Lag Pole: 0.999
Inner loop controller gain (Kp): 4.7
Outer loop controller gain (Kc): 82.86

Figure 14. Ball & Beam Lead-Lag Controller Performance. Left Side: MCS Implementation. Right Side: Simulink Implementation

5.1.3 System Documentation

The MCS is supported by a documentation package that includes a user manual, power point presentations, assembly instructions, and CAD drawings of the MCS. This documentation will help students in the development process and will provide the tools needed to replicate the system for future projects. The documentation package is provided in Appendix E.

5.1.3.1 User Manual

This document contains the MCS electrical specifications, a general system explanation, connection diagrams, quick installation instructions, and a testing and troubleshooting section. The User Manual is provided in Appendix E.

5.1.3.2 Power Point Presentations

Power Point presentations were used to deliver the workshops. Presentations content was determined according to the pedagogical design presented on Section 4.2.2. These presentations are provided in Appendix D.

5.1.3.3 Assembly Instructions and MCS CAD Drawings

These documents contain specific instructions about how to fabricate an MCS based on *Solidworks*® CAD drawings. Additionally, it contains a description of the electrical components and connectors that may be used. This information may be used in the future to develop a commercial product of pedagogical interest that may be used to improve the learning process in a higher education setting.

5.2 Pedagogical Results

This section describes the experimental results of the workshops that were designed according to the Streveler's OBE methodology [9]. First, results obtained from workshops assessment activities are presented and analyzed. Then, in the methodology assessment section, we evaluate the impact of this work on student outcomes on the DCS course project. Students' performance during the course project was compared against performance of those students from previous years, when the workshops were not available.

5.2.1 Workshop Assessment

Assessment activities were designed considering the learning objectives and student outcomes established in the previous analysis to guarantee alignment between the workshop content and delivery. At the beginning of the semester, students received a diagnostic test that explored their background in some relevant courses. Also, pre-test and post-test were administered for each workshop to determine learning gains. All the activities in this research were designed following guidelines for the protection of human subjects and approved by the UPRM Institutional Review Board (IRB), refer to Appendix F for the authorization letters. Results from these activities are presented in the following sections.

5.2.1.1 Diagnostic Test

The objective of this test was to explore student's background on microcontrollers, programming languages, and control systems before each workshop. The questionnaire used appears in Appendix C. The DCS course started with a population of 28 students. Figure 15 shows that most of students took the microprocessors course (INEL 4206). However, most of them got a B or C on the course. This indicated that special attention must be paid to the first workshop.

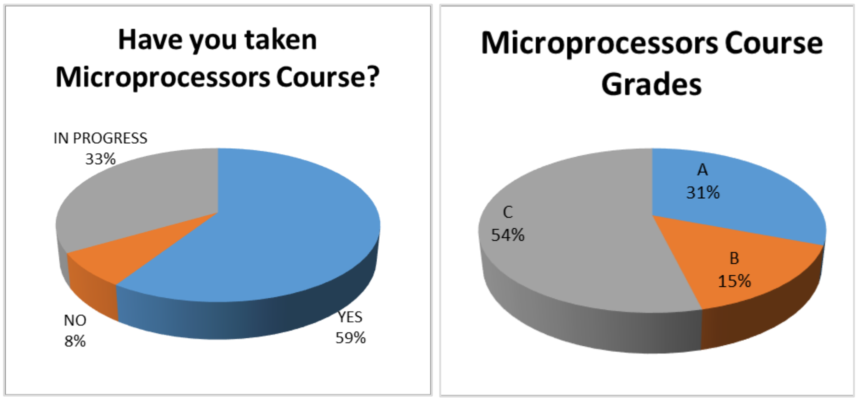


Figure 15. Diagnostic Test Item 1

Those who already took the microprocessors course were asked to specify, in a scale from zero to five, their skills using microcontroller peripherals. Figure 16 shows the diagnostic test results. These values were relatively low. This fact was useful to emphasize some of the topics covered in the second workshop.

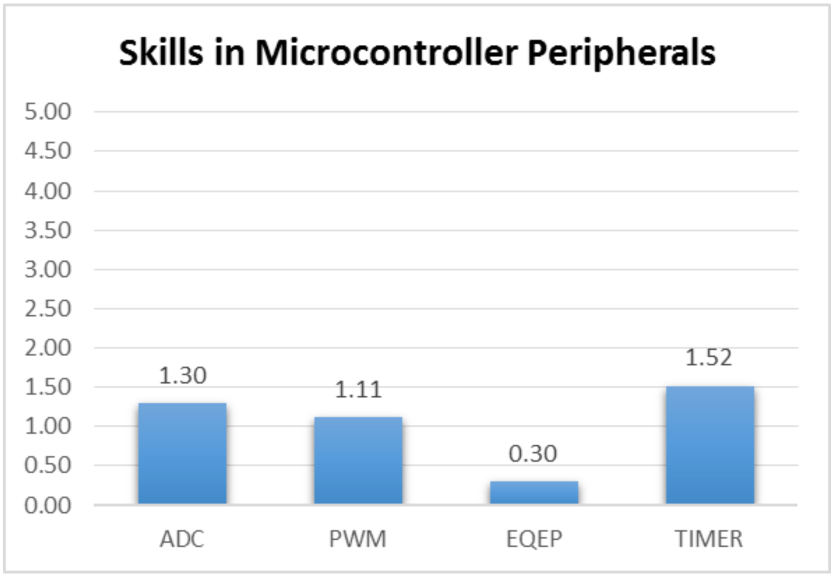


Figure 16. Diagnostic Test Item 5

A relevant fact about students is their background in programming courses. Figure 17 indicates that all students have passed the Programming Algorithms course (INGE 3016) as a pre-requisite. This implies that students may be expected to be able to complete the assigned programming tasks.

One of the questions in the diagnostic test was about students' experience using different programming languages. Students were asked to specify in a scale of five their experience using some of the most important languages used in Electrical Engineering. The best-known programming language was Matlab, followed by C and Assembly language (see Figure 18).

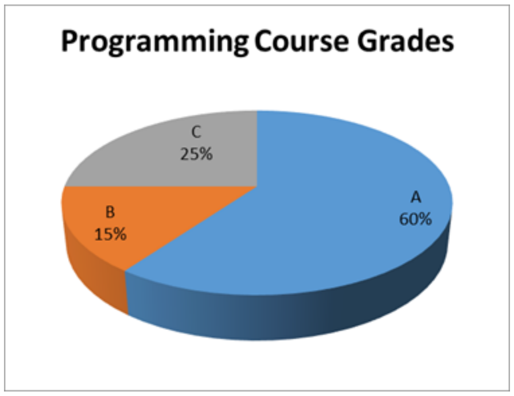


Figure 17. Diagnostic Test Item 3

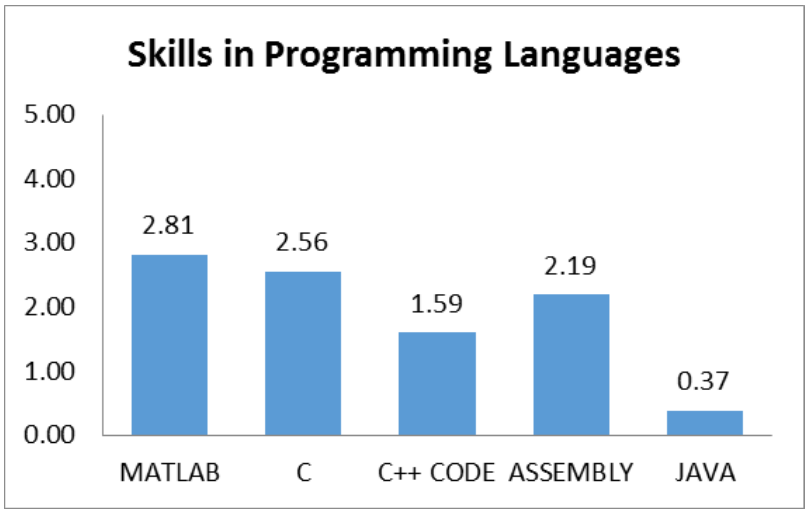


Figure 18. Diagnostic Test Item 4

Although most of the students had passed the microprocessors course, they do not have good understanding on peripheral configuration. Although, all of the students had passed the programming course; they do not have good skills on the most common programming languages taught in the department. These results were a baseline to emphasize some of the topics covered in the workshops. The following subsection shows the results obtained in workshops pre and post-test.

5.2.1.2 Pre-Test and Post-Test

Students took three workshops, two hours long each, in the laboratory. They received a pre-test before and a post-test after each workshop. These tests are provided in Appendix B. Tests results were analyzed by comparing students' performance before and after taking the workshop to determine learning gains. Our quantitative variables were the *Score Gain* (SG), defined as the difference between average scores on each question, and the *Average Normalized Gain* (NG), defined as:$NG = 100SG/(Q - P)$, where Q is the question value and P is the average score obtained by the group in the pre-test question [62]. An *NG* score greater than or equal to 30% is deemed satisfactory according to Hake [63]. He defined this value according to a pedagogical study that involved 6,000 students in 62 courses. Data suggested that a *NG* of 30% was the lower bound of what he called *"medium normalized gain"*. Results above this value could be considered as acceptable.

Workshop 1 - basic knowledge that students need to know about embedded control systems (objectives 1, 2 and 3). A total of 27 students took the workshop. One student was absent. Table 4 summarizes the average results obtained from test 1. Objectives 1 and 2 were accomplished; however, objective 3 showed a small *NG* value. This means that although the goal is accomplished, students find it difficult to identify the main components in an embedded control C code program. The average scores obtained in pre and post-test were added to obtain a total pre and post-test score for the workshop 1. Using these totals, the total SG and NG values were computed for the whole workshop. These appear in the last column of Table 4.

Table 4. Pre-Test and Post-Test analysis for Workshop 1

Question	**1**	**2**	**3**	**4**	**Total**
Objective	**1**	**2**		**3**	
Value	6	4	2	8	20
Pre-Test	2.67	0.58	0.33	2.92	6.5
Post-Test	5.25	2.88	1.54	4.50	14.16
SG	2.58	2.29	1.21	1.58	7.67
NG	78%	67%	73%	31%	57%

Workshop 2 - Configuration of the main peripherals of the C2000 microcontrollers (objectives 4, 5, and 6). A total of 28 students took this workshop. Table 5 shows the results for

pre and post-test for workshop 2. Results show that the *NG* goal was accomplished. Objective 6 was evaluated in the oral exam and is analyzed in the next Section.

Table 5. Pre-Test and Post-Test analysis for Workshop 2

Question	**1**	**2**	**3**	**4**	**5**	**Total**
Objective	**4-5**			**5**	**4**	
Value	5	5	2	5	3	20
Pre-Test	2.52	1.26	0.11	0.48	0.26	4.63
Post-Test	3.59	2.70	1.52	1.85	1.11	10.78
SG	1.07	1.44	1.41	1.37	0.85	6.15
NG	43%	39%	75%	30%	31%	40%

Workshop 3 - The use of difference equations to implement a digital controller for the B&B system (objective 7 and 8). A total of 16 students took this workshop. By this time, twelve students had dropped the course. Table 6 summarizes the obtained results. Objective 7 was assessed in all questions. Objective 8 was evaluated in the oral exam and is analyzed in the next Section.

Table 6. Pre-Test and Post-Test analysis for Workshop 3

Question	**1**	**2**	**3**	**4**	**Total**
Objective	**7**				
Pre-Test	2.00	3.25	2.63	2.63	9.88
Post-Test	3.00	3.50	5.63	5.25	17.38
SG	1.00	0.25	3.00	2.63	7.49
NG	50%	33%	89%	78%	74%

This workshop revealed to be the most effective with a total mean *NG* value of 74%. Classroom observations indicated that students were motivated to keep using microcontrollers in their future developments. Results obtained from the three workshops pre and post-tests demonstrate that the workshops were effective.

5.2.1.3 Oral Exam

A set of rubrics was developed to evaluate students in the Oral Exam. These rubrics were written according to the cognitive domain of Bloom's Taxonomy and ABET criteria [37, 64] (See Appendix B). Objective 6 was evaluated by the instructor when students developed a GUI interface using Code Composer Studio and GUI Composer. All of the students completed the requirements of this task. Objective 8 was evaluated by the instructor during the oral exam. All students developed the B&B controller using the MCU and the objective was accomplished.

5.2.1.4 Microprocessors Course Analysis

Average scores in pre and post-tests were analyzed to determine if the Microprocessors course should be taken as a pre-requisite of the DCS course. A total of 27 students were divided in three categories:

(1) 16 students who already took the Microprocessors course,

(2) 2 Student who did not take the Microprocessors course,

(3) 9 Students who were taking the Microprocessors course at that moment,

Pre-test average scores, presented in Figure 19, show that students who already took the course were better prepared for the workshops than those who have not taken it or were taking it at that moment. Additionally, students who took the course obtained higher post–test scores than the other students. The error bands shown at the top of each bar indicate the standard deviation of the scores of the corresponding bar. These results may suggest that students who take the Microprocessors course tend to have better performance in the workshops than those who did not. However, the sample is quite small to be conclusive.

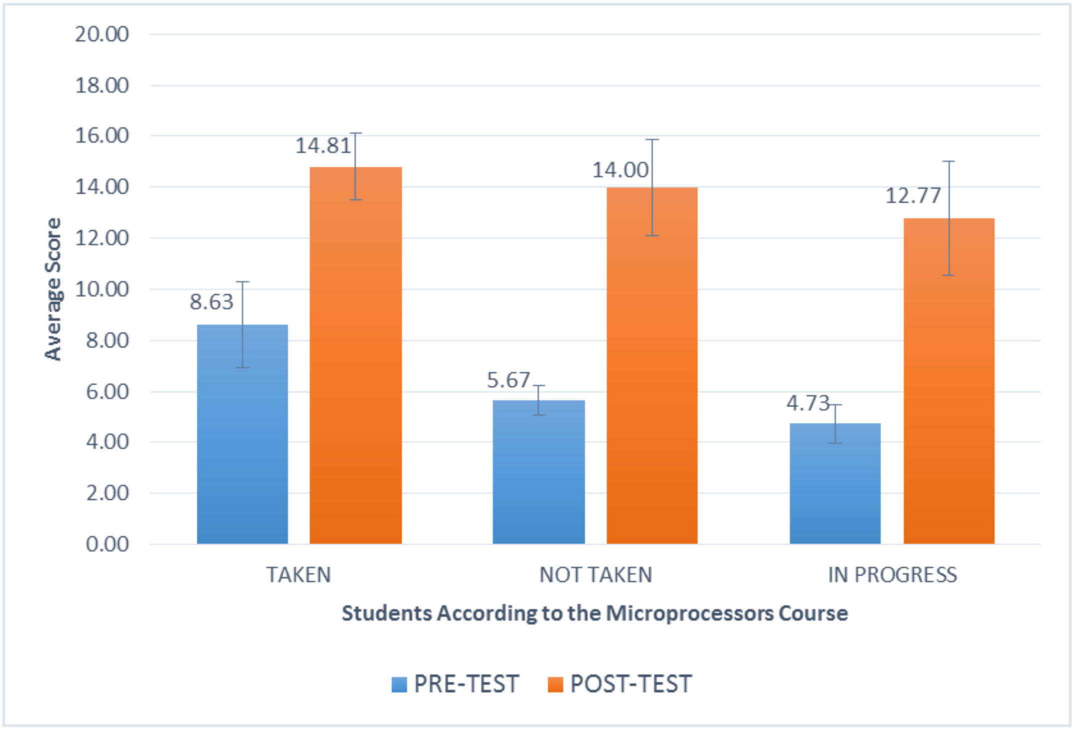


Figure 19. Analysis of the Pre and Post-Test with Respect to the Microprocessors Course.

5.2.2 Methodology Assessment

To assess the methodology, the final project reports were used. Final project reports were assessed according to stablished rubrics based on course outcomes. These are provided in Appendix C. A total of 18 reports were assessed, divided in three groups as mentioned in Section 4.2.3:

- 6 reports of student teams from 2013 who only used Simulink (14 students),
- 4 reports of student teams from 2013 who used Simulink and used the microcontroller voluntarily (9 students),
- 8 reports of student teams from 2014 who used both Simulink and the MCS (16 students).

Reports were assessed in a scale of three different values: Good (80-100), Acceptable (70-79), and Insufficient (0-69). Each team was evaluated according to five main course outcomes:

- Punctuality (Outcome 7),
- Design of Controllers (Outcome 3),

- Simulations (Outcome 8),
- Controller Implementation (Outcome 4):
    - Using Simulink,
    - Using Microcontrollers,
- Controller Performance (Outcome 5):
    - Using Simulink,
    - Using Microcontrollers.

Average results of the assessment analysis are presented in Figure 20. The error bands shown at the top of each bar indicate the standard deviation of the scores of the corresponding bar. Those students who only used Simulink (Orange Bars) had excellent performance on punctuality, design, and simulations. However, they had insufficient results on controller implementation and performance. This is because they did not accomplished the microcontroller component.

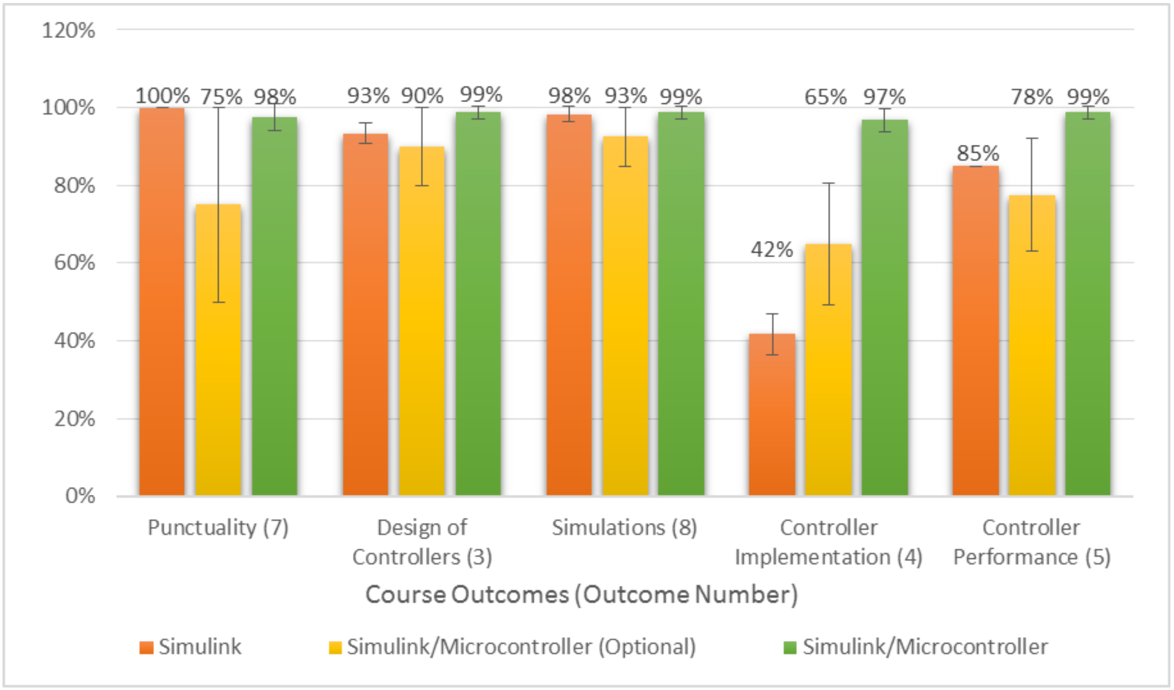


Figure 20. Outcome Assessment Compared to Previous Years

However, those students who used Simulink and used the microcontroller voluntarily (yellow bars) had an inferior performance in all outcomes. These students did not receive any training on the use of microcontrollers for digital control implementation. A deeper analysis on project reports revealed that students could not focus on the project because they spent too much of their time dealing with the microcontroller configuration and hardware connections. In fact, some students could not finish their project and thus, did not pass the course. These students repeated the course in 2014 and had an average performance over 90% on the project using the MCS and attending the workshops.

Finally, those teams who used Simulink and the MCS (green bars) yielded the best performance. These were students who received the workshops developed in this work and learned about the use of microcontrollers for digital control implementation.

To guarantee inter-rater reliability, a second evaluator assessed a representative sample of 9 teams: 3 for the first group, 2 for the second and 4 for the third group [60] [65]. These sample sizes were obtained from (6) and (7) presented in Section 4.2.3. The results of this evaluation was compared against the results obtained by us. A mean correlating factor of 0.93 was obtained [58]. According to Jonsson et al., this indicates that no arbitrary considerations were made in the evaluation of project reports [59].

The methodology assessment revealed that those students who were trained on the use of microcontrollers for digital control implementation have better performance that those who were not. This indicates that the DCS course was improved by enhancing the performance of students in the project. In the past, students were not able to accomplish project outcomes related to the microcontroller implementation.

Chapter 6:
CONCLUDING REMARKS & IMPLICATIONS

Only two of the reviewed approaches [45, 48] teach students how to develop digital controllers directly with microcontrollers using C code language. However, there is no formal course design to implement these tools. Finally, it is not common to find formal assessment methods to measure the effectiveness of educational approaches. Our work demonstrated to be effective by aligning assessment methods with course content and delivery. The contributions of this work to the area of teaching DCS are listed below:

1. A formal methodology to teach students the use of microcontrollers in the Digital Control Systems Course.
2. A set of twelve Embedded Control Stations available to be used by the students in future projects.
3. A validated knowledge on how to implement a new tool to develop a course project.
4. A set of rubrics that will ease the assessment of the DCS course projects according to the ABET criteria.
5. A development documentation package for the MCS stations that includes assembly instructions, blueprints and a troubleshooting guide. This documentation will allow others to replicate the MCS stations.
6. A set of C Code peripheral configuration libraries that will ease the implementation of embedded controllers.
7. A documentation package that includes Power Point presentations, student handouts, and user manuals. This documentation will support designed workshops and will help students in the implementation process of their projects using the MCS.
8. A set of assessment tools that will help instructors to measure students' outcomes for future microcontroller workshops in the DCS course.
9. A set of pending publication to appear in the following conferences:

- American Society for the American Society for Engineering Education (ASEE) 2015 Annual Conference (Paper Accepted).
- A publication pending in the Frontiers in Education 2015 Conference (Abstract Accepted).
- A poster presentation in SACNAS 2015, Los Angeles, CA (Presented).

A set of workshops was developed following an outcome-based education and a backward design approach. Pre and post-test confirmed that learning objectives were accomplished and the workshops were effective. The DCS course was improved by enhancing student outcomes accomplishment in the implementation component of the project.

A microcontroller-based system was developed for the workshops. Experimental results indicated that the system has the same performance compared to the known classic methods (Simulink, LabVIEW). Also, the MCS demonstrated to be well designed since it was an effective tool for implementing the digital controllers of the student projects. In addition, the MCS represents a multi-purpose platform that can become a commercial product of pedagogical interest that may be used to improve the learning process in a higher education setting.

Workshop experiences provided students with a good understanding of embedded control systems. The process of programming the controller provided students a deeper understanding of how digital control systems are implemented in real world through embedded systems. Also, using Texas Instruments® tools helped in the debugging process. In the past, digital controllers were implemented using microcontrollers, but there were no formal debugging methods for troubleshooting.

This methodology could be used in the future to implement a new tool in other courses in the engineering discipline by following these steps:

1. Define content (Section 4.2.2.1):
    a. Analyze the desired student outcomes,
    b. Identify curricular priorities,
    c. Specify Learning Objectives
2. Define workshop delivery according to the learning objectives (Section 4.2.2.2),

3. Define assessment according to the delivery on each workshop and the learning objectives (Section 4.2.2.3).

The following directions for future work that could further improve the current state of the presented work were identified:

1. To explore the applicability of this methodology in the implementation of a new tool in other courses in the engineering discipline.
2. To develop an integrated board with all the components and connectors to drive several laboratory experiments. This board could become an educational product with commercial value.
3. To implement this methodology and the MCS in the course of Linear Systems Analysis and other courses in the control systems area.

Bibliography

[1] G. F. Franklin, M. L. Workman y D. Powell, Digital control of dynamic systems, Addison-Wesley Longman Publishing Co., Inc., 1997.

[2] G. H. Hostetter, Digital Control System Design. Holt, Rinehart and Winston, Inc., New York, New York, 1988.

[3] B. C. Kuo, Digital Control Systems (Holt, Rinehart and Winston, NY), 1980.

[4] C. L. Phillips y H. T. Nagle, Digital control system analysis and design, Prentice Hall Press, 2007.

[5] E. De Graaf y A. Kolmos, «Characteristics of Problem-Based Learning,» *International Journal of Engineering Education,* vol. 19, nº 5, pp. 657-662, 2003.

[6] Q. Consulting, «Quanser Consulting Website-Ball & Beam Experiment,» 2015. [En línea]. Available: http://www.quanser.com/Products/ball_beam.

[7] G. Beauchamp y M. Vélez, «A Process Instrumentation and Control Laboratory,» *ASEE Annual Conference Proceedings,* 1995.

[8] Quanser Consulting, «Quanser Consulting Website-Rotary Servo Base Unit,» 2015. [En línea]. Available: http://quanser.com/Products/rotary_servo.

[9] R. Streveler, K. Smith y M. Pilotte, «Aligning Course Content, Assessment, and Delivery: Creating a Context for Outcome-Based Education,» de *Outcome-Based Education and Engineering Curriculum: Evaluation, Assessment and Accreditation*, K. Mohd Yusof, S. Mohammad, N. Ahmad Azli, M. Noor Hassan, A. Kosnin and S. K, Syed Yusof (Eds.)Hershey, Pennsylvania: IGI Global, 2012.

[10] R. M. Felder y R. Brent, «The ABC'S of Engineering Education ABET, Bloom's Taxonomy, Cooperative Learning and so on,» *Proceedings of the 2004 American Society for Engineering Education Annual Conference,* p. 1, 2004.

[11] M. Milner-Bolotin, A. Kotlicki y G. Rieger, «Can students Learn from Lecture Demonstrations,» *J Coll Sci Teach,* vol. 36, pp. 45-49, 2007.

[12] M. J. Prince y R. M. Felder, «Inductive Teaching and Learning Methods:Definitions, Comparisons, and Research Bases,» *Journal of Engineering Education,* vol. 95, nº 2, pp. 123-138, 2006.

[13] R. P. Hesketh, S. Farell y C. S. Slater, «An Inductive Approach to Teaching Courses in Engineering,» *2003 ASEE Annual Conference,* 2003.

[14] R. M. Felder y L. K. Silverman, «Learning and Teaching Styles in Engineering Education,» *Engineering education,* vol. 78, nº 7, pp. 674-681, 1988.

[15] M. A. Albanese y S. Mitchell, «Problem-based learning: A Review of Literature on its Outcomes and Implementation Issues,» *Academic medicine,* vol. 68, nº 1, pp. 52-81, 1993.

[16] W. L. Bateman, Open to Question. The Art of Teaching and Learning by Inquiry., ERIC, 1990.

[17] J. Kinkead, «Learning through inquiry: An overview of undergraduate research,» *New directions for teaching and learning,* vol. 12, nº 93, pp. 5-18, 2003.

[18] D. Boud y G. Feletti, The Challenge of Problem-Based Learning, Psychology Press, 1998.

[19] S. M. Loyens, P. Kirschner y F. Paas, «Problem-based Learning,» *APA Educational Psychology Handbook,* vol. 3, 2011.

[20] D. E. Allen, R. S. Donham y S. A. Bernhardt, «Problem-Based Learning,» *New Directions for Teaching and Learning, Wiley Online Library,* vol. 2011, pp. 21-29, 2011.

[21] J. E. Mills, D. F. Treagust y others, «Engineering education—Is problem-based or project-based learning the answer?,» *Australasian Journal of Engineering Education,* vol. 3, pp. 2-16, 2003.

[22] A. Aranzabal, «"Project Based Learning" approach to teach gas and steam power systems,» *@ tic. revista d'innovació educativa,* nº 13, pp. 138-148, 2014.

[23] C. K. Riesbeck, «Case-based teaching and constructivism: Carpenters and tools,» *Constructivist learning environments: Case studies in instructional design,* pp. 49--61, 1996.

[24] L. Alfieri, P. J. Brooks, N. J. Aldrich y H. R. Tenenbaum, «Does discovery-based instruction enhance learning?,» *Journal of Educational Psychology,* vol. 103, nº 1, p. 1, 2011.

[25] J. Rhem y S. a. M. M. Simkins, Just-in-time teaching: Across the disciplines, across the academy, Stylus Publishing, LLC., 2010.

[26] 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.

[27] G. P. Wiggins y J. McTighe, Understanding by design, ASCD, 2010.

[28] J. D. Bransford, A. L. Brown, R. R. Cocking y others, How people learn, Washington, DC: National Academy Press, 2000.

[29] K. Gavin, «Case Study of a Project-Based Learning Course in Civil Engineering Design,» *European Journal of Engineering Education,* vol. 36, nº 6, pp. 547--558, 2011.

[30] R. M. Crespo, J. Najjar, M. Derntl, D. Leony, S. Neumann, P. Oberhuemer, M. Totschnig, B. Simon, I. Gutierrez y C. Kloos, «Aligning assessment with learning outcomes in outcome-based education,» *Education Engineering (EDUCON), 2010 IEEE,* pp. 1239-1246, 2010.

[31] S.-L. Wang, «Using Inductive Method to Teach Nuclear Energy,» *2010 1st International Nuclear\&Renewable Energy Conference (INREC),* pp. 1-2, 2010.

[32] V. Kroumov, K. Shibayama y A. Inoue, «Interactive Learning Tools For Enhancing The Education In Control Systems,» *IEEE Frontiers in Education Conference (FIE),* vol. 1, pp. 23-28, 2003.

[33] M. K. Bayrakceken y A. Arisoy, «An educational setup for nonlinear control systems: Enhancing the motivation and learning in a targeted curriculum by experimental practices [Focus on Education],» *Control Systems, IEEE,* vol. 33, nº 2, pp. 64-81, 2013.

[34] P. F. Edson y P. J. Valfredo, «Microprocessors From Theory to Practice a Didactical Experience,» *IEEE Frontiers in Education (FIE),* pp. S1D-4, 2004.

[35] E. P. Ferlin y M. J. Tozzi, «First Integrated Examination of the Computer Engineering Program,» *Frontiers in Education, 2002. FIE 2002. 32nd Annual,* vol. 1, pp. T2B--1, 2002.

[36] V. Subbian y C. Purdy, «Redesigning an advanced embedded systems course: A step towards interdisciplinary engineering education,» *Integrated STEM Education Conference (ISEC), 2013 IEEE,* pp. 1-4, 2013.

[37] B. S. Bloom y D. R. Krathwohl, Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain, Longmans, 1956.

[38] A. Kumar, S. Fernando y R. C. Panicker, «Project-based learning in embedded systems education using an fpga platform,» *Education, IEEE Transactions on,* vol. 56, nº 4, pp. 407-415, 2013.

[39] G. Beauchamp, M. Jimenez, R. Mulero y A. Ortiz, «Tradeoffs Implementing Digital Control Systems for Electrical Engineering Course Projects,» *IEEE Frontiers in Education Conference (FIE),* pp. 1-6, 2012.

[40] M. Jimenez, G. Beauchamp, R. Mulero y M. Gonzalez, «Integrating Control Concepts in an Embedded Systems Design Course,» *IEEE Frontiers in Education Conference (FIE),* pp. 1273-1278, 2013.

[41] G. Beauchamp Baez, «INEL 5508 Project Development Instructions,» *University of Puerto Rico at Mayaguez,* 2014.

[42] Texas Instruments Incorporated, TMS320F28069 Technical Reference Manual, 2014.

[43] Texas Instruments Incorporated, DRV8833 User's Manual, 2014.

[44] J. F. Patarroyo Montenegro, «Summer Work Internal Report: A Microcontroller Based System for Educational Purposes on the Digital Control Systems Course,» *Texas Instruments Assistanship,* 2014.

[45] R. Krauss y J. Croxell, «A Low-Cost Microcontroller-in-the-Loop Platform for Controls Education,» *American Control Conference (ACC),* pp. 4478-4483, 2012.

[46] C. H. Choi, «A Linear Control Systems Course with Emphasis on Embedded Control,» *American Society for Engineering Education (ASEE),* 2010.

[47] Y. Roshan y M. Moallem, «Developing a Course and Laboratory for Embedded Control of Mechatronic Systems,» *American Society for Engineering Education,* 2011.

[48] P. Martí, M. Velasco y J. Fuertes, «Design of an Embedded Control System Laboratory Experiment,» *Industrial Electronics, IEEE Transactions on,* vol. 57, nº 10, pp. 3297-3307, 2010.

[49] M. J. Bass y C. M. Christensen, «The future of the microprocessor business,» *IEEE Spectrum,* vol. 39, nº 4, pp. 34-39, 2002.

[50] J. F. Patarroyo-Montenegro, «A Microcontroller Based System for Improving the Learning Process of Undergraduate Students in the Area of Control Systems,» *SACNAS Annual Conference,* 2014.

[51] J. F. Patarroyo-Montenegro, G. Beauchamp-Baez y A. Santiago-Roman, «Design and Development of an Educational Microcontroller Based Control Station,» *IEEE Frontiers In Education (FIE),* 2015.

[52] J. F. Patarroyo-Montenegro, G. Beauchamp-Baez y A. Santiago-Roman, «A Methodology To Teach Students In The Digital Control Systems Course To Implement Digital Controllers Using Embedded Systems,» *ASEE annual meeting,* 2015.

[53] Texas Instruments Incorporated, «ControlSUITE™ Software Suite: Essential Software and Development Tools for C2000™ Microcontrollers,» 2015. [En línea]. Available: http://www.ti.com/tool/controlsuite.

[54] J. McTighe y R. S. Thomas, «Backward Design,» *Educational Leadership,* 2005.

[55] E. A. C. a. others, «Criteria for Accrediting Engineering Programs,» *ABET Report E1 11/19,* vol. 3, 2003.

[56] G. Beauchamp, «INEL 5508 Digital Control Systems Course Syllabus,» UPRM Electrical and Computer Engineering Department, Mayaguez, PR., 2007.

[57] W. M. Sullivan, Work and integrity: The crisis and promise of professionalism in America, HarperBusiness New York, 1995.

[58] J. Lee Rodgers y W. A. Nicewander, «Thirteen ways to look at the correlation coefficient,» *The American Statistician,* vol. 42, nº 1, pp. 59--66, 1988.

[59] A. Jonsson y G. Svingby, «The use of scoring rubrics: Reliability, validity and educational consequences,» *Educational research review,* vol. 2, nº 2, pp. 130--144, 2007.

[60] G. D. Israel, «Determining sample size,» University of Florida Cooperative Extension Service, Institute of Food and Agriculture Sciences, EDIS, 1992.

[61] Quanser Consulting Incorporated, «Ball & Beam Experiment Student's Handout,» 2014.

[62] A. E. Kelly, R. A. Lesh y J. Y. Baek, Handbook of design research methods in education: Innovations in science, technology, engineering, and mathematics learning and teaching, Routledge, 2008.

[63] 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, pp. 64--74, 1998.

[64] Engineering Accreditation Commission, «Criteria for Accrediting Engineering Programs,» *ABET Report E1 11/19,* vol. 3, 2003.

[65] A. a. E. M. Donner, «Sample size requirements for reliability studies,» *Statistics in medicine,* vol. 6, nº 4, pp. 441-448, 1987.

[66] G. Beauchamp Baez, «Digital Control Systems Course Curriculum,» *University of Puerto Rico at Mayaguez,* 2014.

[67] Q. Consulting, «Quanser Consulting Website,» 2015. [En línea]. Available: http://www.quanser.com/.

Appendix A: C Code Fragments

In this Appendix, the C code fragments used for programming and configuring the MCS are presented. First, the configuration libraries for the main peripherals of the TMS320F28069 are shown. Finally, we present a code fragment that performs a Lead-Lag controller for the Ball & Beam system.

1. ADC Unit

ADC interrupts are enabled and configured to work in continuous mode. Input channels are selected to be in ADCINA6 and ADCINA7.

```
        AdcRegs.ADCCTL2.bit.ADCNONOVERLAP    = 1;   // Enable non-overlap mode
        AdcRegs.ADCCTL1.bit.INTPULSEPOS      = 1;   // ADCINT2 trips after AdcResults latch
        AdcRegs.INTSEL1N2.bit.INT2E          = 1;   // Enabled ADCINT2
        AdcRegs.INTSEL1N2.bit.INT2CONT       = 1;   // ENABLE ADCINT2 Continuous mode
        AdcRegs.INTSEL1N2.bit.INT2SEL        = 1;   // setup EOC1 to trigger ADCINT2 to fire
        AdcRegs.ADCSOC0CTL.bit.CHSEL         = 6;   // set SOC0 channel select to ADCINA6
        AdcRegs.ADCSOC1CTL.bit.CHSEL         = 7;   // set SOC1 channel select to ADCINA7
        AdcRegs.ADCINTSOCSEL1.bit.SOC0       = 0x02;//Set SOC0 to trigger after ADCINT2
        AdcRegs.ADCINTSOCSEL1.bit.SOC1       = 0x02;//Set SOC1 to trigger after ADCINT2
        AdcRegs.ADCSOC0CTL.bit.ACQPS         = 6;   // set SOC0 S/H Window to 7 ADC Clock Cycles, (6 ACQPS plus 1)
        AdcRegs.ADCSOC1CTL.bit.ACQPS         = 6;   // set SOC1 S/H Window to 7 ADC Clock Cycles, (6 ACQPS plus 1)
        AdcRegs.ADCSOCFRC1.bit.SOC1          = 1;
```

Code Fragment 1. ADC Unit Configuration

2. Enhanced Quadrature Encoder Unit

This unit only requires to configure the max position count value for position reading (4096).

```
        EQep1Regs.QDECCTL.bit.QSRC=00;       // QEP quadrature count mode
        EQep1Regs.QEPCTL.bit.FREE_SOFT=2;
        EQep1Regs.QEPCTL.bit.PCRM=01;        // PCRM=01 mode - QPOSCNT reset on Max Position
        EQep1Regs.QEPCTL.bit.QCLM=0;         // Latch on unit time out
        EQep1Regs.QPOSMAX=0x00000FFF;        //4096 counts
        EQep1Regs.QEPCTL.bit.QPEN=1;         // QEP enable
        EQep1Regs.QCAPCTL.bit.UPPS=1;        // for unit position
```

Code Fragment 2. eQEP Configuration

3. Enhanced Pulse Width Modulation Unit

This function sets ePWM module to work in up-down count mode and sets PWM frequency. To change PWM duty cycle, CMPA and CMPB registers should be changed from 0 to 4096.

```
#define CPU_CLK   80e6                          //Clock Freq.
#define PWM_CLK   9765                          //Freq. can be changed here
#define SP        CPU_CLK/(2*PWM_CLK)
#define TBCTLVAL  0x200E                        // NO Shadow Register. up-down count, timebase=SYSCLKOUT

void initEpwm()
{
         EPwm1Regs.TBCTR=0;                     //restart counter
         EPwm1Regs.CMPCTL.all=0x50;             // immediate mode for CMPA and CMPB
         EPwm1Regs.AQCTLA.all=0x90;             // CTR=CMPA when inc->EPWM1A=1, when dec->EPWM1A=0
         EPwm1Regs.AQCTLB.all=0x900;            // CTR=PRD ->EPWM1B=1, CTR=0 ->EPWM1B=0
         EPwm1Regs.ETPS.all=1;
         EPwm1Regs.TBCTL.all=0x0010+TBCTLVAL; // UP-DOWN Count mode - Enable Timer
         EPwm1Regs.TBPRD=SP;                    // sets PWM frequency to 10KHz
}
```

Code Fragment 3. ePWM Configuration

To convert voltage levels to duty cycle values, timer compare registers must be configured according to the description in Section 4.1.4.3.

```
void volt2Pwm(float32 volt){
         if(enable==0)
                  volt=0;
         if(volt>5)        //Volts Saturation
                  volt=5;
         if(volt<-5)
                  volt=-5;
         if(volt>=0){
                  EPwm1Regs.CMPA.half.CMPA=volt*818; //  818=5volts/4090
                  EPwm1Regs.CMPB=0;
         }else{
                  EPwm1Regs.CMPA.half.CMPA=0;
                  EPwm1Regs.CMPB=volt*(-818);
         }
}
```

Code Fragment 4. Volts to PWM conversion routine.

4. Timer Interrupt Unit

Timer Interrupt Unit only needs to be configured in the prescaling factor and max count value.

```
void ConfigCpuTimer(struct CPUTIMER_VARS *Timer, float Freq, float Period)
{
         Uint32   temp;
         // Initialize timer period:
         Timer->CPUFreqInMHz = Freq;
         Timer->PeriodInUSec = Period;
         temp = (long) (Freq * Period);
         Timer->RegsAddr->PRD.all = temp;
         // Set pre-scale counter to divide by 1 (SYSCLKOUT):
         Timer->RegsAddr->TPR.all  = 0;
         Timer->RegsAddr->TPRH.all  = 0;
         // Initialize timer control register:
         Timer->RegsAddr->TCR.bit.TSS = 1;      // 1 = Stop timer, 0 = Start/Restart Timer
         Timer->RegsAddr->TCR.bit.TRB = 1;      // 1 = reload timer
         Timer->RegsAddr->TCR.bit.TIE = 1;      // 0 = Disable/ 1 = Enable Timer Interrupt
         Timer->InterruptCount = 0; // Reset interrupt counter:
}
```

Code Fragment 5. Timer Interrupt Unit Configuration

5. Ball & Beam Lead-Lag Controller Implementation

This subroutine implements digitally a continuous-time a Lead-Lag controller for the B&B system using difference equations. Sensor registers should be multiplied by a sensitivity value to convert their reading to international system units.

```
 volatile float64 motorVolts=0,errorTheta=0,thetaRef=0,thetaRef1=0,thetaRef2=0,Error=0,Error1=0,Error2=0;
 volatile float64 Z0=0.986,Z01=0.995,Zp=0.75,Zp1=0.999,K=4.7,Kc=55;
__interrupt void cpu_timer0_isr(void)
{
         CpuTimer0.InterruptCount++;
         GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;        // toggle GPIO34 which controls LD3 on most controlCARDs
         PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;                //interrupt Acknowledge
         theta=EQep1Regs.QPOSCNT*ENCODER_SENSITIVITY+PI;        //Conversion to Radians
         thetaGrados=theta*57.295779;                           //Conversion to Degrees (for visualization)
         adcMaster=AdcResult.ADCRESULT0;                        //ADC READ0
         adcSlave=AdcResult.ADCRESULT1;                         //ADC READ1
         posMaster=(adcMaster-2047)*BAR_SENSITIVITY;   //Conversion to meters and sign Inversion <-=pos  ->=neg
         posSlave=(adcSlave-2047)*BAR_SENSITIVITY;
         Error=posMaster-posSlave;                              //Position Error
         thetaRef=Kc*Error-Kc*(Z01+Z0)*Error1+Kc*Z01*Z0*Error2+(Zp1+Zp)*thetaRef1-Zp1*Zp*thetaRef2;//Ball Position
                                                                                                  //Controller
         thetaRef2=thetaRef1; //Register update
         thetaRef1=thetaRef;
         Error2=Error1;
         Error1=Error;
         if(thetaRef>1.5707)//Angle Saturation
            thetaRef=1.5707;
         if(thetaRef<-1.5707)
            thetaRef=-1.5707;
         errorTheta=thetaRef-theta; //Angle Error
         motorVolts=K*errorTheta; //Servo Angle Controller
         volt2Pwm(motorVolts);   //Output Function
}
```

Code Fragment 6. Ball & Beam Lead-Lag Controller Implementation

Appendix B: Workshops Tests

This section presents the diagnostic test that was administered before the workshops and the pre and post-tests that were administered on each workshop.

1. Diagnostic Test

1. Have you taken the *Microprocessors I* course?
   a. Yes
   b. No
   c. Taking
   - If your answer is yes, indicate the grade obtained in this course: __

2. Have you taken the *Microprocessors Interfacing* course?
   a. Yes
   b. No
   c. Taking
   - If your answer is yes, indicate the grade obtained in this course: __

3. Have you taken the *Programming Algorithms* course?
   a. Yes
   b. No
   c. Taking
   - If your answer is yes, indicate the grade obtained in this course: __

4. Being 0 the lowest level of understanding and 5 the highest, indicate your knowledge level on the following programming languages.
   a. Matlab
   b. C Code
   c. C++ Code
   d. Assembly
   e. Java

5. Being 0 the lowest level of understanding and 5 the highest, indicate your knowledge level on the following microcontroller modules.
   a. Analog to Digital Converter (ADC)
   b. Pulse Width Modulator (PWM)
   c. Quadrature Encoder Unit (eQEP)
   d. Timer Interrupts

2. Workshop 1 Pre and Post-Test

1) Indicate 3 differences between implementing a digital controller using a MCU and using Simulink (6pts).

2) Indicate which are the four main modules of the microcontroller used for the Ball & Beam control (4pts).

3) Besides the microcontroller and power supply, which components are needed to control the Ball & Beam? (2 pts).

4) Explain the task done by each of the following sections of a basic C code microcontroller program (8pts).
    - General Declarations
    - Initializations
    - Infinite Loop
    - Timer Interrupts

---

3. Workshop 2 Pre and Post-Test

1) Explain the working principle of the sensor that measures the ball position in the Ball & Beam System. Which microcontroller module should be used to read this signal?

2) Explain the working principle of the sensor that measures the angle of the motor in the Ball & Beam system. Which microcontroller module should be used to read this signal?

3) Explain the working principle of an H Bridge motor driver (DRV8833).

4) Explain how the microcontroller could be used to generate an analog output that moves the motor using an H bridge. Which microcontroller module could be used in this task?

5) Explain how to make an analog signal conditioning circuit for the sensor that measures the position of the ball in the Ball & Beam system.

---

4. Workshop 3 Pre and Post-Test

1) In a microcontroller based control program, which peripheral is used to define the sampling period?

2) In which part of the microcontroller program should we insert the digital controller statements to make it work at the defined sampling period?

3) Explain how to implement the following transfer function in the microcontroller using C Code language.

$$G(z) = \frac{Y(z)}{X(z)} = \frac{z}{z-1}$$

4) For the following system:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{z^2 + 3.2z + 7}{z^3 + 2.7z^2 + 1.2z + 1}$$

   a) Indicate how many samples of the input should be stored to compute an output signal.
   b) Indicate how many samples of the output are needed to compute an output signal.

Appendix C: Rubrics for Assessing the DCS Project Report

Group:_______ Name:____ Date:_________________

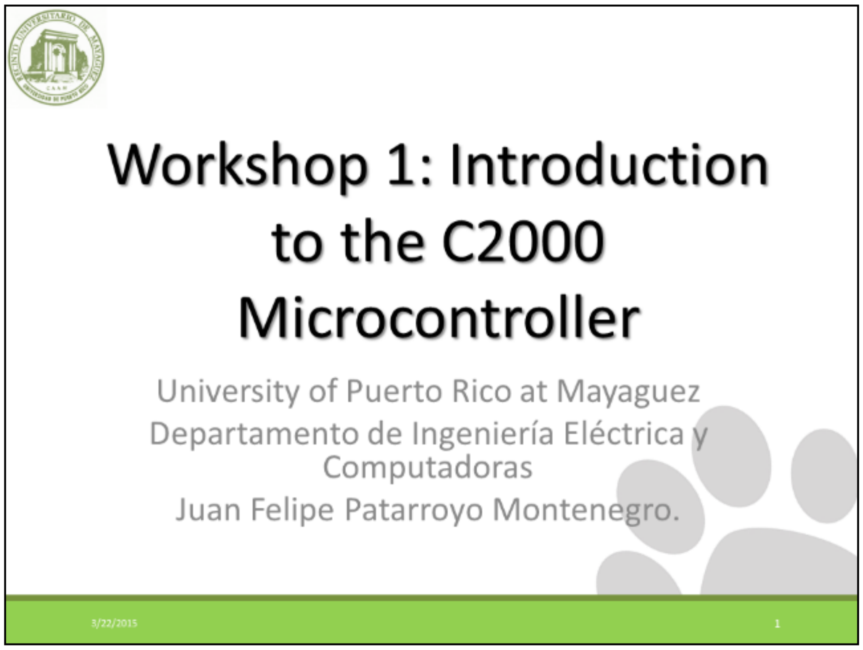
#	Item	Outcome	%	Good	Acceptable	Insufficient	Points	Total
1	Punctuality	7	20%	The student presents the report on time (80-100)	The student presents the report one day late. (70-79)	The student has two or more days without delivering his/her work. (0-69)		
2	Design of Controllers	3	10%	The student designed different types of controllers (P, PI, PID, Lead, Lag, Lead-Lag) (80-100).	The student designed a few types of controllers (70-79)	The student does not have a final controller design (0-69)		
			10%	Student evaluated the performance of the designed controllers (80-100)	Student gave a vague evaluation of the performance of the designed controllers (70-79)	Student is not capable of evaluating performance the designed controllers. (0-69)		
3	Simulations	8	10%	The student simulated the designed controllers in Simulink using saturation blocks. (80-100)	The student simulated some of the designed controllers in Simulink but did not use saturation blocks. (70-79)	The student does not simulate any of the proposed controllers (0-69)		
			10%	The student is capable of explaining whether the controller meets or not the design requirements (80-100)	The student barely explains whether the controller meets or not the design requirements (70-79)	The student cannot explain the design and simulation process.(0-69)		
4	Controller Implementation	4	7%	The student implemented the designed controller effectively on Simulink-Quarc (80-100)	The student implemented the designed controller on Simulink with some inaccuracies (70-79)	The student did not implement the designed controller on Simulink (0-69)		
			7%	The student used difference equations to develop a C Code program that performs the designed controller. (80-100)	The student developed a C Code program that performs the designed controller with errors. (70-79)	The student did not develop a C Code program that performs the designed controller. (0-69)		
			6%	The student explains how he/she used the block diagram of the system to develop an embedded digital controller. (80-100)	The student barely explains how to use the block diagram of the system to develop an embedded digital controller. (70-79)	The student is not capable of explaining the implementation process of an embedded controller. (0-69)		
5	Controller Performance	5	10%	The student implemented controller meets the design requirements using Simulink and the Microcontroller. (80-100)	The implemented controller works but does not meet the design requirements using Simulink and/or the Microcontroller. (70-79)	The implemented controller is unstable or it does not work. (0-69)		
			10%	The student can explain the discrepancies between the simulated and the real controller performance. (80-100)	The student barely explains the performance of the implemented controller. (70-79)	The student is not capable of justifying this behavior.(0-69)		
						Final Grade		

#	Course Outcomes	ABET Outcome
3	Design a single-input single-output feedback controller capable of achieving the design criteria for the system.	c
4	Implement a digital controller using a digital computer and software	e
5	Validate the performance of the closed-loop system	b
7	Preparation of a written report about the final project	g
8	Use modern engineering tools (MATLAB, labVIEW, PSPICE…) for the design and implementation of a process control and instrumentation system	k

Appendix D: Workshops Presentations

In this section we show the Power Point Presentations that were used to dictate the workshops.

1. Workshop 1 Presentation

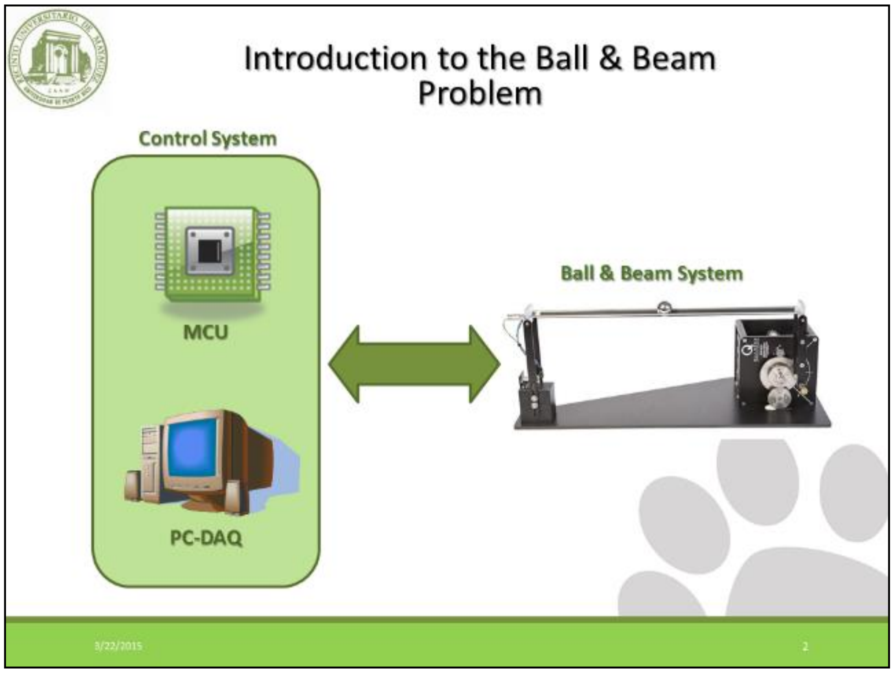


Workshop 1: Introduction to the C2000 Microcontroller

University of Puerto Rico at Mayaguez
Departamento de Ingeniería Eléctrica y Computadoras
Juan Felipe Patarroyo Montenegro.

3/22/2015 1

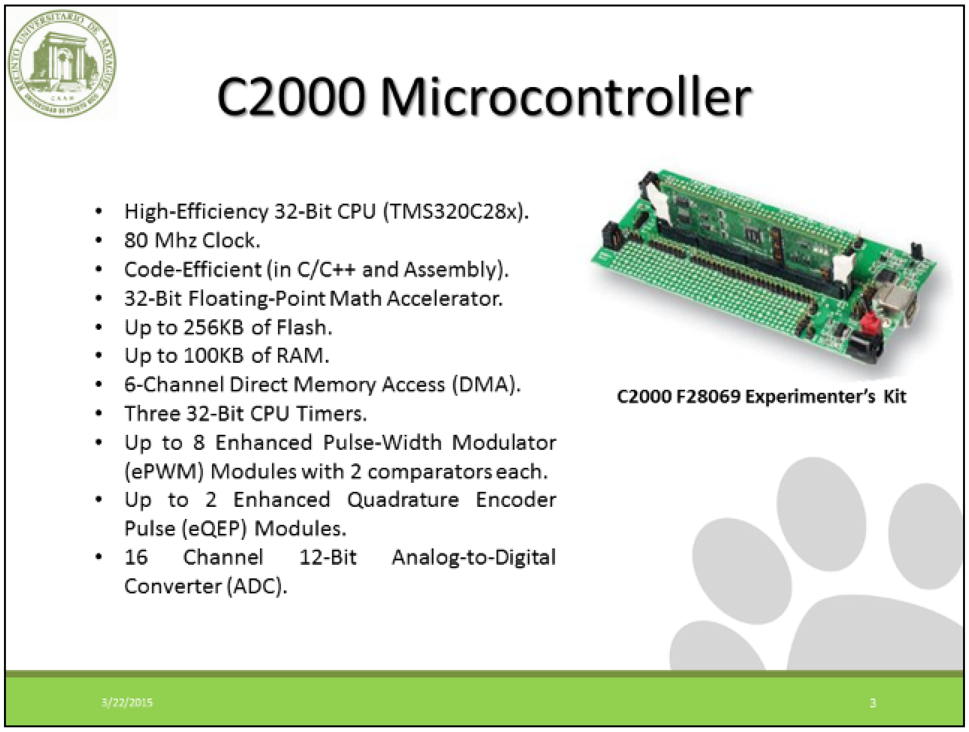
Workshop 1 Slide 1



Introduction to the Ball & Beam Problem

3/22/2015 2

Workshop 1 Slide 2



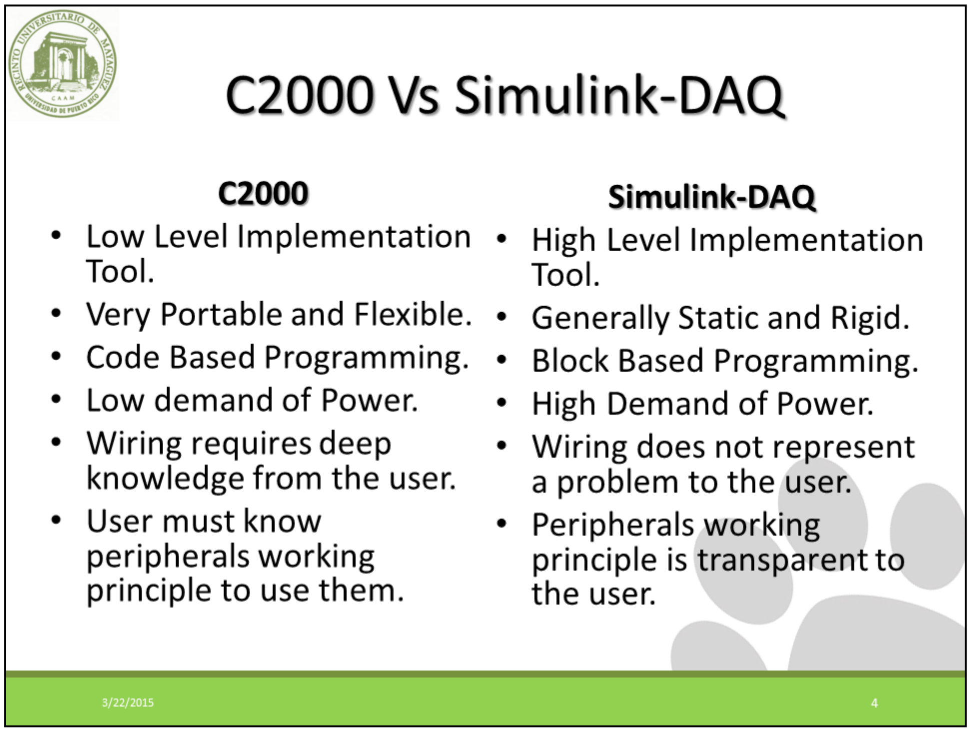
C2000 Microcontroller

- High-Efficiency 32-Bit CPU (TMS320C28x).
- 80 Mhz Clock.
- Code-Efficient (in C/C++ and Assembly).
- 32-Bit Floating-Point Math Accelerator.
- Up to 256KB of Flash.
- Up to 100KB of RAM.
- 6-Channel Direct Memory Access (DMA).
- Three 32-Bit CPU Timers.
- Up to 8 Enhanced Pulse-Width Modulator (ePWM) Modules with 2 comparators each.
- Up to 2 Enhanced Quadrature Encoder Pulse (eQEP) Modules.
- 16 Channel 12-Bit Analog-to-Digital Converter (ADC).

C2000 F28069 Experimenter's Kit

3/22/2015 3

Workshop 1 Slide 3



C2000 Vs Simulink-DAQ

C2000

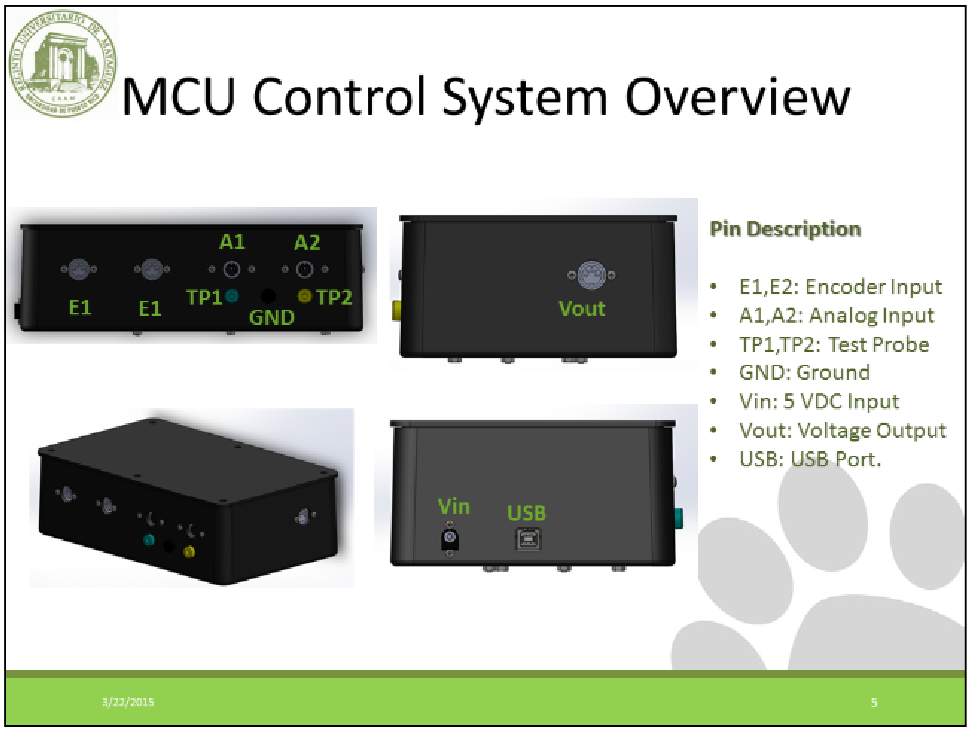
- Low Level Implementation Tool.
- Very Portable and Flexible.
- Code Based Programming.
- Low demand of Power.
- Wiring requires deep knowledge from the user.
- User must know peripherals working principle to use them.

Simulink-DAQ

- High Level Implementation Tool.
- Generally Static and Rigid.
- Block Based Programming.
- High Demand of Power.
- Wiring does not represent a problem to the user.
- Peripherals working principle is transparent to the user.

3/22/2015 4

Workshop 1 Slide 4



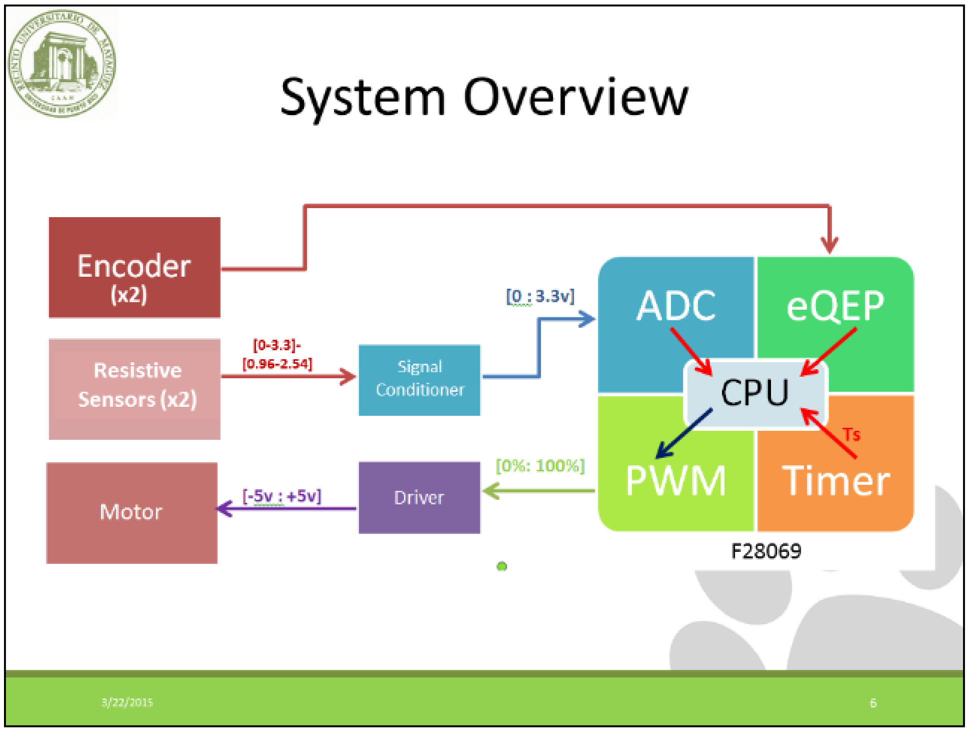
MCU Control System Overview

Pin Description

- E1,E2: Encoder Input
- A1,A2: Analog Input
- TP1,TP2: Test Probe
- GND: Ground
- Vin: 5 VDC Input
- Vout: Voltage Output
- USB: USB Port.

3/22/2015 5

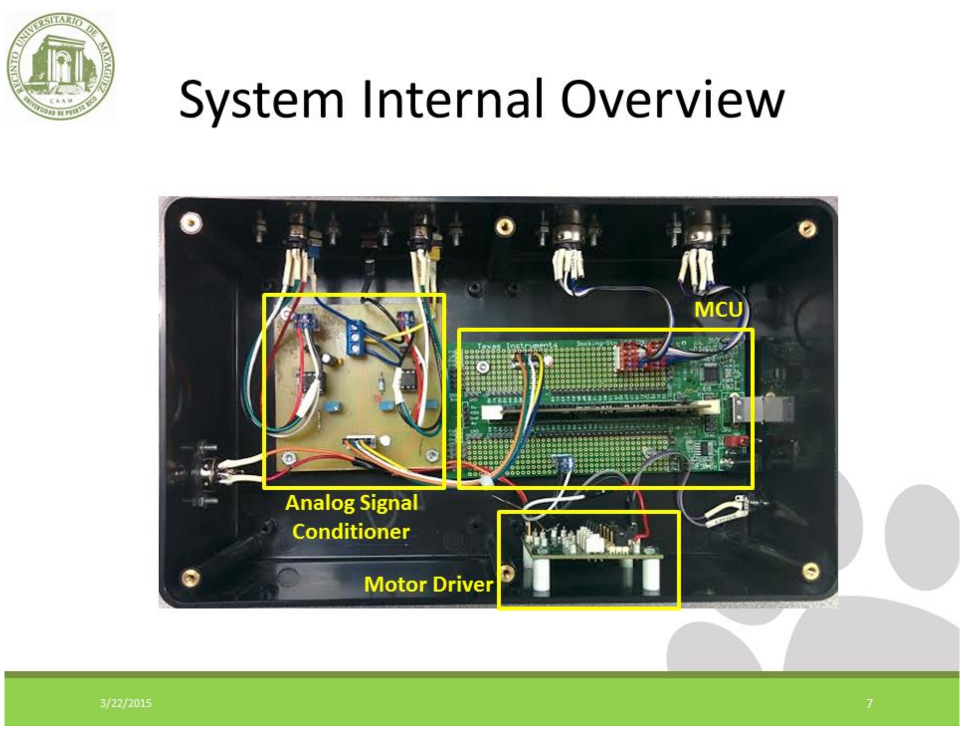
Workshop 1 Slide 5



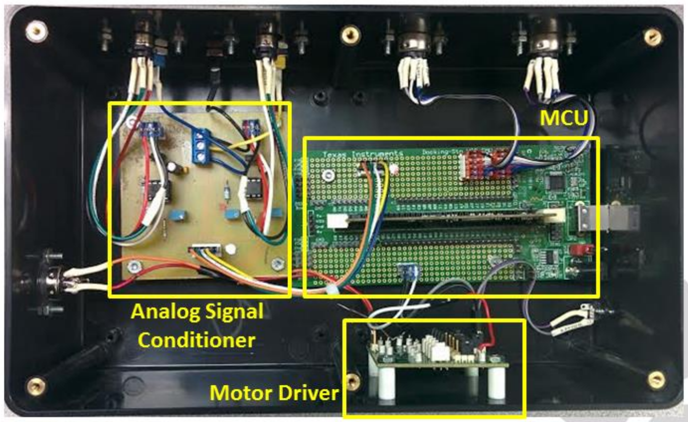
System Overview

3/22/2015 6

Workshop 1 Slide 6

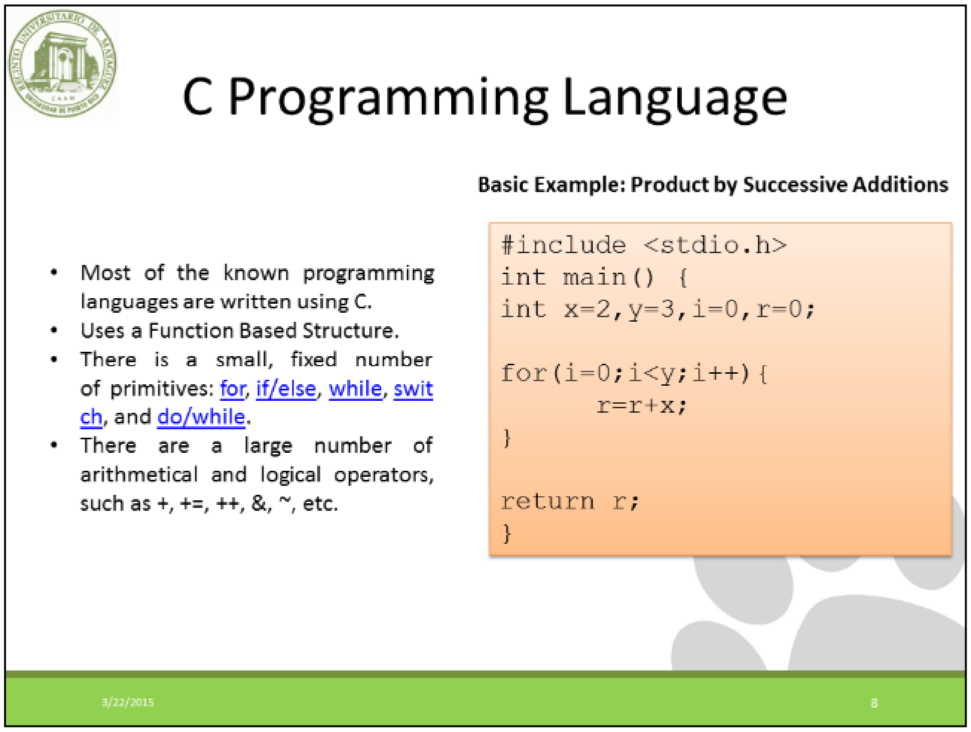


System Internal Overview



3/22/2015 7

Workshop 1 Slide 7



C Programming Language

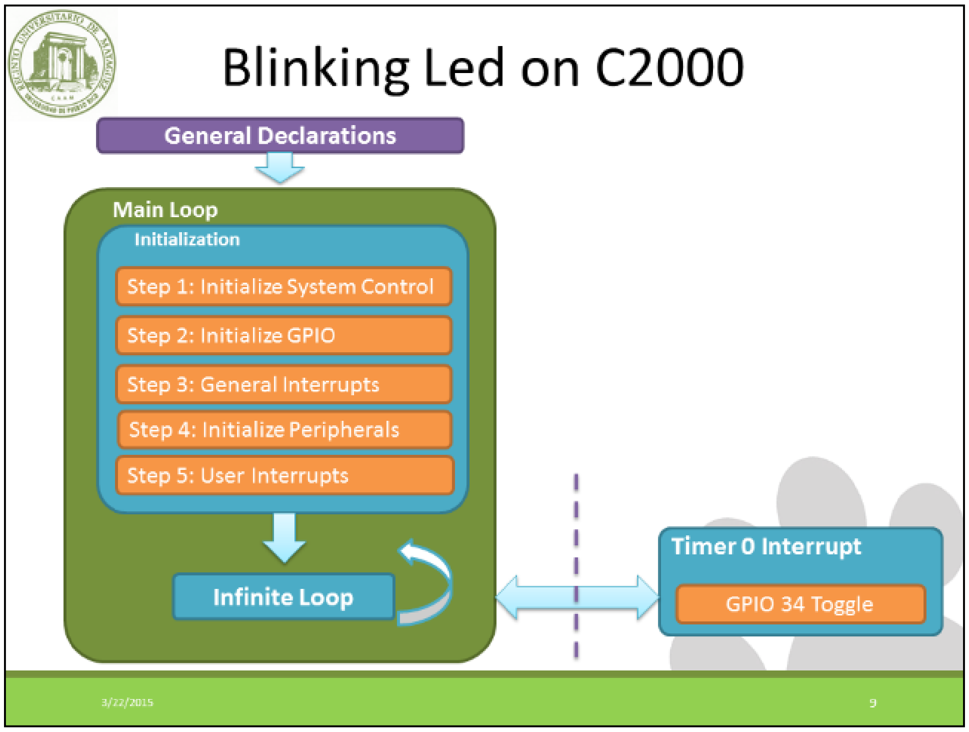
- Most of the known programming languages are written using C.
- Uses a Function Based Structure.
- There is a small, fixed number of primitives: for, if/else, while, switch, and do/while.
- There are a large number of arithmetical and logical operators, such as +, +=, ++, &, ~, etc.

Basic Example: Product by Successive Additions

```
#include <stdio.h>
int main() {
int x=2,y=3,i=0,r=0;

for(i=0;i<y;i++){
        r=r+x;
}

return r;
}
```

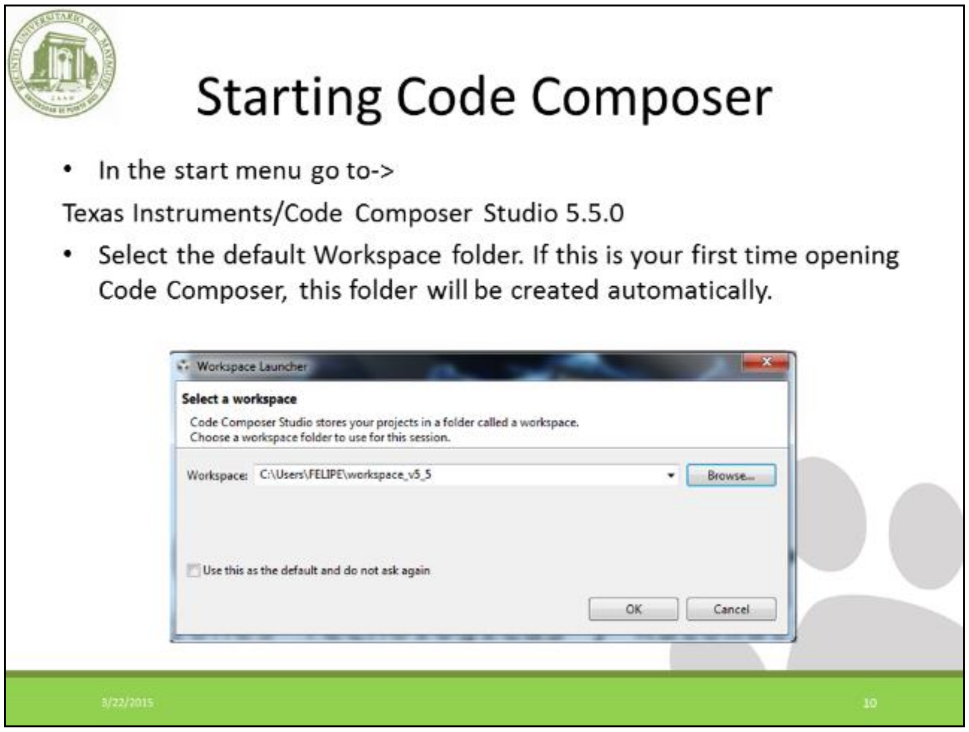
3/22/2015 8

Workshop 1 Slide 8



Blinking Led on C2000

Workshop 1 Slide 9



Starting Code Composer

- In the start menu go to->

Texas Instruments/Code Composer Studio 5.5.0

- Select the default Workspace folder. If this is your first time opening Code Composer, this folder will be created automatically.

Workspace Launcher

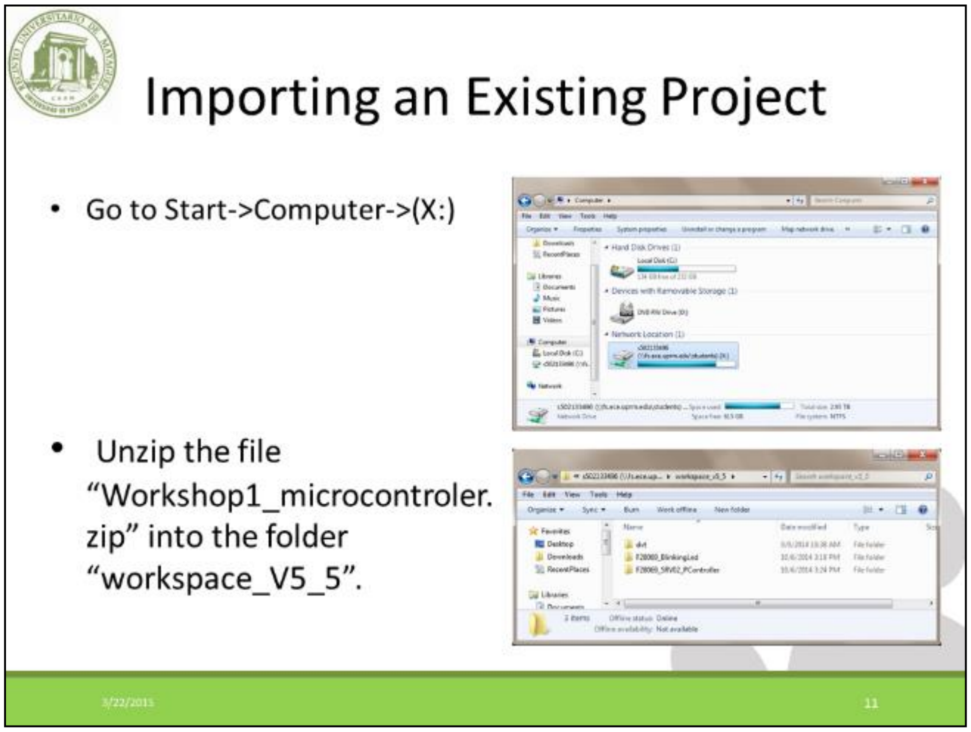
Select a workspace

Code Composer Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\FELIPE\workspace_v5_5

☐ Use this as the default and do not ask again

Workshop 1 Slide 10

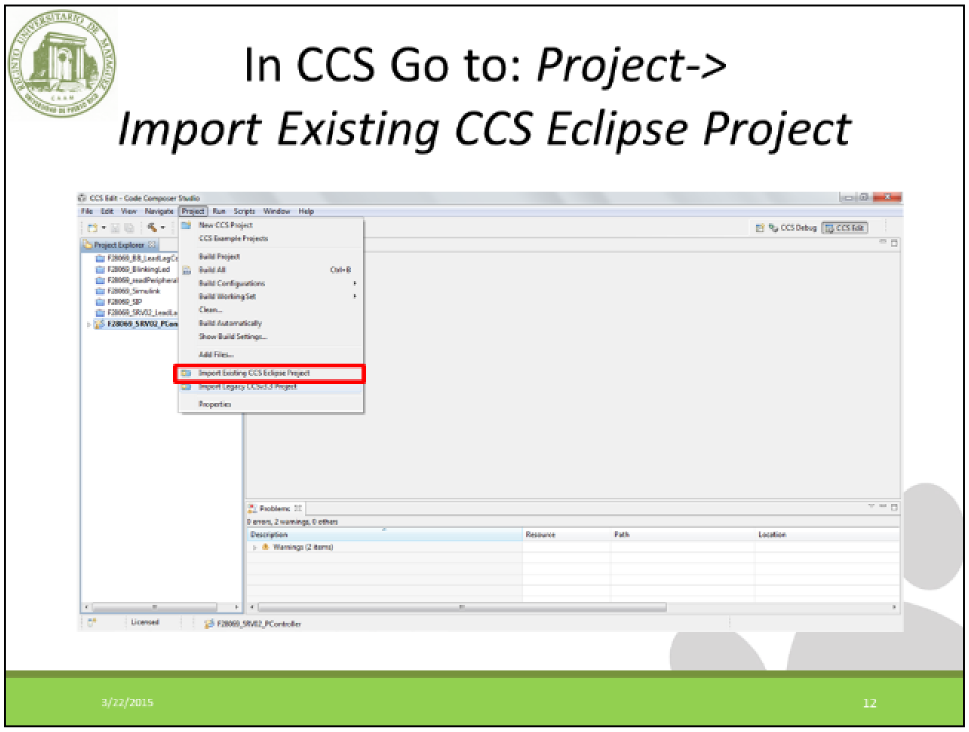


Importing an Existing Project

- Go to Start->Computer->(X:)
- Unzip the file "Workshop1_microcontroler.zip" into the folder "workspace_V5_5".

3/22/2015 11

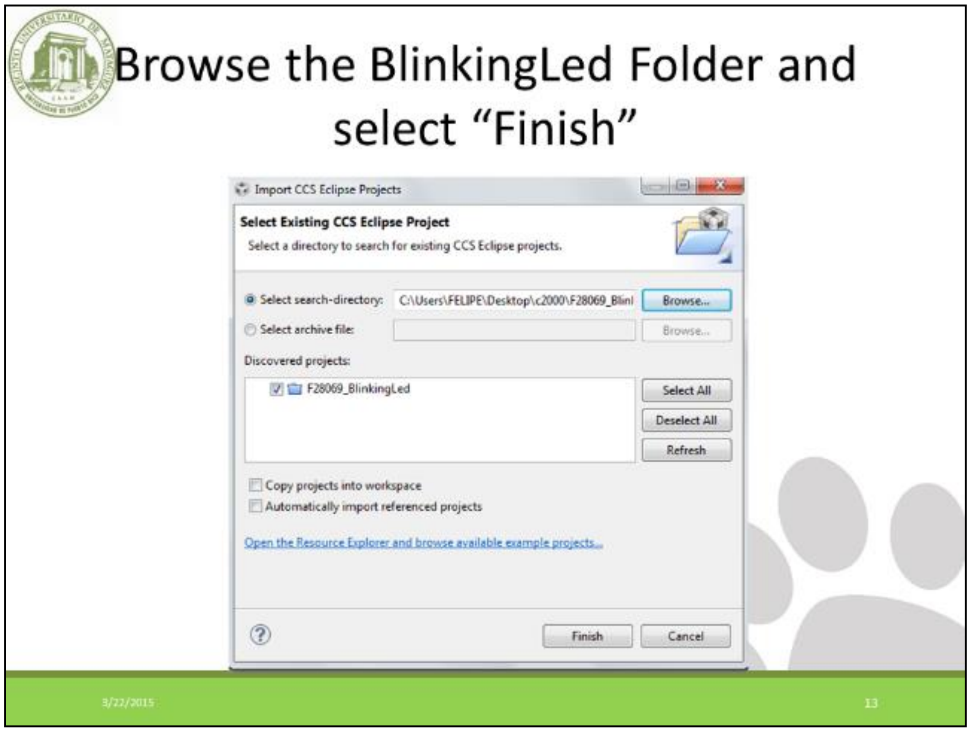
Workshop 1 Slide 11



In CCS Go to: *Project-> Import Existing CCS Eclipse Project*

3/22/2015 12

Workshop 1 Slide 12



Browse the BlinkingLed Folder and select "Finish"

Import CCS Eclipse Projects

Select Existing CCS Eclipse Project

Select a directory to search for existing CCS Eclipse projects.

Select search-directory: C:\Users\FELIPE\Desktop\c2000\F28069_Blinl

Select archive file:

Discovered projects:

[✓] F28069_BlinkingLed

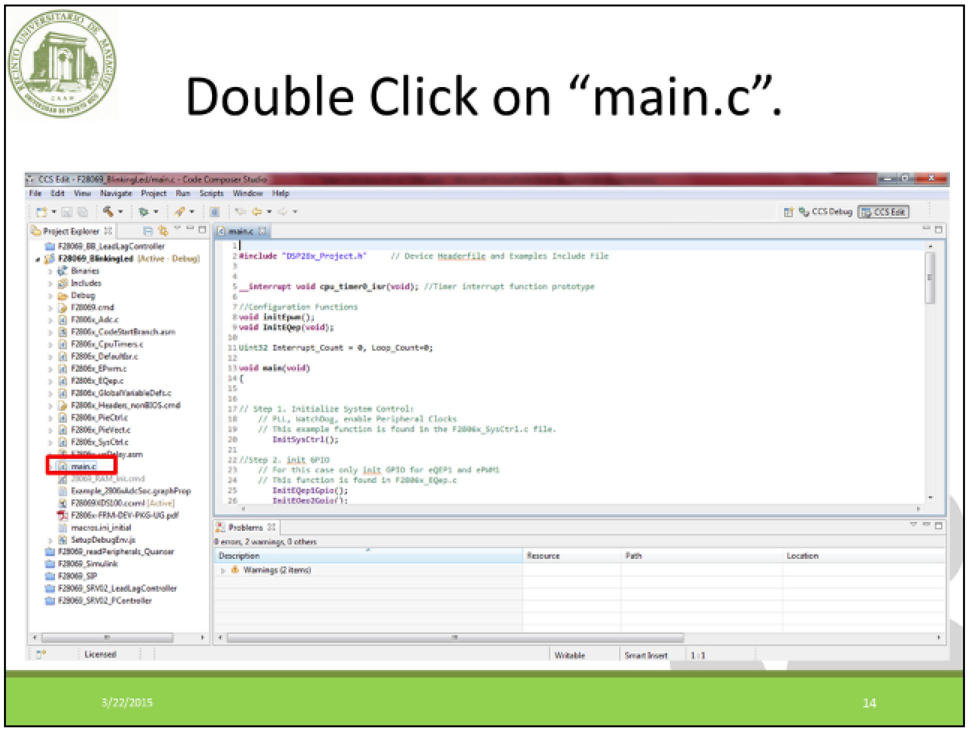
[ ] Copy projects into workspace

[ ] Automatically import referenced projects

Open the Resource Explorer and browse available example projects...

3/22/2015 13

Workshop 1 Slide 13



Double Click on "main.c".

CCS Edit - F28069_BlinkingLed/main.c - Code Composer Studio

```
F28069_B8_LeadLagController
F28069_BlinkingLed [Active - Debug]
    Binaries
    Includes
    Debug
    F28069.cmd
    F2806x_Adc.c
    F2806x_CodeStartBranch.asm
    F2806x_CpuTimers.c
    F2806x_DefaultIsr.c
    F2806x_EPwm.c
    F2806x_EQep.c
    F2806x_GlobalVariableDefs.c
    F2806x_Headers_nonBIOS.cmd
    F2806x_PieCtrl.c
    F2806x_PieVect.c
    F2806x_SysCtrl.c
    F2806x_usDelay.asm
    main.c
    28069_RAM_lnk.cmd
    Example_2806xAdcSoc.graphProp
    F28069XDS100.ccxml [Active]
    F2806x-FRM-DEV-PKG-UG.pdf
    macros.ini_initial
    SetupDebugEnv.js
F28069_readPeripherals_Quanser
F28069_Simulink
F28069_SIP
F28069_SRV02_LeadLagController
F28069_SRV02_PController
```

```c

#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File


__interrupt void cpu_timer0_isr(void); //Timer interrupt function prototype

//Configuration Functions
void initEpwm();
void InitEQep(void);

Uint32 Interrupt_Count = 0, Loop_Count=0;

void main(void)
{


// Step 1. Initialize System Control:
    // PLL, WatchDog, enable Peripheral Clocks
    // This example function is found in the F2806x_SysCtrl.c file.
       InitSysCtrl();

//Step 2. init GPIO
    // For this case only init GPIO for eQEP1 and ePWM1
    // This function is found in F2806x_EQep.c
       InitEQep1Gpio();
       InitEQep2Gpio();
```

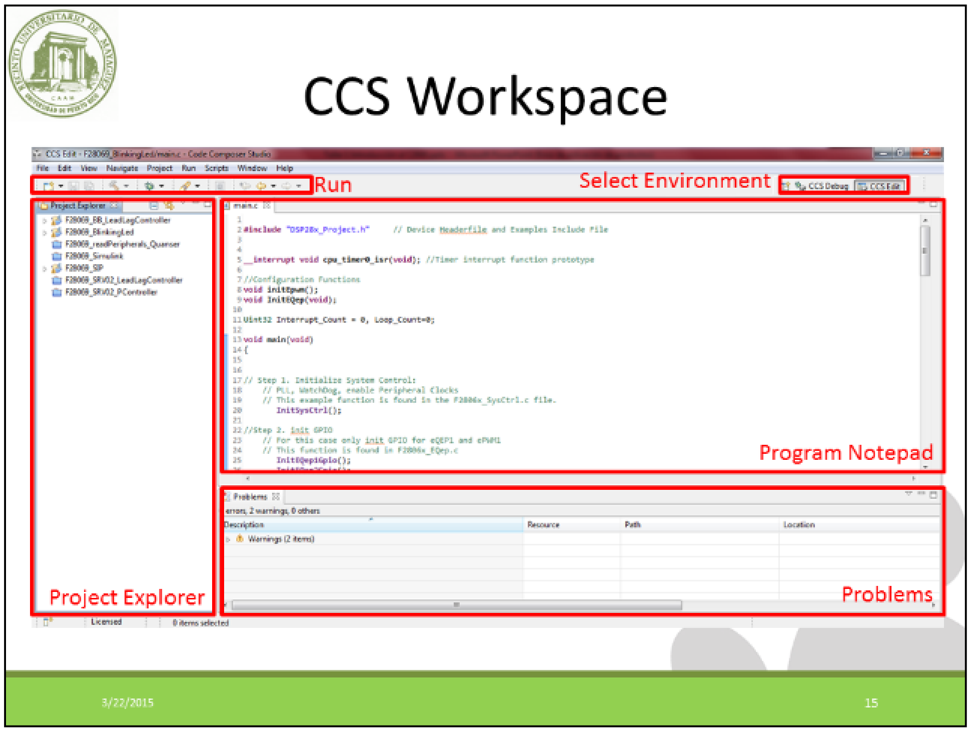
Problems

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location
Warnings (2 items)			

3/22/2015 14

Workshop 1 Slide 14



CCS Workspace

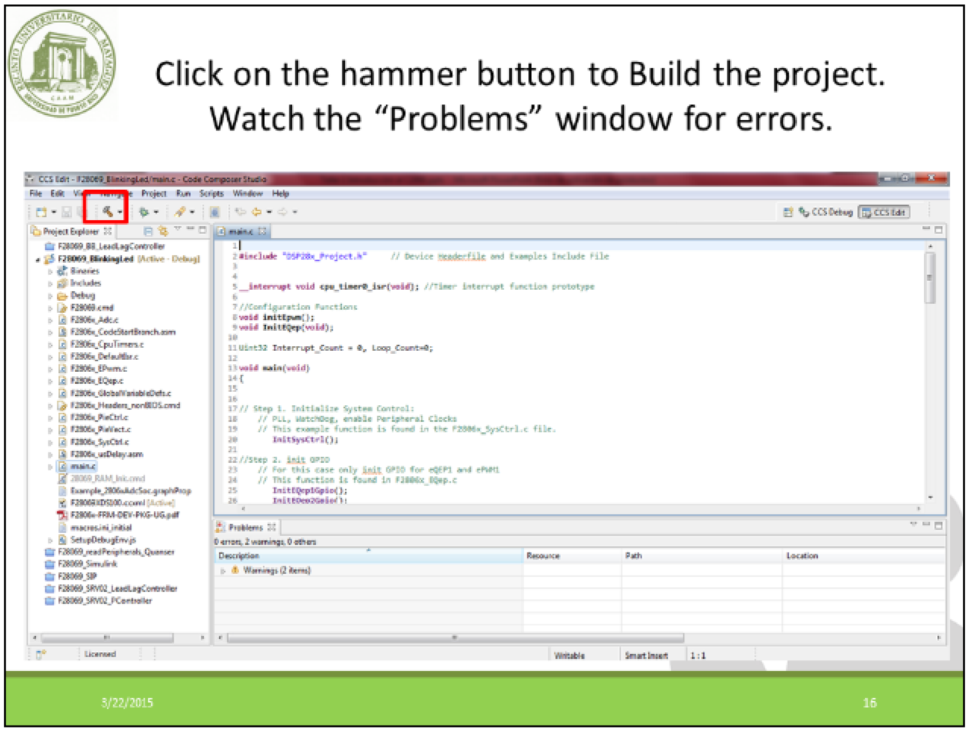
```
#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File


__interrupt void cpu_timer0_isr(void); //Timer interrupt function prototype

//Configuration Functions
void initEpwm();
void InitEQep(void);

Uint32 Interrupt_Count = 0, Loop_Count=0;

void main(void)
{


// Step 1. Initialize System Control:
    // PLL, WatchDog, enable Peripheral Clocks
    // This example function is found in the F2806x_SysCtrl.c file.
        InitSysCtrl();

//Step 2. init GPIO
    // For this case only init GPIO for eQEP1 and ePWM1
    // This function is found in F2806x_EQep.c
        InitEQep1Gpio();
```

errors, 2 warnings, 0 others

Description	Resource	Path	Location
Warnings (2 items)			

3/22/2015 15

Workshop 1 Slide 15



Click on the hammer button to Build the project. Watch the "Problems" window for errors.

CCS Edit - F28069_BlinkingLed/main.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
CCS Debug
CCS Edit
Project Explorer
F28069_BB_LeadLagController
F28069_BlinkingLed [Active - Debug]
Binaries
Includes
Debug
F28069.cmd
F2806x_Adc.c
F2806x_CodeStartBranch.asm
F2806x_CpuTimers.c
F2806x_DefaultIsr.c
F2806x_EPwm.c
F2806x_EQep.c
F2806x_GlobalVariableDefs.c
F2806x_Headers_nonBIOS.cmd
F2806x_PieCtrl.c
F2806x_PieVect.c
F2806x_SysCtrl.c
F2806x_usDelay.asm
main.c
28069_RAM_lnk.cmd
Example_2806xAdcSoc.graphProp
F28069XDS100.ccxml [Active]
F2806x-FRM-DEV-PKG-UG.pdf
macros.ini_initial
SetupDebugEnv.js
F28069_readPeripherals_Quanser
F28069_Simulink
F28069_SIP
F28069_SRV02_LeadLagController
F28069_SRV02_PController
main.c
Problems
Licensed
Writable
Smart Insert
1 : 1

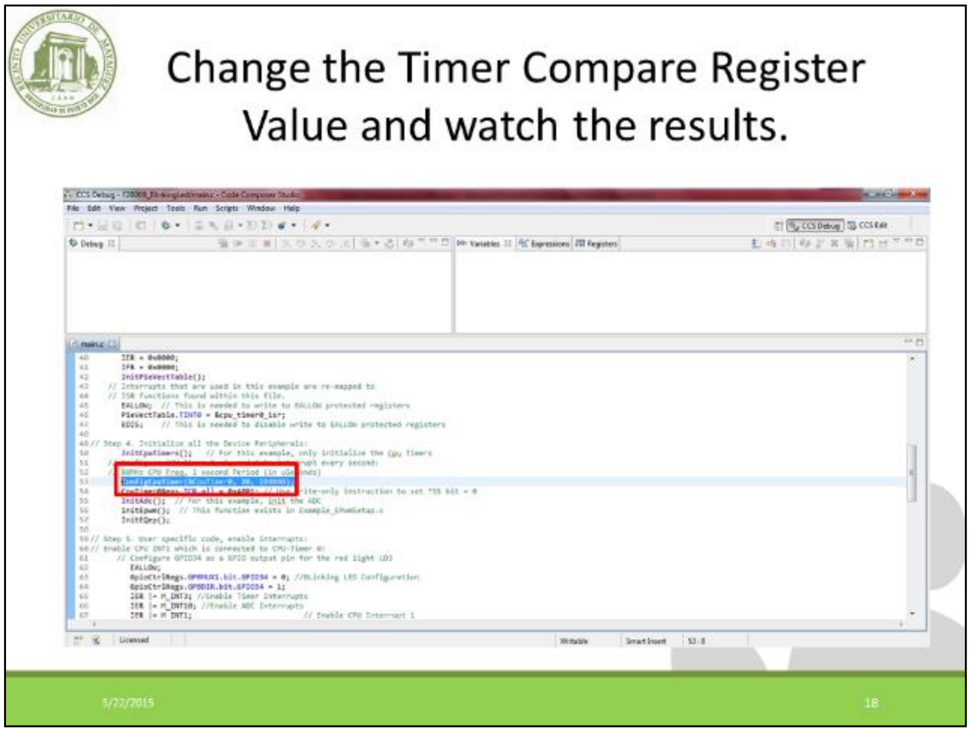
```
#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File


__interrupt void cpu_timer0_isr(void); //Timer interrupt function prototype

//Configuration Functions
void initEpwm();
void InitEQep(void);

Uint32 Interrupt_Count = 0, Loop_Count=0;

void main(void)
{


// Step 1. Initialize System Control:
    // PLL, WatchDog, enable Peripheral Clocks
    // This example function is found in the F2806x_SysCtrl.c file.
        InitSysCtrl();

//Step 2. init GPIO
    // For this case only init GPIO for eQEP1 and ePWM1
    // This function is found in F2806x_EQep.c
        InitEQep1Gpio();
        InitEQep2Gpio();
```

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location
Warnings (2 items)			

3/22/2015 16

Workshop 1 Slide 16



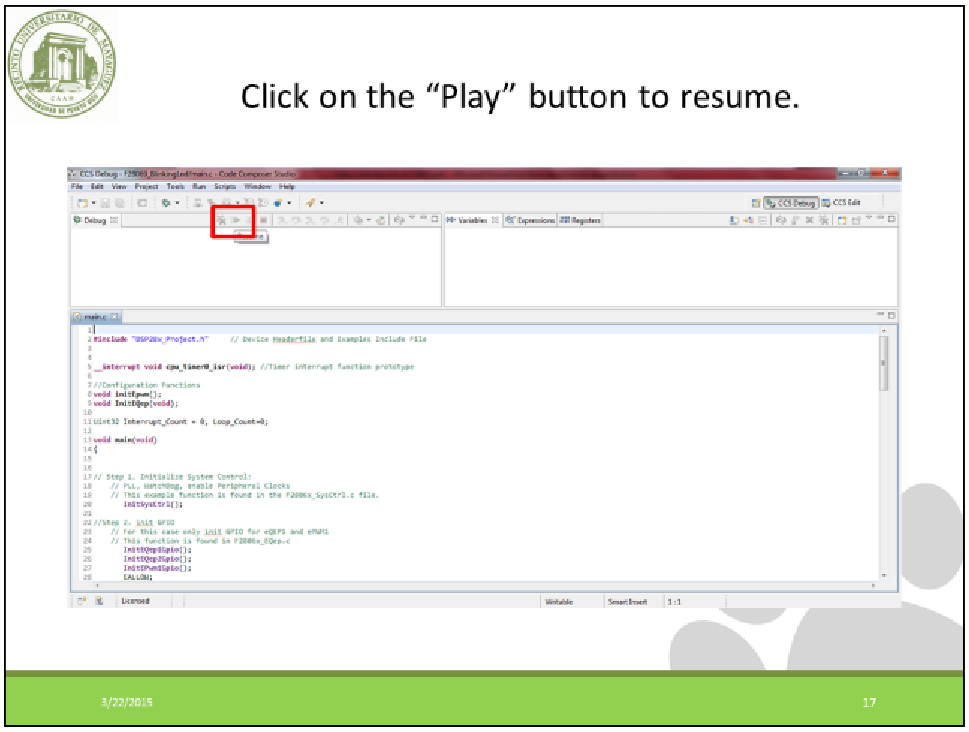
Change the Timer Compare Register Value and watch the results.

CCS Debug - F28069_BlinkingLed/main.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
CCS Edit
Debug
Variables
Expressions
Registers
main.c
Licensed
Writable
Smart Insert
53 : 8

```
        IER = 0x0000;
        IFR = 0x0000;
        InitPieVectTable();
    // Interrupts that are used in this example are re-mapped to
    // ISR functions found within this file.
        EALLOW;  // This is needed to write to EALLOW protected registers
        PieVectTable.TINT0 = &cpu_timer0_isr;
        EDIS;    // This is needed to disable write to EALLOW protected registers

// Step 4. Initialize all the Device Peripherals:
        InitCpuTimers();   // For this example, only initialize the Cpu Timers
    //                               rupt every second:
    // 80MHz CPU Freq, 1 second Period (in uSeconds)
        ConfigCpuTimer(&CpuTimer0, 80, 100000);
        CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
        InitAdc();  // For this example, init the ADC
        InitEpwm();  // This function exists in Example_EPwmSetup.c
        InitEQep();

// Step 5. User specific code, enable Interrupts:
// Enable CPU INT1 which is connected to CPU-Timer 0:
      // Configure GPIO34 as a GPIO output pin for the red light LED
          EALLOW;
          GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0; //Blinking LED Configuration
          GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;
          IER |= M_INT3; //Enable Timer Interrupts
          IER |= M_INT10; //Enable ADC Interrupts
          IER |= M_INT1;                          // Enable CPU Interrupt 1
```

3/22/2015 18

Workshop 1 Slide 17



Click on the "Play" button to resume.

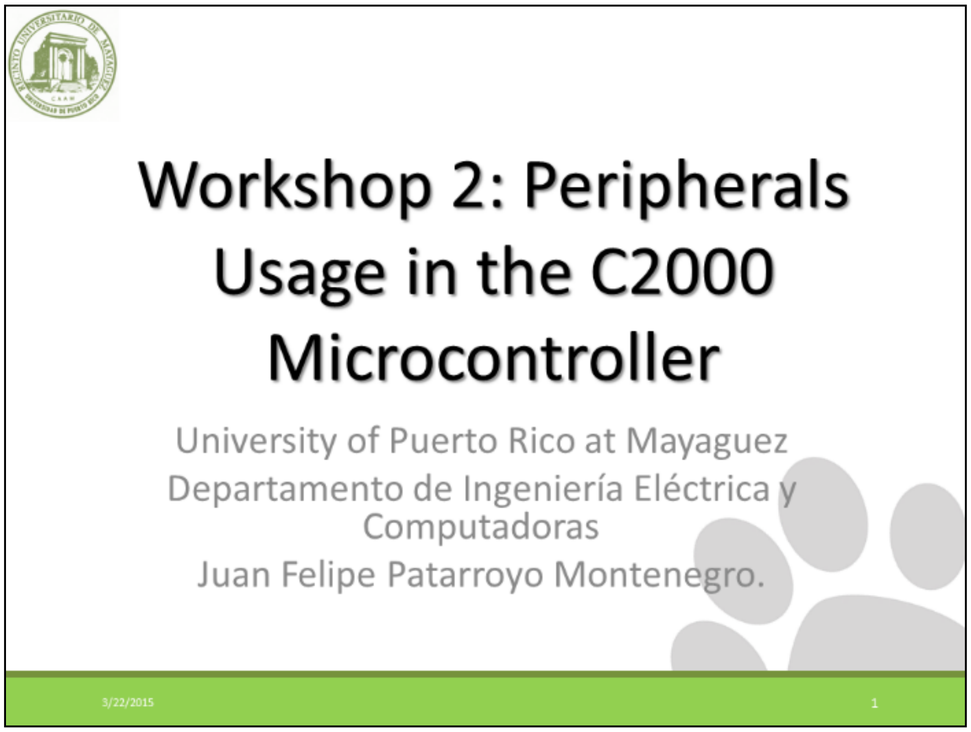
CCS Debug - F28069_BlinkingLed/main.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
CCS Edit
Debug
Variables
Expressions
Registers
main.c
Licensed
Writable
Smart Insert
1 : 1

```

#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File


__interrupt void cpu_timer0_isr(void); //Timer interrupt function prototype

//Configuration Functions
void initEpwm();
void InitEQep(void);

Uint32 Interrupt_Count = 0, Loop_Count=0;

void main(void)
{


// Step 1. Initialize System Control:
    // PLL, WatchDog, enable Peripheral Clocks
    // This example function is found in the F2806x_SysCtrl.c file.
        InitSysCtrl();

//Step 2. Init GPIO
    // For this case only init GPIO for eQEP1 and ePWM1
    // This function is found in F2806x_EQep.c
        InitEQep1Gpio();
        InitEQep2Gpio();
        InitEPwm1Gpio();
        EALLOW;
```

3/22/2015 17

Workshop 1 Slide 18

2. Workshop 2 Presentation

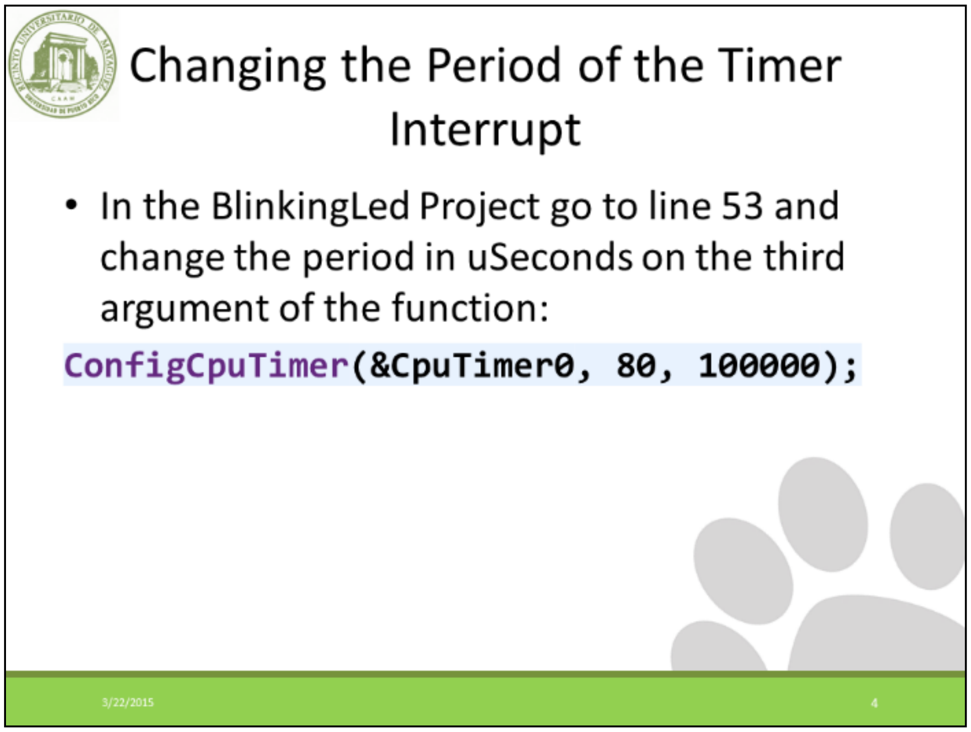


Workshop 2: Peripherals Usage in the C2000 Microcontroller

University of Puerto Rico at Mayaguez
Departamento de Ingeniería Eléctrica y Computadoras
Juan Felipe Patarroyo Montenegro.

3/22/2015 1

Workshop 2 Slide 1



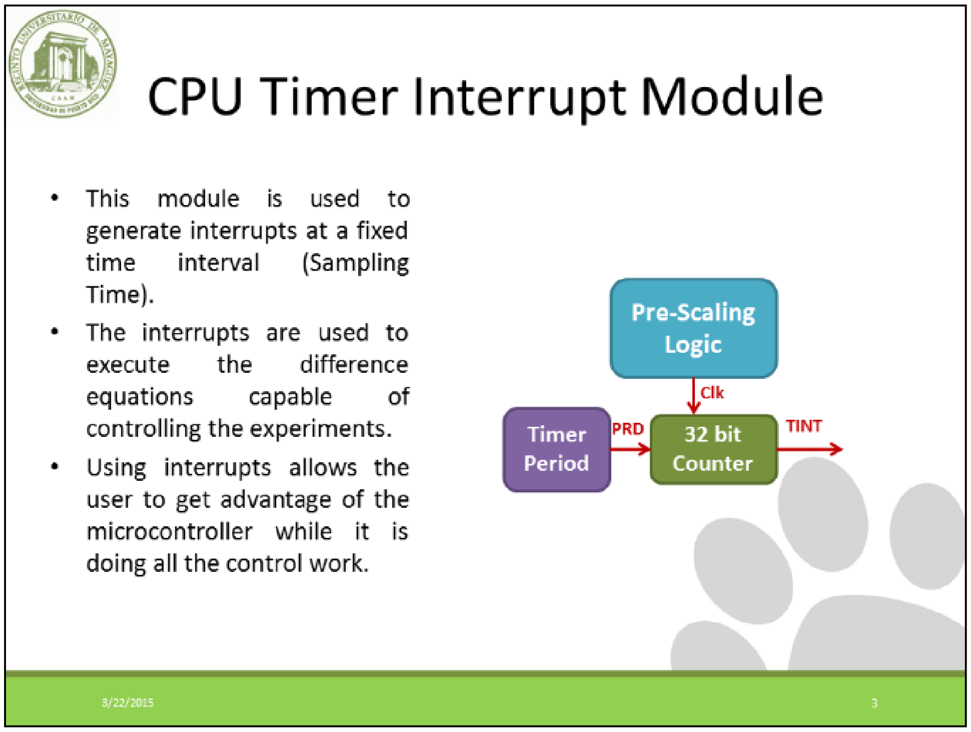
Changing the Period of the Timer Interrupt

- In the BlinkingLed Project go to line 53 and change the period in uSeconds on the third argument of the function:

```
ConfigCpuTimer(&CpuTimer0, 80, 100000);
```

3/22/2015 4

Workshop 2 Slide 2

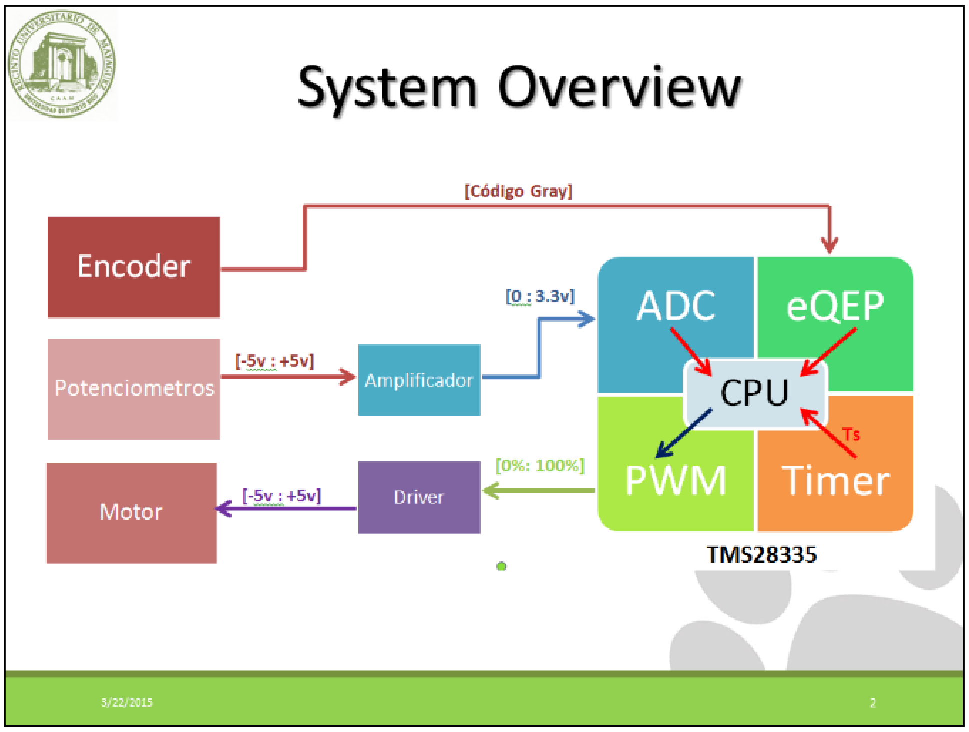


CPU Timer Interrupt Module

- This module is used to generate interrupts at a fixed time interval (Sampling Time).
- The interrupts are used to execute the difference equations capable of controlling the experiments.
- Using interrupts allows the user to get advantage of the microcontroller while it is doing all the control work.

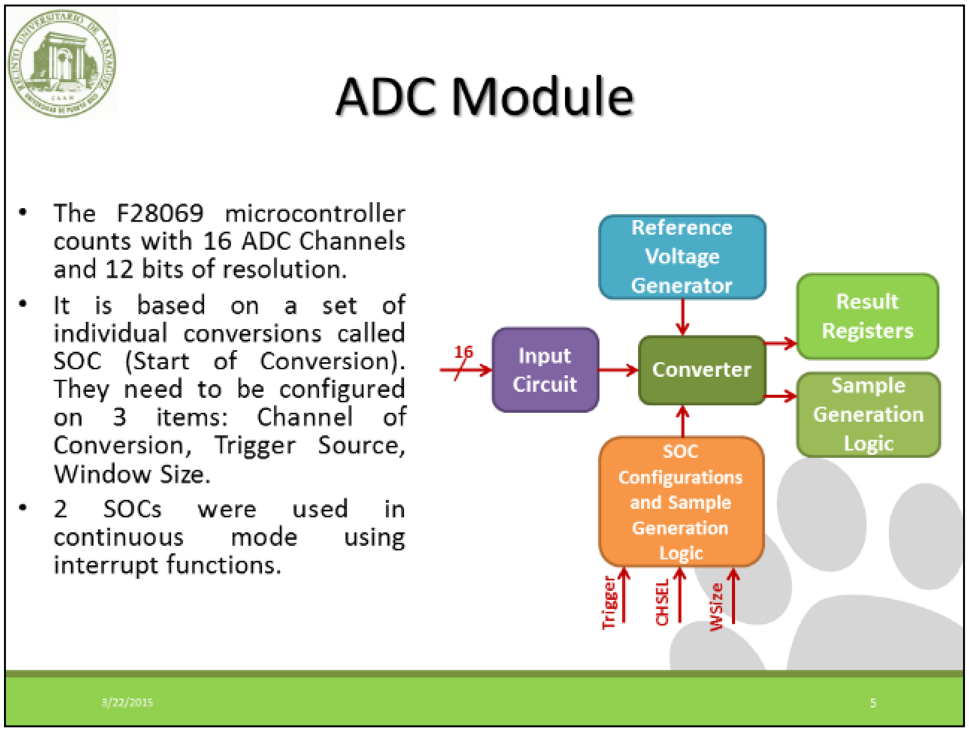
Pre-Scaling Logic
Clk
Timer Period
PRD
32 bit Counter
TINT
3/22/2015
3

Workshop 2 Slide 3



System Overview

Workshop 2 Slide 4



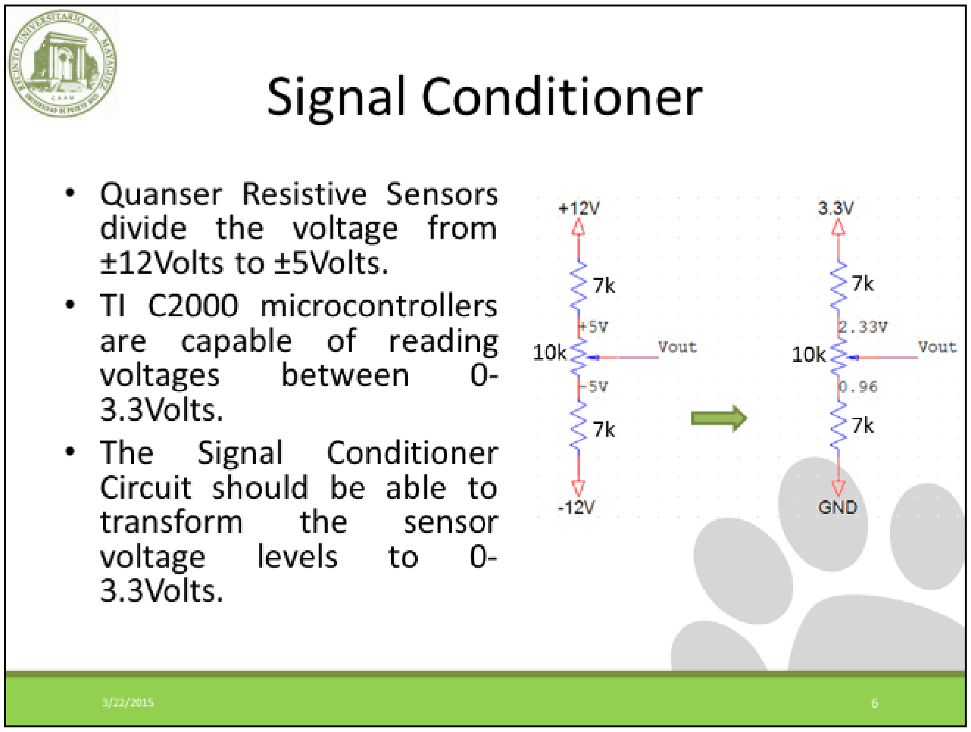
ADC Module

- The F28069 microcontroller counts with 16 ADC Channels and 12 bits of resolution.
- It is based on a set of individual conversions called SOC (Start of Conversion). They need to be configured on 3 items: Channel of Conversion, Trigger Source, Window Size.
- 2 SOCs were used in continuous mode using interrupt functions.

Reference Voltage Generator
Result Registers
16
Input Circuit
Converter
Sample Generation Logic
SOC Configurations and Sample Generation Logic
Trigger
CHSEL
WSize

3/22/2015 5

Workshop 2 Slide 5



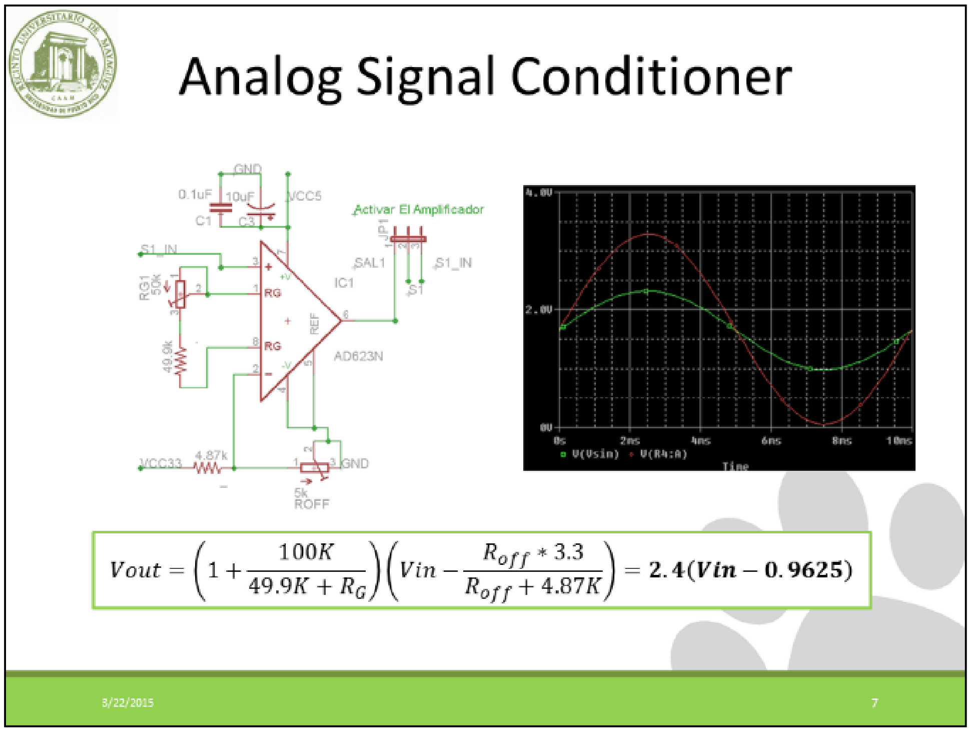
Signal Conditioner

- Quanser Resistive Sensors divide the voltage from ±12Volts to ±5Volts.
- TI C2000 microcontrollers are capable of reading voltages between 0-3.3Volts.
- The Signal Conditioner Circuit should be able to transform the sensor voltage levels to 0-3.3Volts.

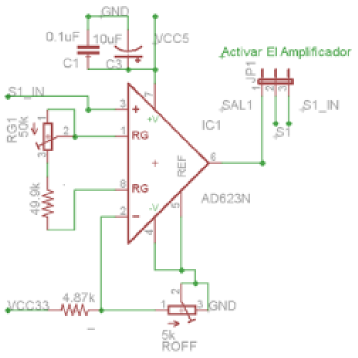
+12V
7k
+5V
10k
Vout
-5V
7k
-12V
3.3V
7k
2.33V
10k
Vout
0.96
7k
GND

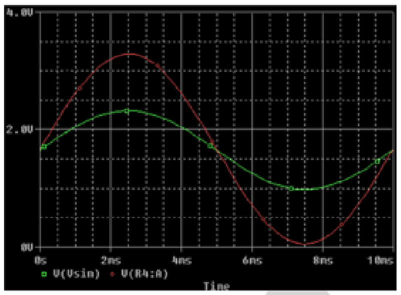
3/22/2015 6

Workshop 2 Slide 6



Analog Signal Conditioner

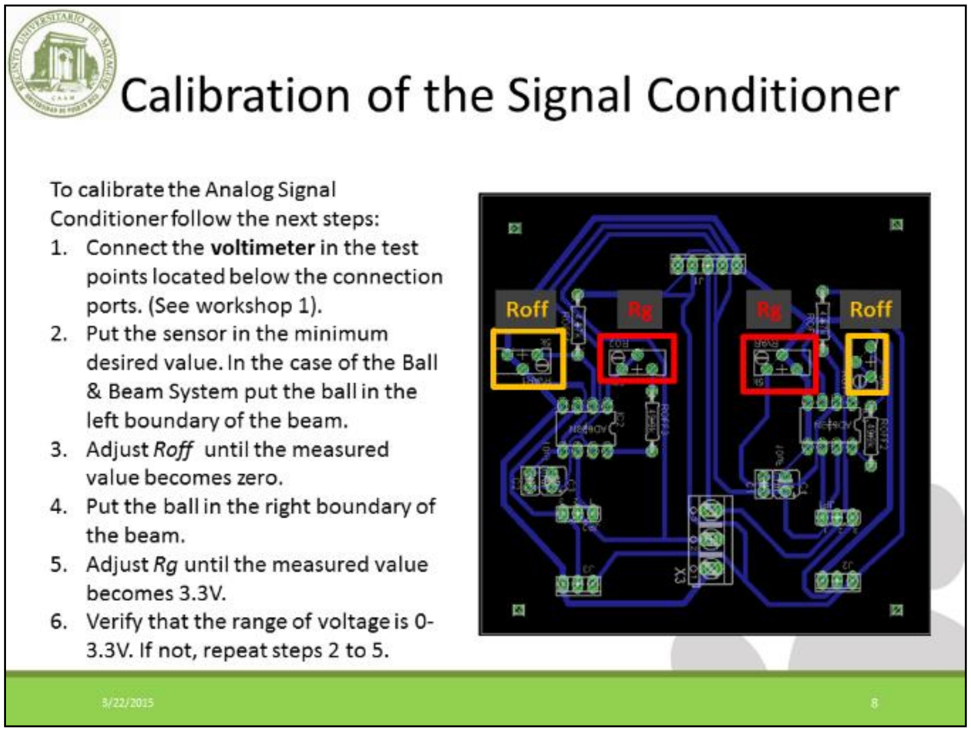




$$Vout = \left(1 + \frac{100K}{49.9K + R_G}\right)\left(Vin - \frac{R_{off} * 3.3}{R_{off} + 4.87K}\right) = \mathbf{2.4(}\boldsymbol{Vin} \mathbf{- 0.9625)}$$

3/22/2015 7

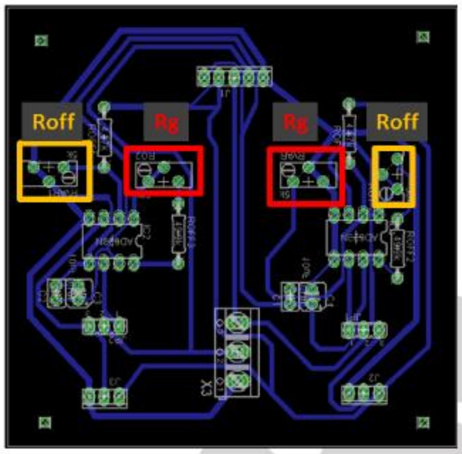
Workshop 2 Slide 7



Calibration of the Signal Conditioner

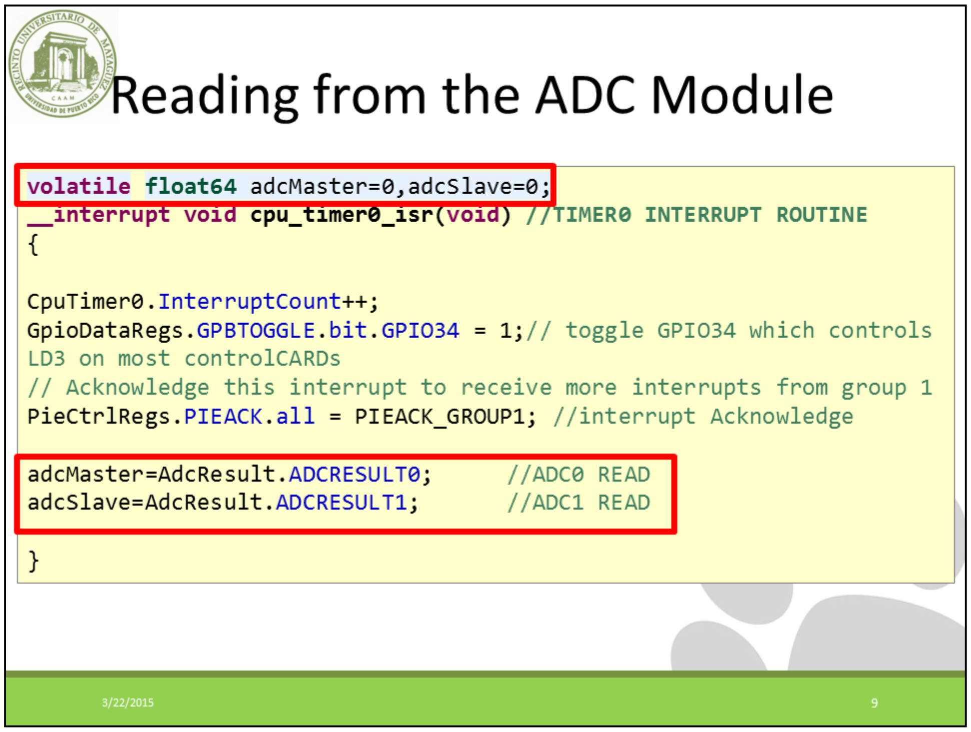
To calibrate the Analog Signal Conditioner follow the next steps:

1. Connect the **voltimeter** in the test points located below the connection ports. (See workshop 1).
2. Put the sensor in the minimum desired value. In the case of the Ball & Beam System put the ball in the left boundary of the beam.
3. Adjust *Roff* until the measured value becomes zero.
4. Put the ball in the right boundary of the beam.
5. Adjust *Rg* until the measured value becomes 3.3V.
6. Verify that the range of voltage is 0-3.3V. If not, repeat steps 2 to 5.



3/22/2015 8

Workshop 2 Slide 8

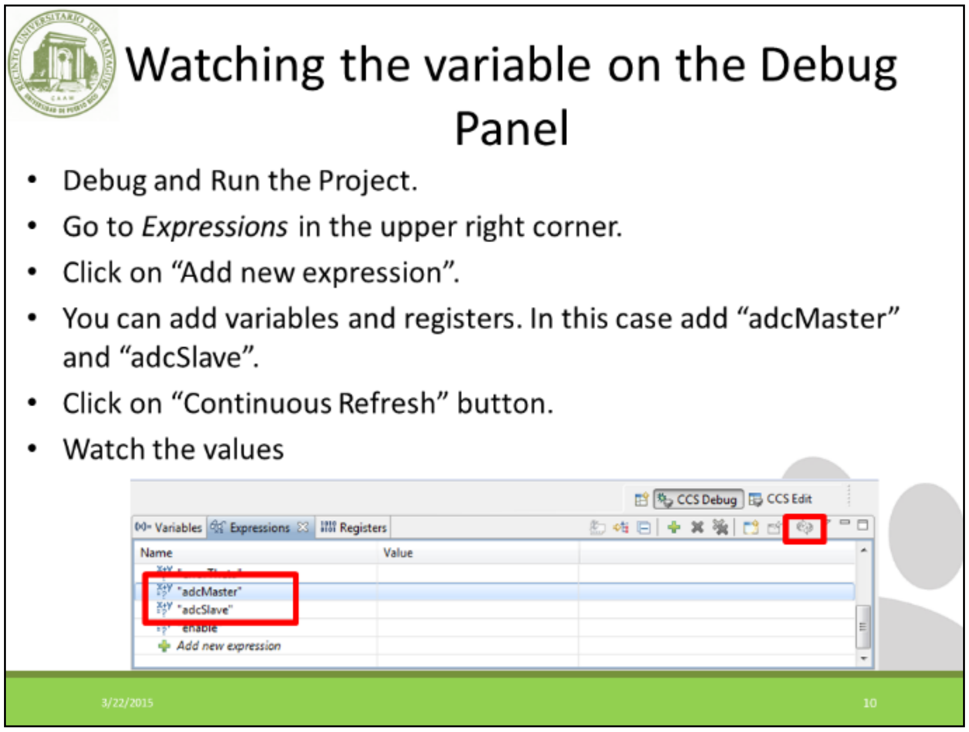


Reading from the ADC Module

```
volatile float64 adcMaster=0,adcSlave=0;
__interrupt void cpu_timer0_isr(void) //TIMER0 INTERRUPT ROUTINE
{

CpuTimer0.InterruptCount++;
GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;// toggle GPIO34 which controls
LD3 on most controlCARDs
// Acknowledge this interrupt to receive more interrupts from group 1
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //interrupt Acknowledge

adcMaster=AdcResult.ADCRESULT0;       //ADC0 READ
adcSlave=AdcResult.ADCRESULT1;        //ADC1 READ

}
```

3/22/2015 9

Workshop 2 Slide 9

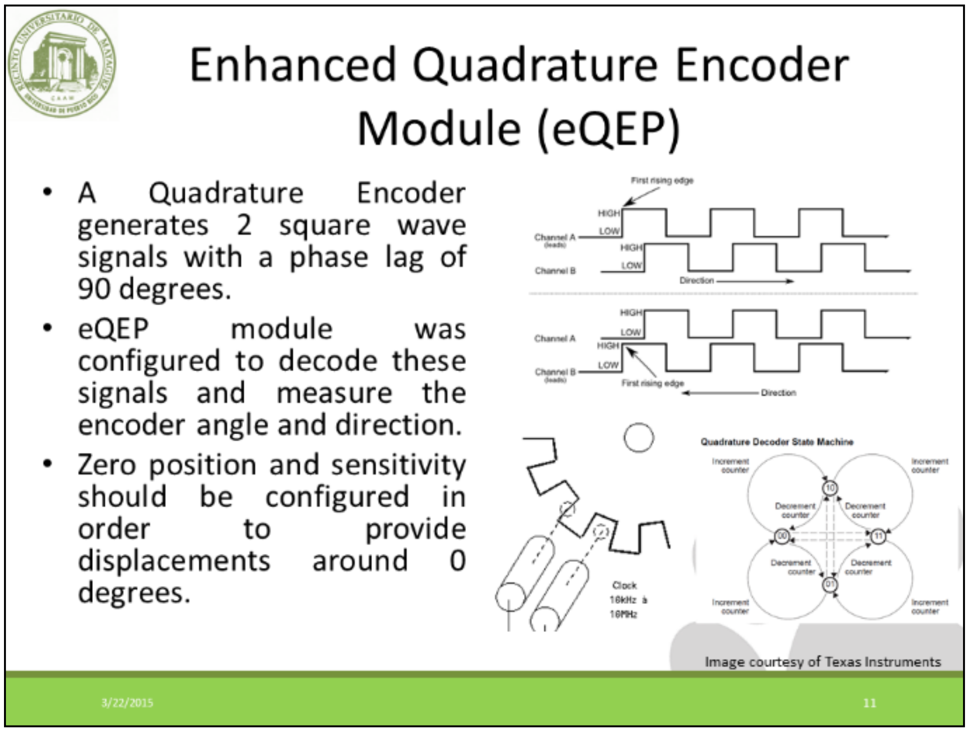


Watching the variable on the Debug Panel

- Debug and Run the Project.
- Go to *Expressions* in the upper right corner.
- Click on “Add new expression”.
- You can add variables and registers. In this case add “adcMaster” and “adcSlave”.
- Click on “Continuous Refresh” button.
- Watch the values

3/22/2015 10

Workshop 2 Slide 10



Enhanced Quadrature Encoder Module (eQEP)

- A Quadrature Encoder generates 2 square wave signals with a phase lag of 90 degrees.
- eQEP module was configured to decode these signals and measure the encoder angle and direction.
- Zero position and sensitivity should be configured in order to provide displacements around 0 degrees.

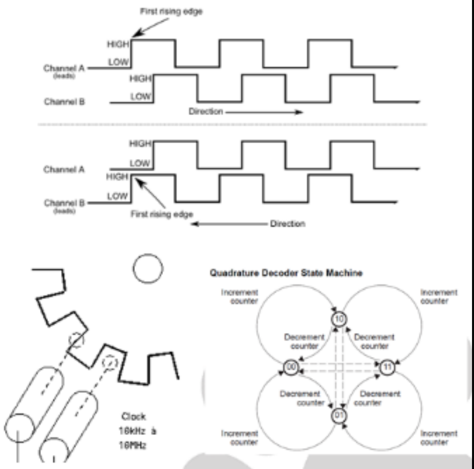
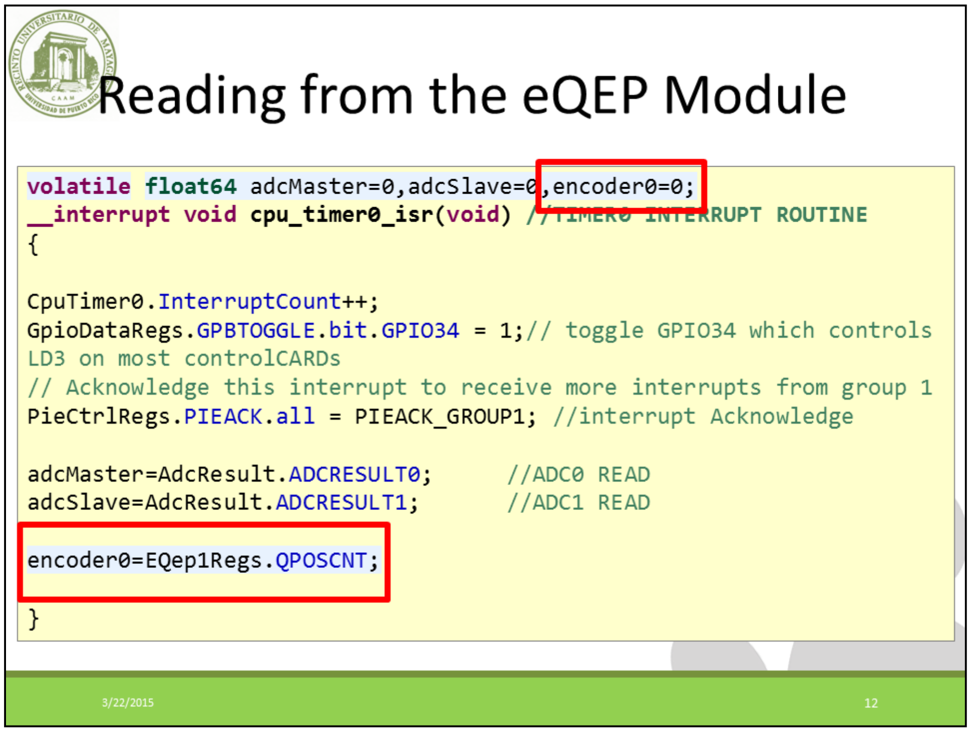


Image courtesy of Texas Instruments

3/22/2015 11

Workshop 2 Slide 11

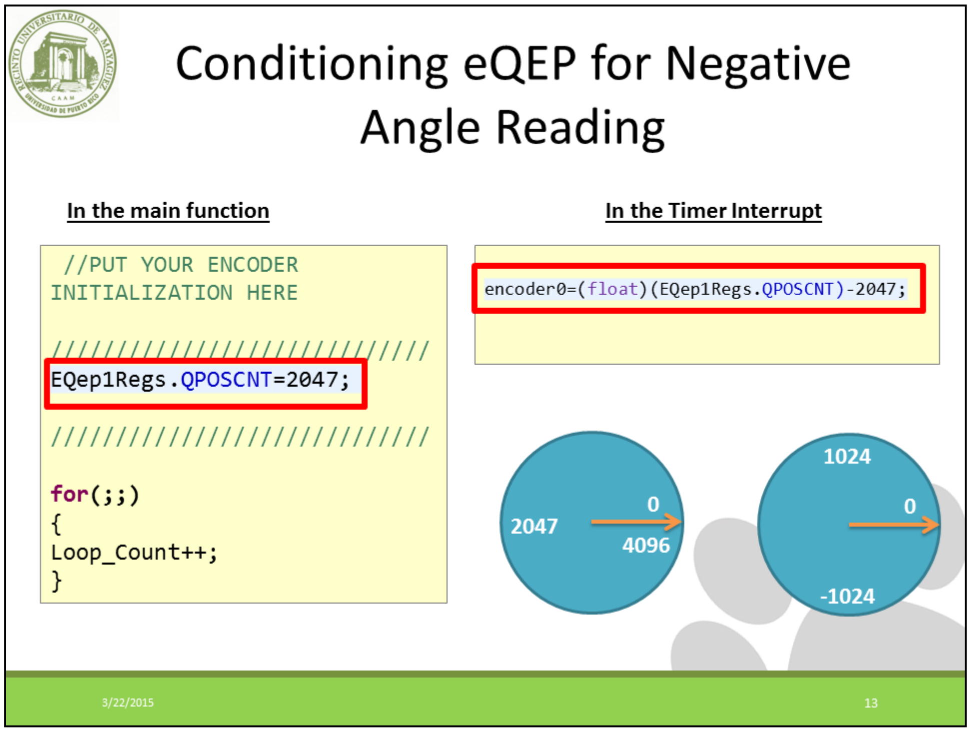


Reading from the eQEP Module

```
volatile float64 adcMaster=0,adcSlave=0,encoder0=0;
__interrupt void cpu_timer0_isr(void) //TIMER0 INTERRUPT ROUTINE
{

CpuTimer0.InterruptCount++;
GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;// toggle GPIO34 which controls
LD3 on most controlCARDs
// Acknowledge this interrupt to receive more interrupts from group 1
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //interrupt Acknowledge

adcMaster=AdcResult.ADCRESULT0;      //ADC0 READ
adcSlave=AdcResult.ADCRESULT1;       //ADC1 READ

encoder0=EQep1Regs.QPOSCNT;

}
```

3/22/2015 12

Workshop 2 Slide 12



Conditioning eQEP for Negative Angle Reading

In the main function

```
 //PUT YOUR ENCODER
INITIALIZATION HERE

/////////////////////////////
EQep1Regs.QPOSCNT=2047;
/////////////////////////////

for(;;)
{
Loop_Count++;
}
```

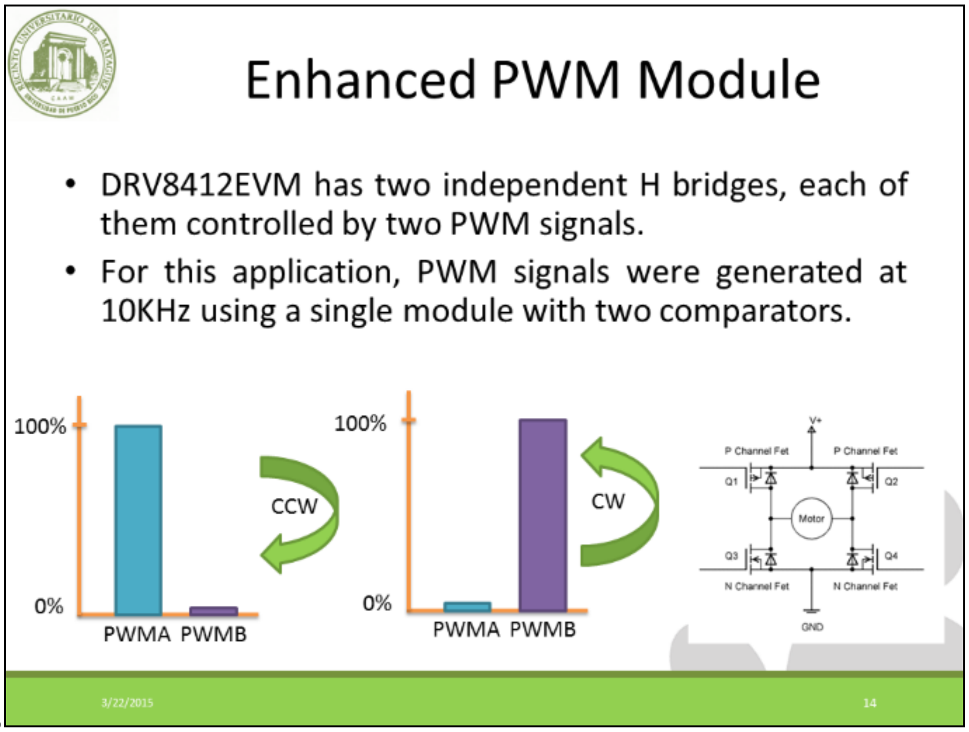
In the Timer Interrupt

```
encoder0=(float)(EQep1Regs.QPOSCNT)-2047;
```

2047
0
4096
1024
0
-1024

3/22/2015 13

Workshop 2 Slide 13



Enhanced PWM Module

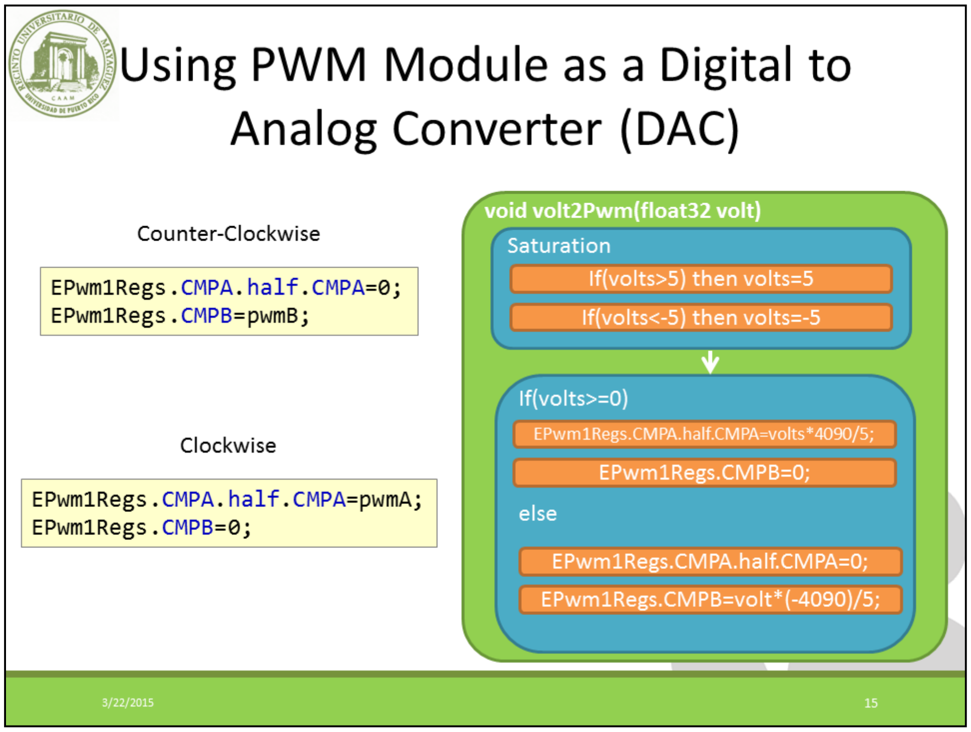
- DRV8412EVM has two independent H bridges, each of them controlled by two PWM signals.
- For this application, PWM signals were generated at 10KHz using a single module with two comparators.

100%
0%
PWMA PWMB
CCW
100%
0%
PWMA PWMB
CW
V+
P Channel Fet
P Channel Fet
Q1
Q2
Motor
Q3
Q4
N Channel Fet
N Channel Fet
GND

3/22/2015 14

9,5

Workshop 2 Slide 14



Using PWM Module as a Digital to Analog Converter (DAC)

Counter-Clockwise

```
EPwm1Regs.CMPA.half.CMPA=0;
EPwm1Regs.CMPB=pwmB;
```

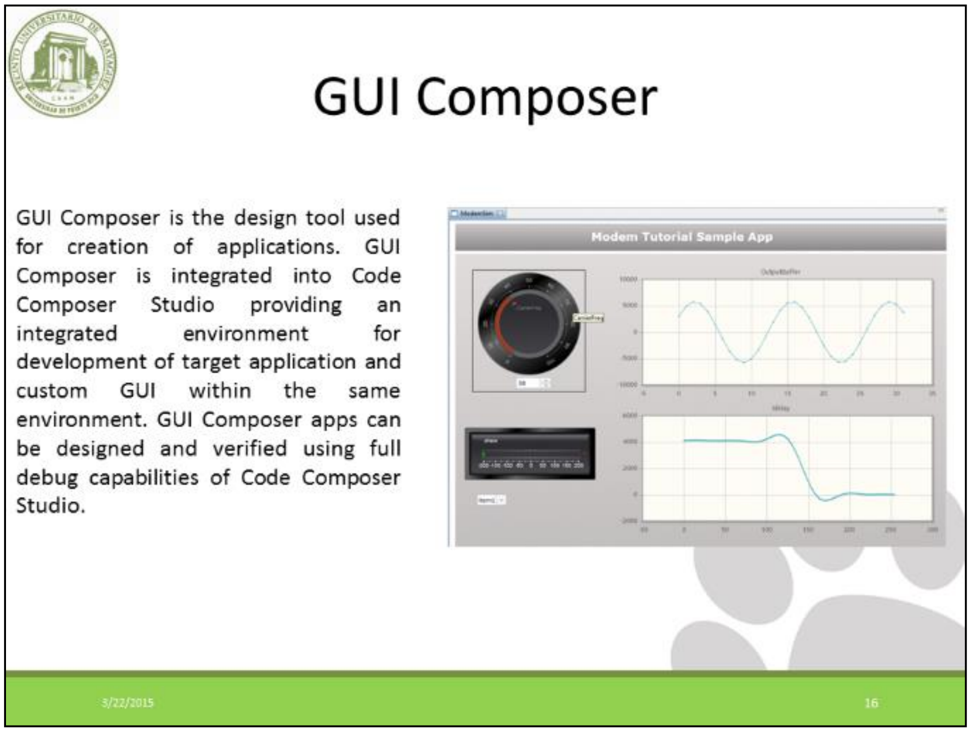
Clockwise

```
EPwm1Regs.CMPA.half.CMPA=pwmA;
EPwm1Regs.CMPB=0;
```

```
void volt2Pwm(float32 volt)
    Saturation
        If(volts>5) then volts=5
        If(volts<-5) then volts=-5
    If(volts>=0)
        EPwm1Regs.CMPA.half.CMPA=volts*4090/5;
        EPwm1Regs.CMPB=0;
    else
        EPwm1Regs.CMPA.half.CMPA=0;
        EPwm1Regs.CMPB=volt*(-4090)/5;
```

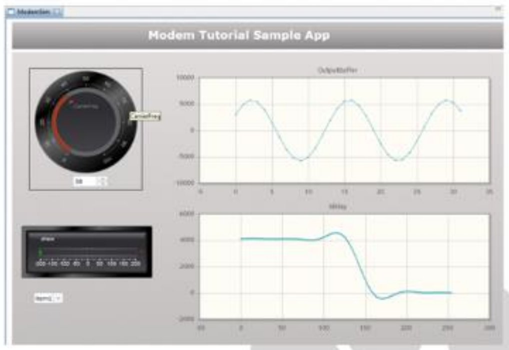
3/22/2015 15

Workshop 2 Slide 15



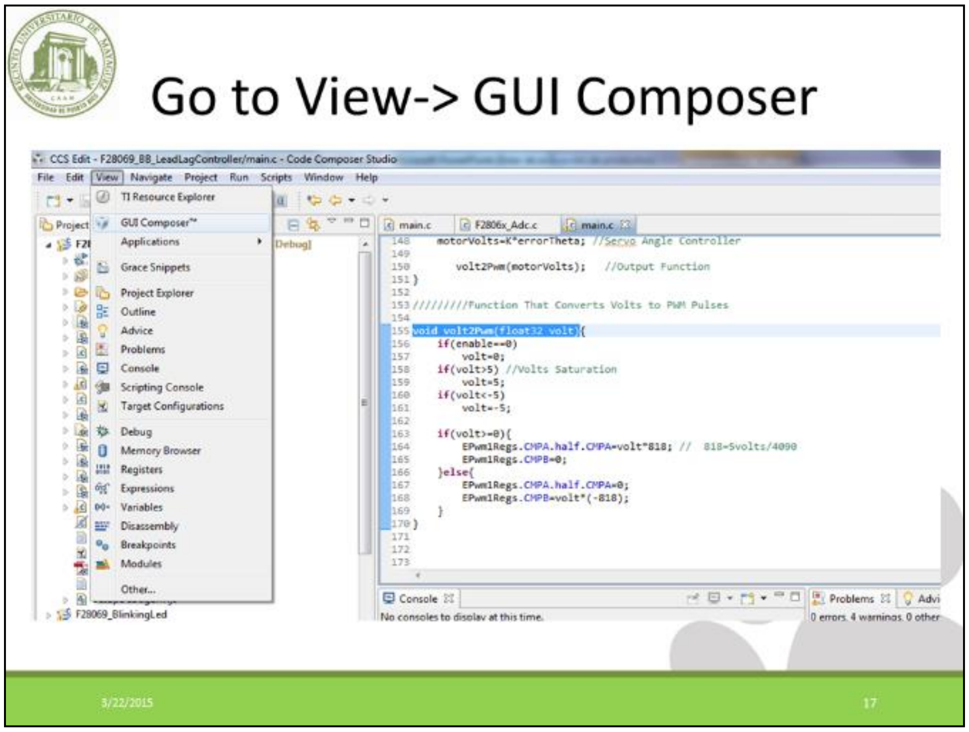
GUI Composer

GUI Composer is the design tool used for creation of applications. GUI Composer is integrated into Code Composer Studio providing an integrated environment for development of target application and custom GUI within the same environment. GUI Composer apps can be designed and verified using full debug capabilities of Code Composer Studio.



3/22/2015 16

Workshop 2 Slide 16



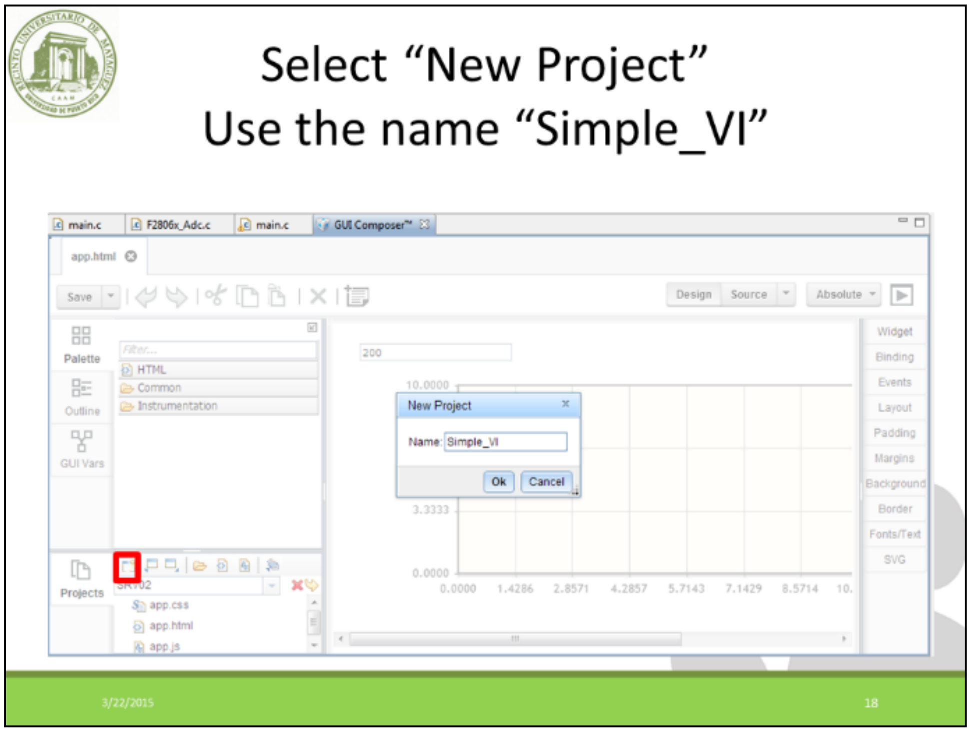
Go to View-> GUI Composer

```
    motorVolts=K*errorTheta; //Servo Angle Controller

        volt2Pwm(motorVolts);   //Output Function
}

/////////Function That Converts Volts to PWM Pulses

void volt2Pwm(float32 volt){
    if(enable==0)
        volt=0;
    if(volt>5) //Volts Saturation
        volt=5;
    if(volt<-5)
        volt=-5;

    if(volt>=0){
        EPwm1Regs.CMPA.half.CMPA=volt*818; //  818=5volts/4090
        EPwm1Regs.CMPB=0;
    }else{
        EPwm1Regs.CMPA.half.CMPA=0;
        EPwm1Regs.CMPB=volt*(-818);
    }
}
```

No consoles to display at this time.

0 errors, 4 warnings, 0 other

Workshop 2 Slide 17

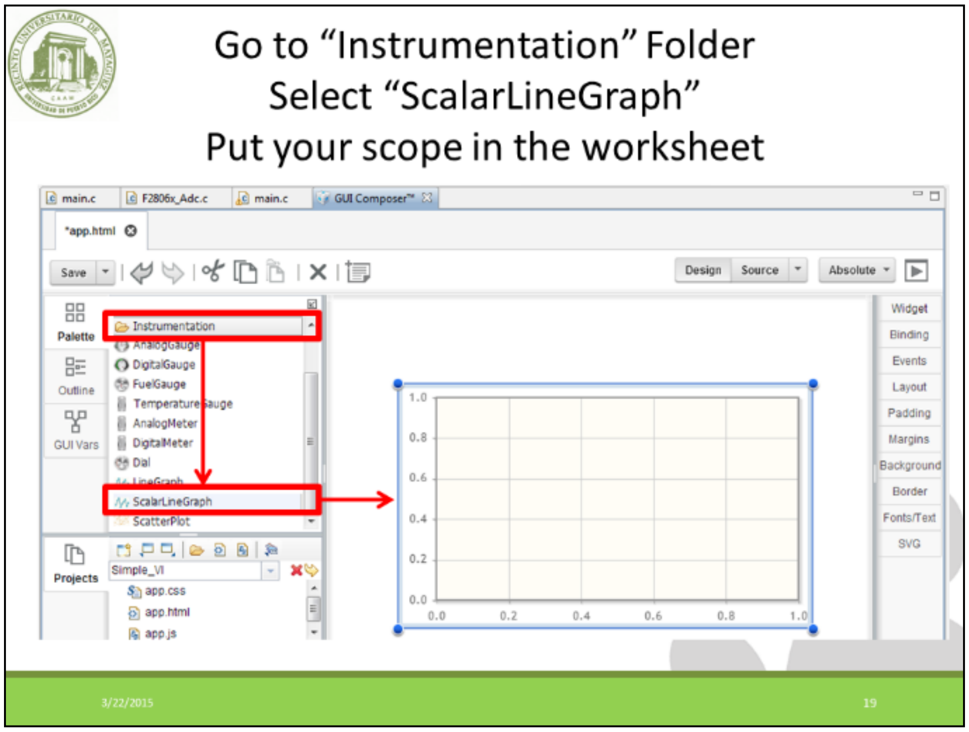


Select "New Project"
Use the name "Simple_VI"

New Project

Name: Simple_VI

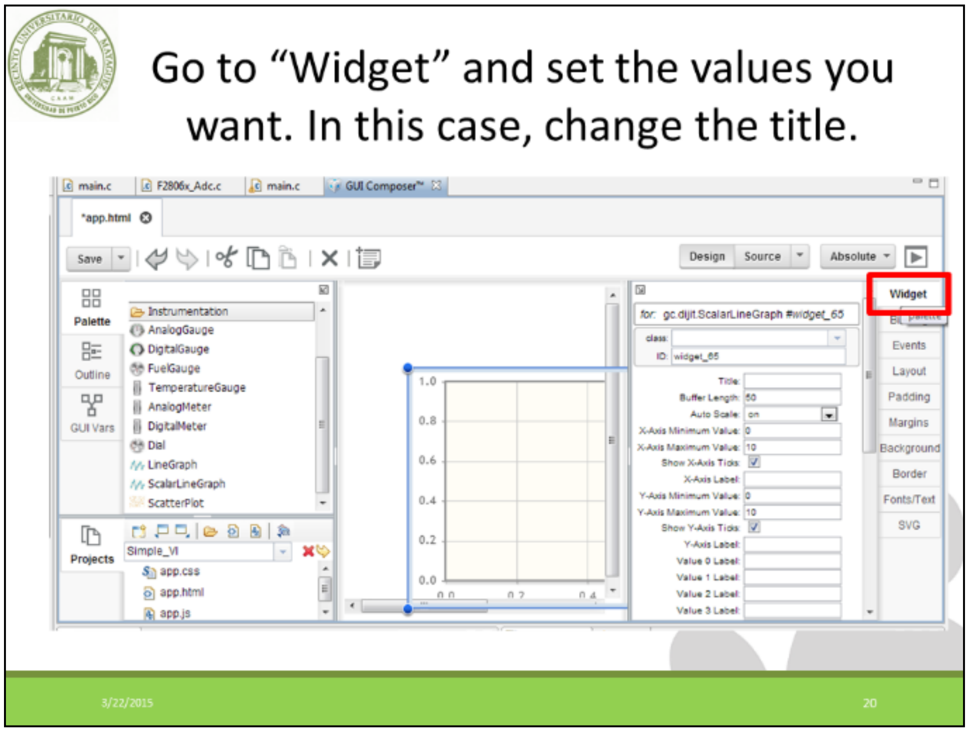
Workshop 2 Slide 18



Go to "Instrumentation" Folder Select "ScalarLineGraph" Put your scope in the worksheet

3/22/2015 19

Workshop 2 Slide 19



Go to "Widget" and set the values you want. In this case, change the title.

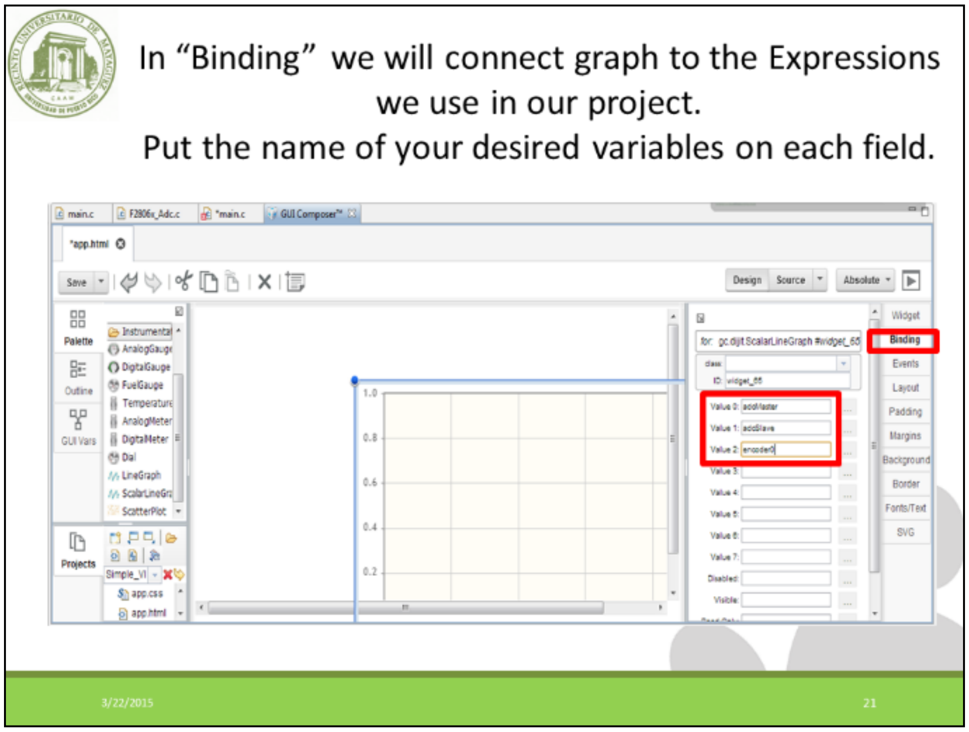
for: gc.dijit.ScalarLineGraph #widget_65

class:
ID: widget_65

Title:
Buffer Length: 50
Auto Scale: on
X-Axis Minimum Value: 0
X-Axis Maximum Value: 10
Show X-Axis Ticks: ✓
X-Axis Label:
Y-Axis Minimum Value: 0
Y-Axis Maximum Value: 10
Show Y-Axis Ticks: ✓
Y-Axis Label:
Value 0 Label:
Value 1 Label:
Value 2 Label:
Value 3 Label:

3/22/2015 20

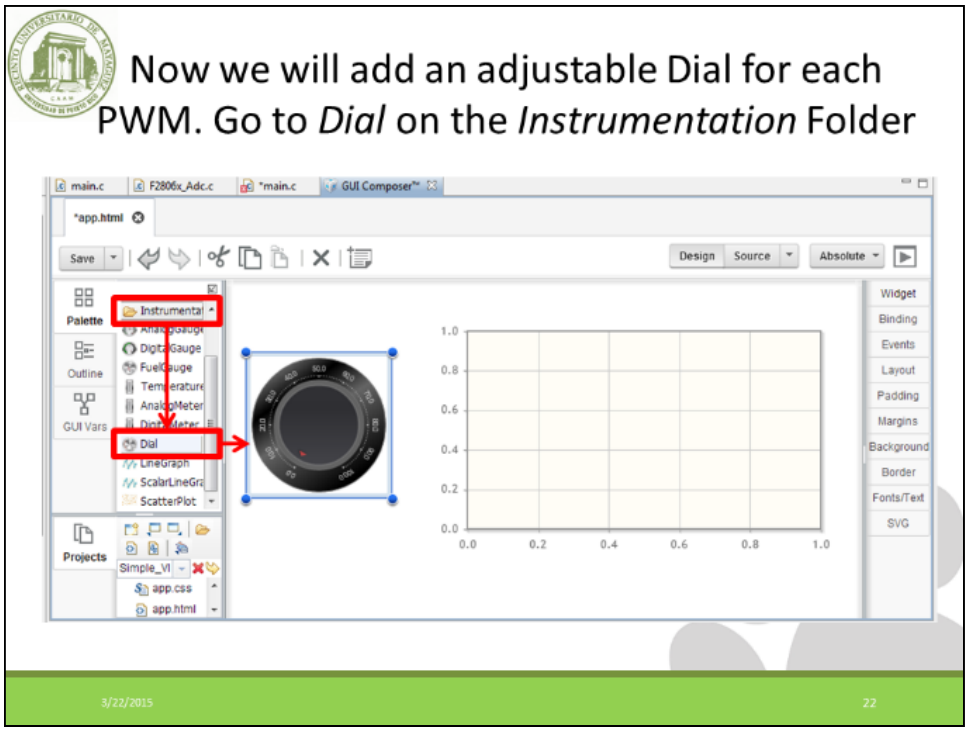
Workshop 2 Slide 20



In "Binding" we will connect graph to the Expressions we use in our project.
Put the name of your desired variables on each field.

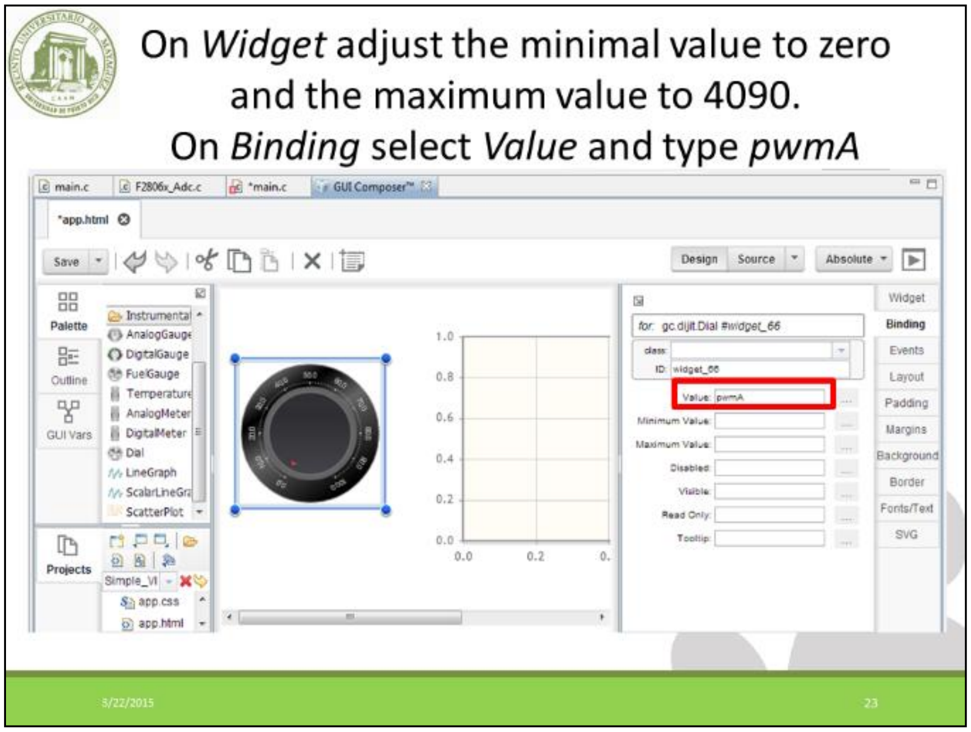
for: gc.dijit.ScalarLineGraph #widget_65
ID: widget_65
Value 0: addMaster
Value 1: addSlave
Value 2: encoder0

Workshop 2 Slide 21



Now we will add an adjustable Dial for each PWM. Go to *Dial* on the *Instrumentation* Folder

Workshop 2 Slide 22



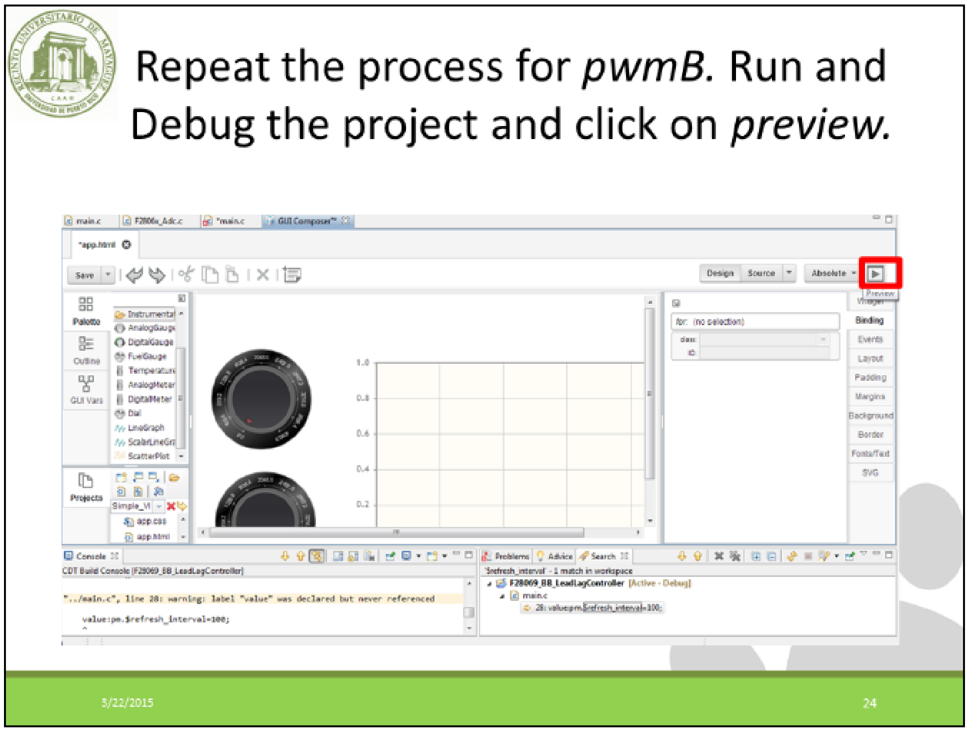
On *Widget* adjust the minimal value to zero and the maximum value to 4090. On *Binding* select *Value* and type *pwmA*

for: gc.dijit.Dial #widget_66

ID: widget_66

Value: pwmA

Workshop 2 Slide 23



Repeat the process for *pwmB*. Run and Debug the project and click on *preview*.

for: (no selection)

CDT Build Console [F28069_BB_LeadLagController]

```
"../main.c", line 28: warning: label "value" was declared but never referenced

    value:pm.$refresh_interval=100;
    ^
```

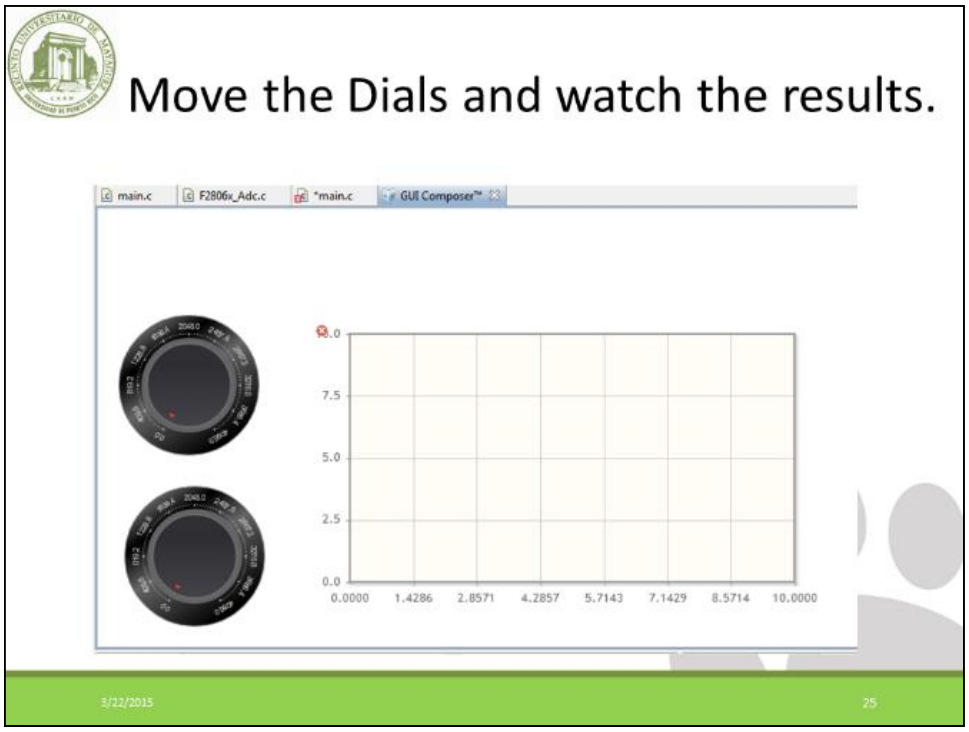
'$refresh_interval' - 1 match in workspace

F28069_BB_LeadLagController [Active - Debug]

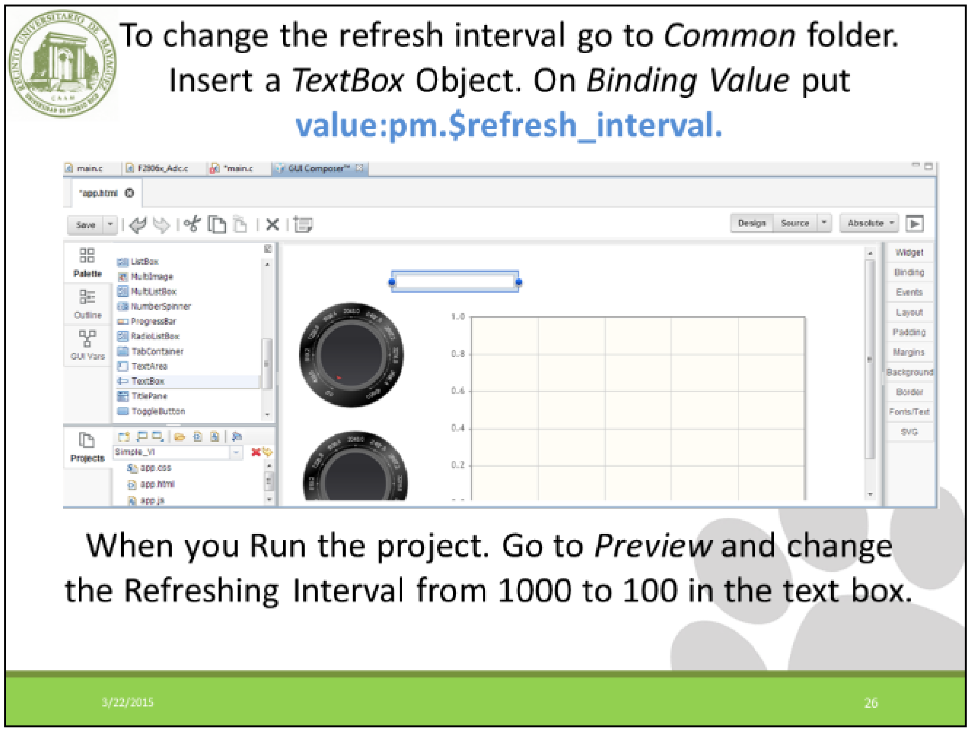
main.c

28: value:pm.$refresh_interval=100;

Workshop 2 Slide 24



Workshop 2 Slide 25



To change the refresh interval go to *Common* folder. Insert a *TextBox* Object. On *Binding Value* put value:pm.$refresh_interval.

When you Run the project. Go to *Preview* and change the Refreshing Interval from 1000 to 100 in the text box.

Workshop 2 Slide 26

4. Workshop 3 Presentation

Workshop 3: Implementation of a Digital Controller on an MCU

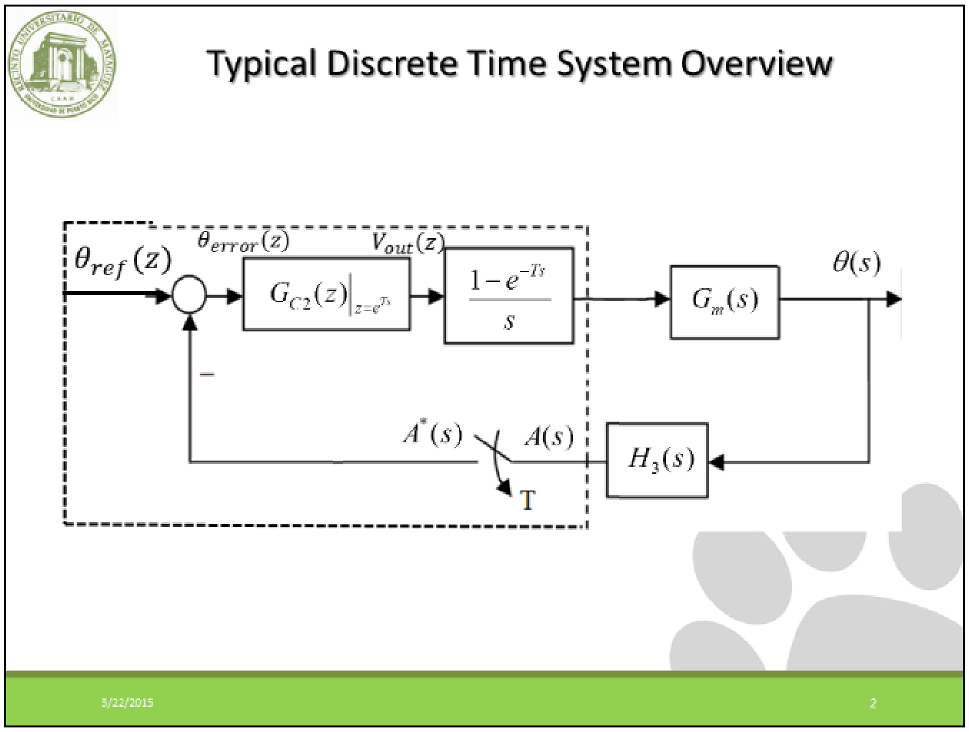
University of Puerto Rico at Mayaguez

Departamento de Ingeniería Eléctrica y Computadoras

Juan Felipe Patarroyo Montenegro.

3/22/2015 1

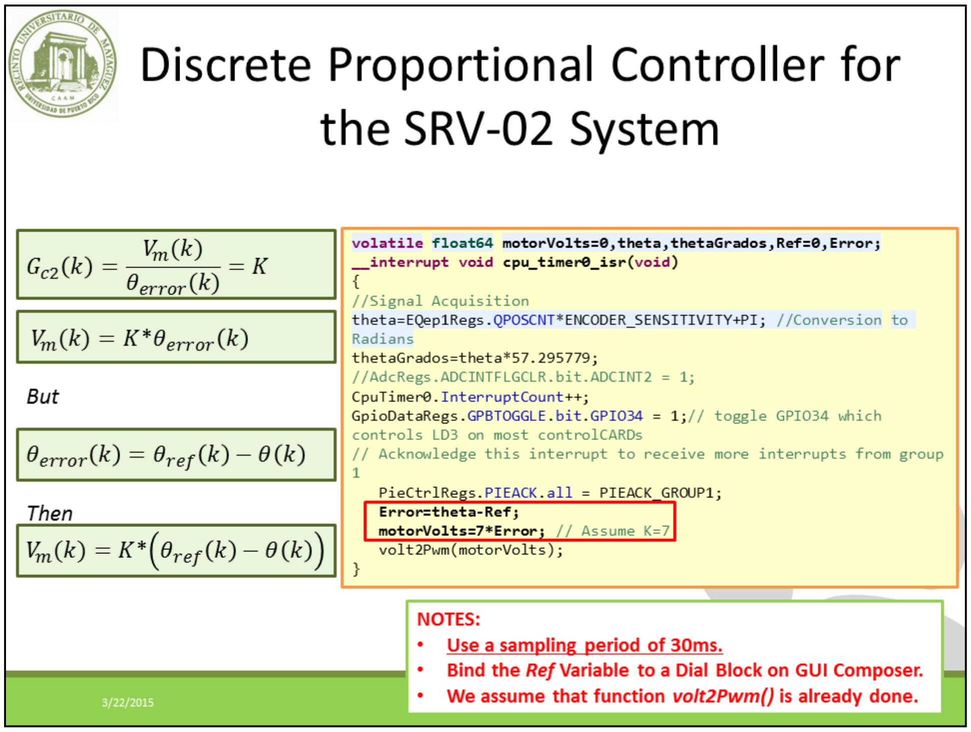
Workshop 3 Slide 1



Typical Discrete Time System Overview

3/22/2015 2

Workshop 3 Slide 2



Discrete Proportional Controller for the SRV-02 System

$$G_{c2}(k) = \frac{V_m(k)}{\theta_{error}(k)} = K$$

$$V_m(k) = K^*\theta_{error}(k)$$

But

$$\theta_{error}(k) = \theta_{ref}(k) - \theta(k)$$

Then

$$V_m(k) = K^*\left(\theta_{ref}(k) - \theta(k)\right)$$

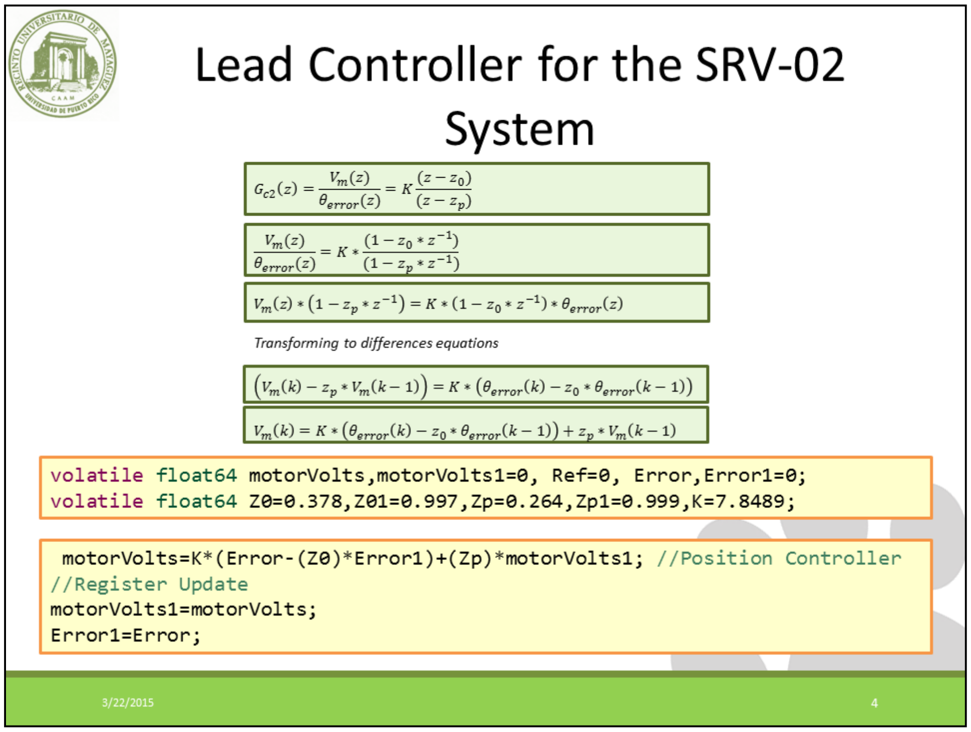
```
volatile float64 motorVolts=0,theta,thetaGrados,Ref=0,Error;
__interrupt void cpu_timer0_isr(void)
{
//Signal Acquisition
theta=EQep1Regs.QPOSCNT*ENCODER_SENSITIVITY+PI; //Conversion to
Radians
thetaGrados=theta*57.295779;
//AdcRegs.ADCINTFLGCLR.bit.ADCINT2 = 1;
CpuTimer0.InterruptCount++;
GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1;// toggle GPIO34 which
controls LD3 on most controlCARDs
// Acknowledge this interrupt to receive more interrupts from group
1
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
   Error=theta-Ref;
   motorVolts=7*Error; // Assume K=7
   volt2Pwm(motorVolts);
}
```

NOTES:

- **Use a sampling period of 30ms.**
- **Bind the *Ref* Variable to a Dial Block on GUI Composer.**
- **We assume that function *volt2Pwm()* is already done.**

3/22/2015

Workshop 3 Slide 3



Lead Controller for the SRV-02 System

$$G_{c2}(z) = \frac{V_m(z)}{\theta_{error}(z)} = K\frac{(z - z_0)}{(z - z_p)}$$

$$\frac{V_m(z)}{\theta_{error}(z)} = K * \frac{(1 - z_0 * z^{-1})}{(1 - z_p * z^{-1})}$$

$$V_m(z) * \left(1 - z_p * z^{-1}\right) = K * (1 - z_0 * z^{-1}) * \theta_{error}(z)$$

Transforming to differences equations

$$\left(V_m(k) - z_p * V_m(k-1)\right) = K * \left(\theta_{error}(k) - z_0 * \theta_{error}(k-1)\right)$$

$$V_m(k) = K * \left(\theta_{error}(k) - z_0 * \theta_{error}(k-1)\right) + z_p * V_m(k-1)$$

```
volatile float64 motorVolts,motorVolts1=0, Ref=0, Error,Error1=0;
volatile float64 Z0=0.378,Z01=0.997,Zp=0.264,Zp1=0.999,K=7.8489;
```

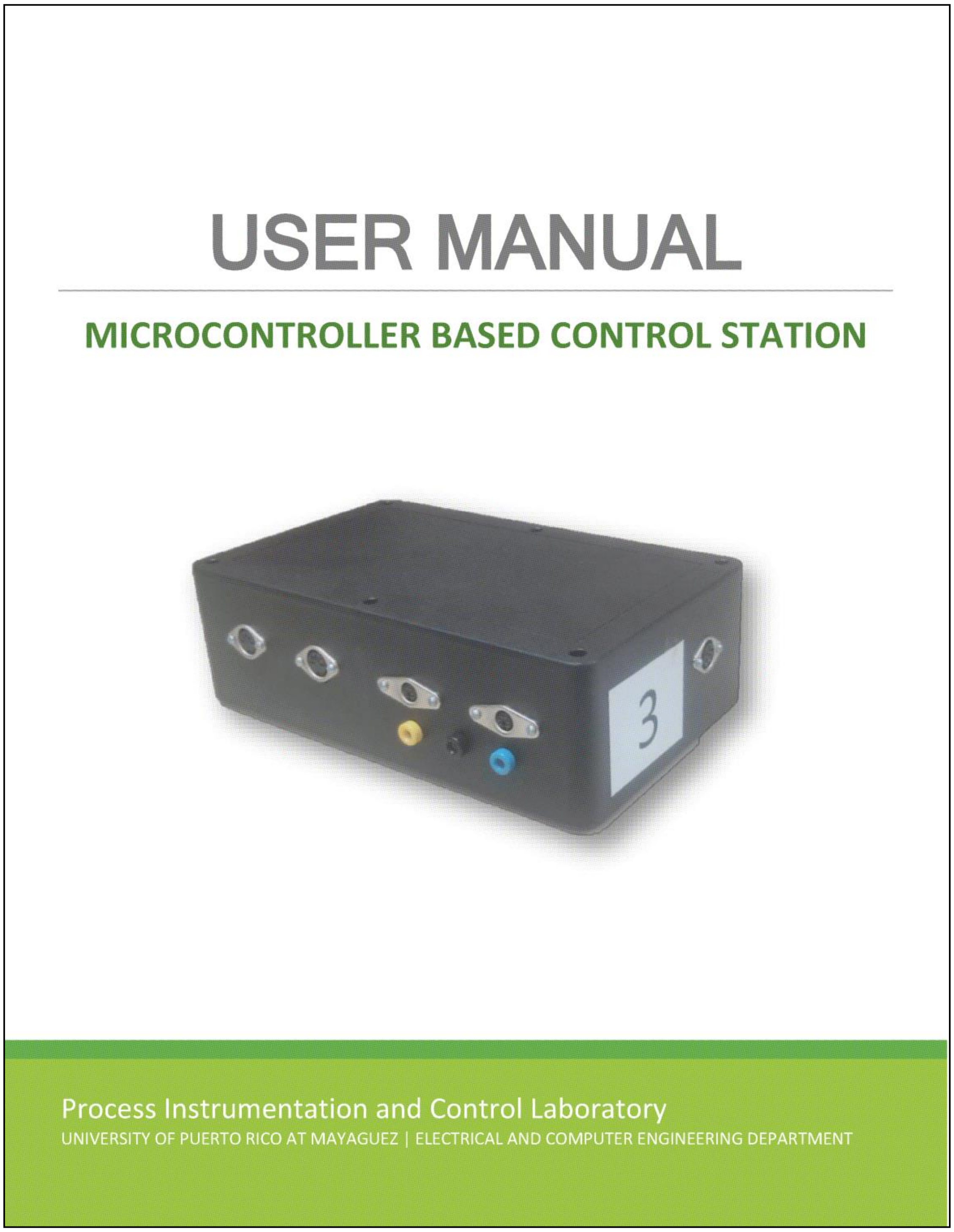
```
 motorVolts=K*(Error-(Z0)*Error1)+(Zp)*motorVolts1; //Position Controller
//Register Update
motorVolts1=motorVolts;
Error1=Error;
```

3/22/2015 4

Workshop 3 Slide 4

Appendix E: Documentation Package

This section presents the User Manual for the MCS.



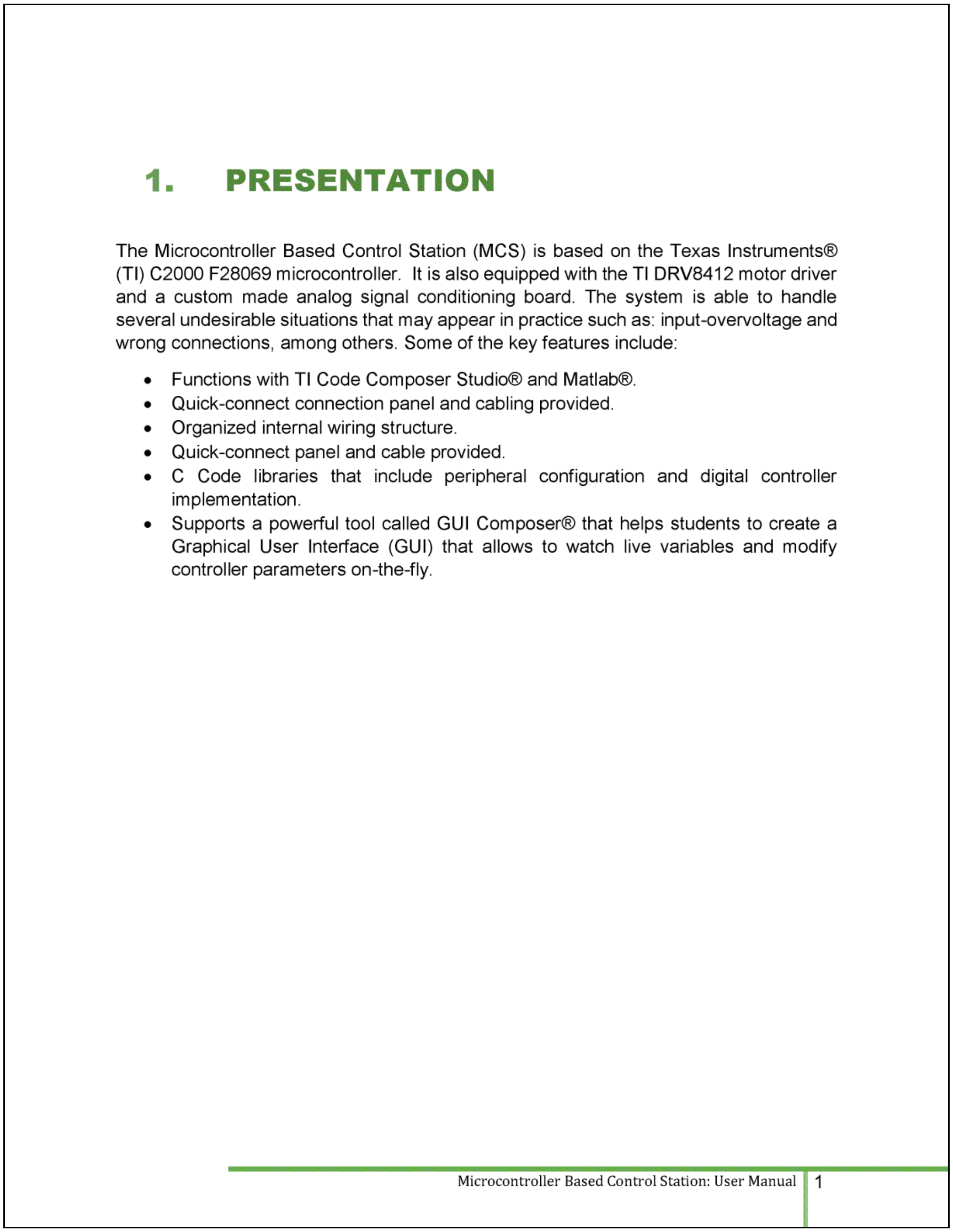
USER MANUAL

MICROCONTROLLER BASED CONTROL STATION

Process Instrumentation and Control Laboratory

UNIVERSITY OF PUERTO RICO AT MAYAGUEZ | ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

User Manual Page 1

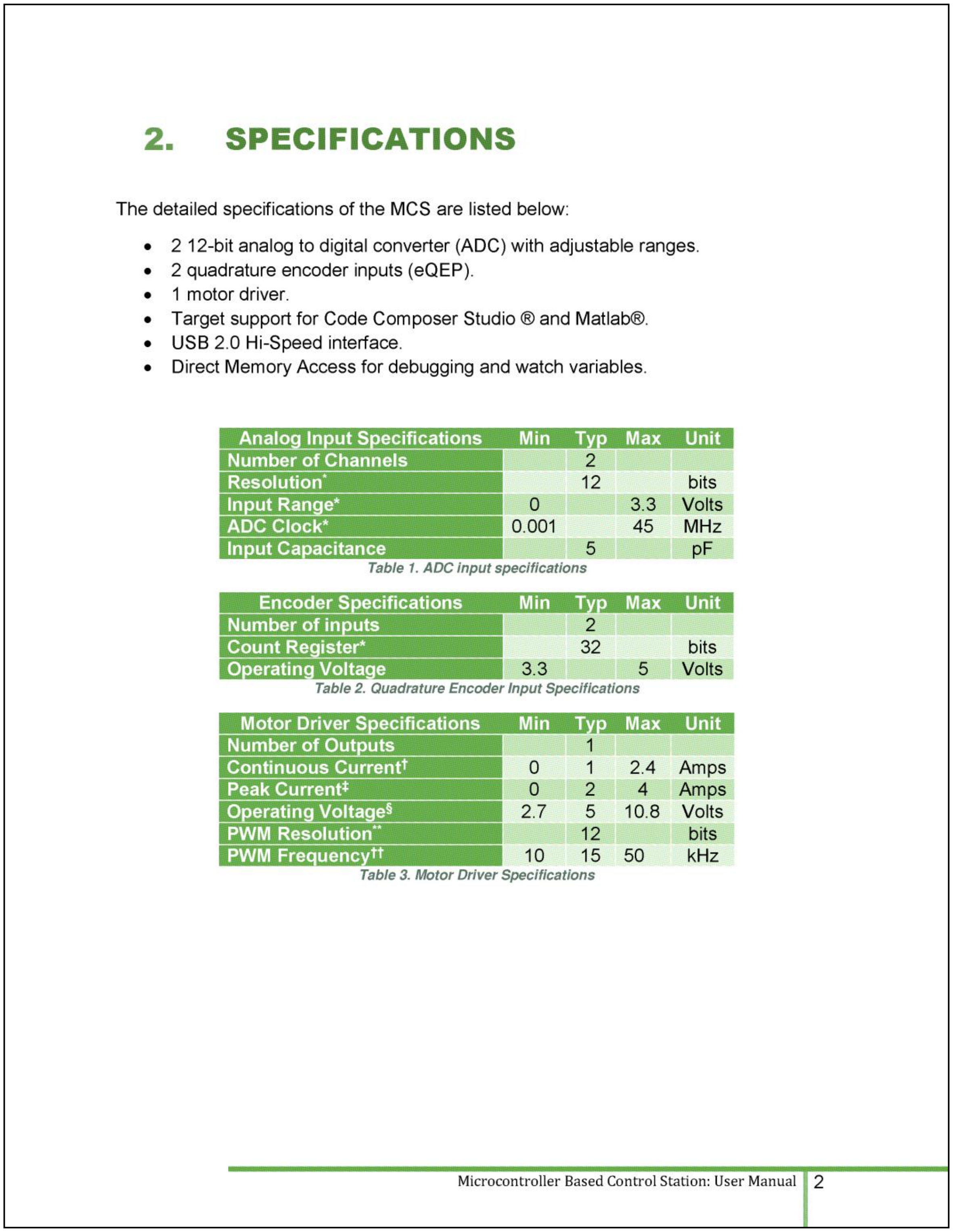


1. PRESENTATION

The Microcontroller Based Control Station (MCS) is based on the Texas Instruments® (TI) C2000 F28069 microcontroller. It is also equipped with the TI DRV8412 motor driver and a custom made analog signal conditioning board. The system is able to handle several undesirable situations that may appear in practice such as: input-overvoltage and wrong connections, among others. Some of the key features include:

- Functions with TI Code Composer Studio® and Matlab®.
- Quick-connect connection panel and cabling provided.
- Organized internal wiring structure.
- Quick-connect panel and cable provided.
- C Code libraries that include peripheral configuration and digital controller implementation.
- Supports a powerful tool called GUI Composer® that helps students to create a Graphical User Interface (GUI) that allows to watch live variables and modify controller parameters on-the-fly.

User Manual Page 2



2. SPECIFICATIONS

The detailed specifications of the MCS are listed below:

- 2 12-bit analog to digital converter (ADC) with adjustable ranges.
- 2 quadrature encoder inputs (eQEP).
- 1 motor driver.
- Target support for Code Composer Studio ® and Matlab®.
- USB 2.0 Hi-Speed interface.
- Direct Memory Access for debugging and watch variables.

Analog Input Specifications	Min	Typ	Max	Unit
Number of Channels		2		
Resolution*		12		bits
Input Range*	0		3.3	Volts
ADC Clock*	0.001		45	MHz
Input Capacitance		5		pF

Table 1. ADC input specifications

Encoder Specifications	Min	Typ	Max	Unit
Number of inputs		2		
Count Register*		32		bits
Operating Voltage	3.3		5	Volts

Table 2. Quadrature Encoder Input Specifications

Motor Driver Specifications	Min	Typ	Max	Unit
Number of Outputs		1		
Continuous Current†	0	1	2.4	Amps
Peak Current‡	0	2	4	Amps
Operating Voltage§	2.7	5	10.8	Volts
PWM Resolution**		12		bits
PWM Frequency††	10	15	50	kHz

Table 3. Motor Driver Specifications

User Manual Page 3

3. COMPONENTS

3.1 General System Overview

A general system overview of the MCS is shown in Figure 1. The Analog Signal Conditioner is required to convert the voltage levels from the analog sensors to a range that could be read from the microcontroller (0-3.3V). A DC motor driver (DRV8412) was configured and used to drive the system motor from the PWM signals generated by the microcontroller. The Optical Encoder reads the angular position of the motor. Finally, a Timer Interrupt Unit sends a signal to the CPU indicating when the control action should be performed.

A C2000 F28069 microcontroller was used. It operates at 80MHz and is equipped 16 Analog-to-Digital Conversion (ADC) channels with a resolution of 12 bits, 2 Quadrature Encoder read-modules (eQep), 16 independent 32 bits PWM channels, Floating Point Unit (FPU), JTAG emulation tool and other characteristics that make this microcontroller ideal for high capacity control purposes. (For more information refer to the TMS320F28069 User Manual).

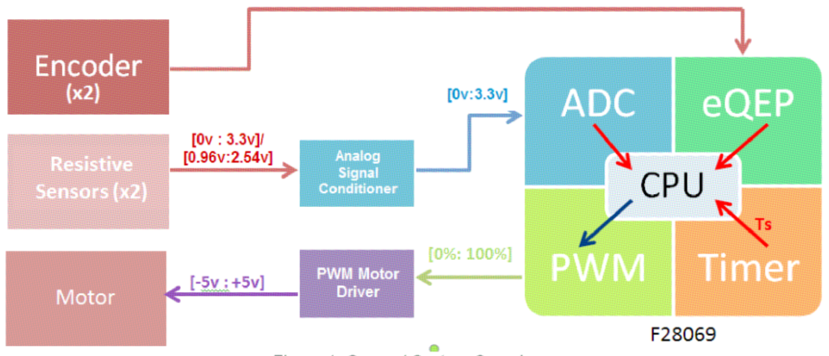


Figure 1. General System Overview

3.1.1 Analog Signal Conditioning Board

The MCS was designed to work with two *Quanser* analog sensors, these resistive sensors are designed to have a power supply of -12V to +12V. Using a couple of bias resistors, they convert this voltage level to -5V to +5V. For this reason, a Signal

User Manual Page 4

Conditioner Circuit should be used to convert these voltage levels to a range of 0V to 3.3V which is the operative range of the microcontroller ADC unit.

As seen on the left side of Figure 2, two 7KΩ bias resistors are used to transform the output range from -5V to +5V. This represents a disadvantage because the system should use a dual voltage power supply, making it less portable and incrementing costs of production and flexibility. The proposed solution was to use a single power supply of 3.3V on the resistive sensor; this will yield an output voltage range between 0.96V and 2.33V.

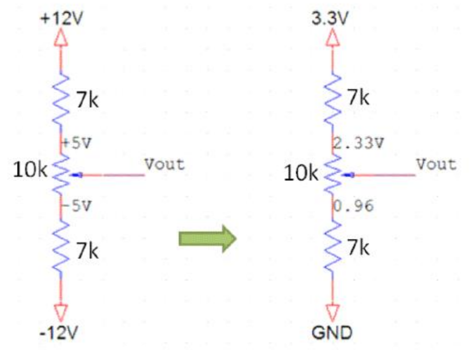


Figure 2. Quanser Resistive Sensors Voltage Levels. Left: Quanser Power Supply Voltage Levels. Right: Proposed Power Supply Voltage Levels.

Thus it is necessary to amplify and add an offset to this signal in order to read it with the microprocessor. The AD623AN instrumentation amplifier was used for this task. This amplifier may work using a single supply and its gain is set using only one resistor. Figure 3 shows the proposed analog instrumentation design.

The variable resistor R_G between pin 1 and the 49.9KΩ resistor changes the amplifier gain according to the manufacturer gain equation. The output voltage is given by

$$Vout = \left(1 + \frac{100K}{49.9K + R_G}\right)\left(Vin - \frac{R_{OFF} * 3.3}{R_{OFF} + 4.87K}\right) = \mathbf{2.4(Vin - 0.9625)}$$

User Manual Page 5

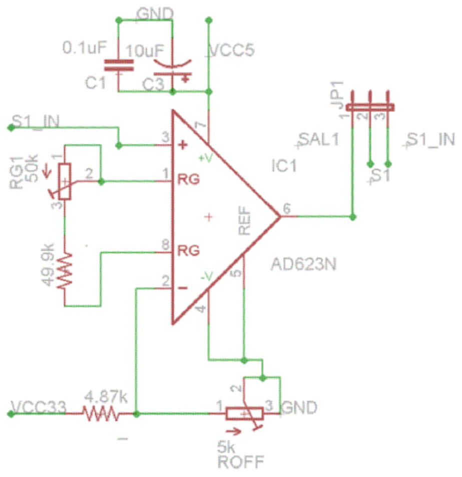


Figure 3. Analog Signal Conditioning Circuit

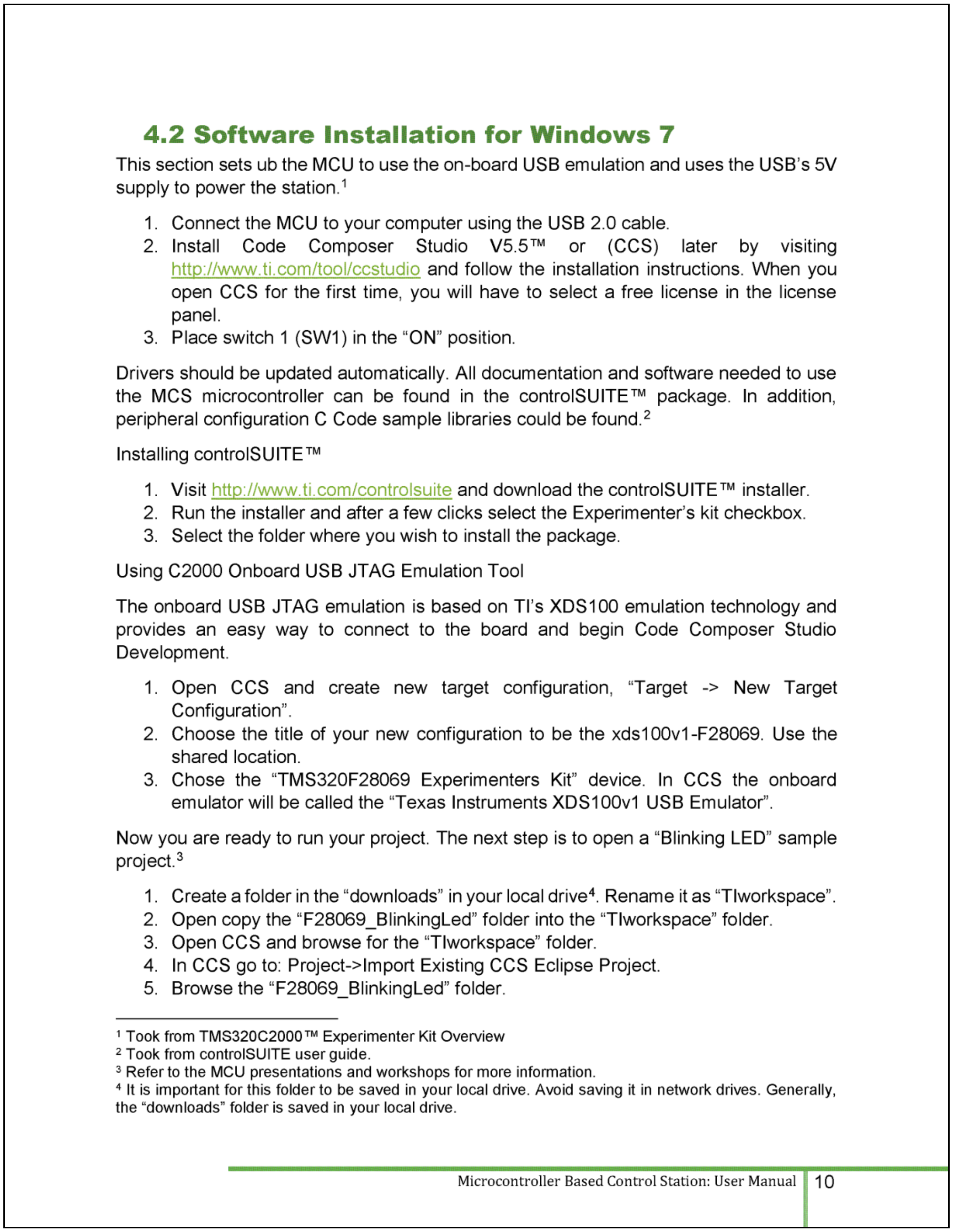
A voltage divider was used to provide an offset voltage of 0.96V on pin 2. The variable resistor R_{OFF} is used for calibration to minimize tolerance errors. The two capacitors between pin 7 and ground are used to isolate noise from the microcontroller and other digital circuits. Furthermore, a 3 pin jumper JP1 was used at the output to bypass the amplifier in the case when the sensors already work at the same voltage range of the microcontroller ADC and their signals do not need to be conditioned.

Additionally, the microcontroller development board also has all the protection circuitry to prevent from over-voltage or inverse-voltage. This is accomplished using voltage clippers at the input of the ADC pins of the microcontroller.

3.1.2 System Assembly

The MCU is packed in an enclosure with all the ports needed to make the connections with the Quanser Consulting Experiments and the power supply. This system should be able to handle different situations that may appear in practice (Overvoltage, bad connections, etc.). Figure 4 shows the internal box assembly. The MCS is organized so that the student may visually identify its main modules and manipulate them with little risk of damage.

User Manual Page 6



4.2 Software Installation for Windows 7

This section sets ub the MCU to use the on-board USB emulation and uses the USB's 5V supply to power the station.[1]

1. Connect the MCU to your computer using the USB 2.0 cable.
2. Install Code Composer Studio V5.5™ or (CCS) later by visiting http://www.ti.com/tool/ccstudio and follow the installation instructions. When you open CCS for the first time, you will have to select a free license in the license panel.
3. Place switch 1 (SW1) in the "ON" position.

Drivers should be updated automatically. All documentation and software needed to use the MCS microcontroller can be found in the controlSUITE™ package. In addition, peripheral configuration C Code sample libraries could be found.[2]

Installing controlSUITE™

1. Visit http://www.ti.com/controlsuite and download the controlSUITE™ installer.
2. Run the installer and after a few clicks select the Experimenter's kit checkbox.
3. Select the folder where you wish to install the package.

Using C2000 Onboard USB JTAG Emulation Tool

The onboard USB JTAG emulation is based on TI's XDS100 emulation technology and provides an easy way to connect to the board and begin Code Composer Studio Development.

1. Open CCS and create new target configuration, "Target -> New Target Configuration".
2. Choose the title of your new configuration to be the xds100v1-F28069. Use the shared location.
3. Chose the "TMS320F28069 Experimenters Kit" device. In CCS the onboard emulator will be called the "Texas Instruments XDS100v1 USB Emulator".

Now you are ready to run your project. The next step is to open a "Blinking LED" sample project.[3]

1. Create a folder in the "downloads" in your local drive[4]. Rename it as "TIworkspace".
2. Open copy the "F28069_BlinkingLed" folder into the "TIworkspace" folder.
3. Open CCS and browse for the "TIworkspace" folder.
4. In CCS go to: Project->Import Existing CCS Eclipse Project.
5. Browse the "F28069_BlinkingLed" folder.

[1] Took from TMS320C2000™ Experimenter Kit Overview
[2] Took from controlSUITE user guide.
[3] Refer to the MCU presentations and workshops for more information.
[4] It is important for this folder to be saved in your local drive. Avoid saving it in network drives. Generally, the "downloads" folder is saved in your local drive.

User Manual Page 7

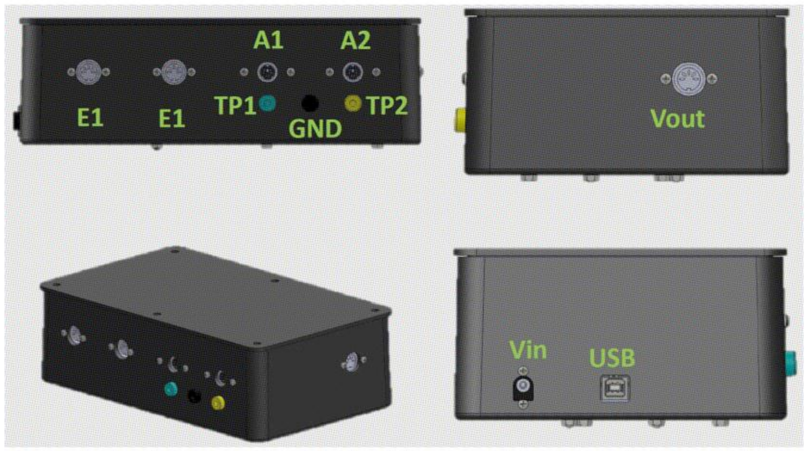


Figure 5. MCS Connection Ports

3.3 Connection Description

3.3.1 Motor Driver Output

The MCS has one 6-pin DIN motor driver output connector, shown in the top right side of the Figure 5 with ID Vout. The 12 bit output has a range of ±5V. Pin can drive up to 2.4A continuous and 4A peak. The motor driver pin-out is shown in Figure 6 when facing the front of the station.

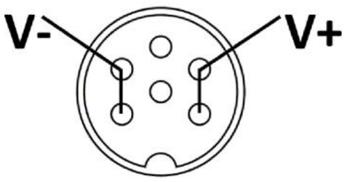


Figure 6. Motor Driver pin-out

User Manual Page 8

3.3.2 Analog Input

The MCS has two 6-pin mini DIM Analog Input Connectors, shown in the top left side of the Figure 5 with ID A1 and A2. The 12 bit input has a range of 0-3.3V. These ports are connected to the analog signal conditioning circuit board shown in section 3.1.1.The board can be configured to adjust the amplifier gain and offset signal. Also, a bypass jumper could be changed in case of the connected sensors operate at a full range of 0-3.3V. The analog input pin-out is shown in Figure 7 when facing the front of the module.

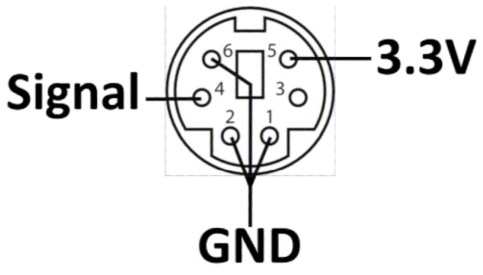


Figure 7. Analog input pin-out

Test points (ID TP1, TP2) are connected between the signal conditioning board and microcontroller ADC port (see Figure 1). These are used for gain and offset configuration. The analog ground connector (ID GND) is connected to the general ground plane. It is provided as an analog ground reference for testing purposes.

3.3.3. Encoder input

The MCS has two 5-pin DIN Encoder Input connectors, Figure 5 shows in the top left side the connector with ID E1, E2. Each encoder can provide a 32-bit count values and supports quadrature and non-quadrature (count and direction) modes. The encoder pin-out is shown in Figure 8 when facing the front of the module.

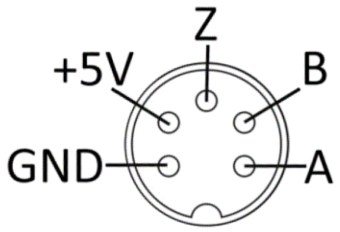
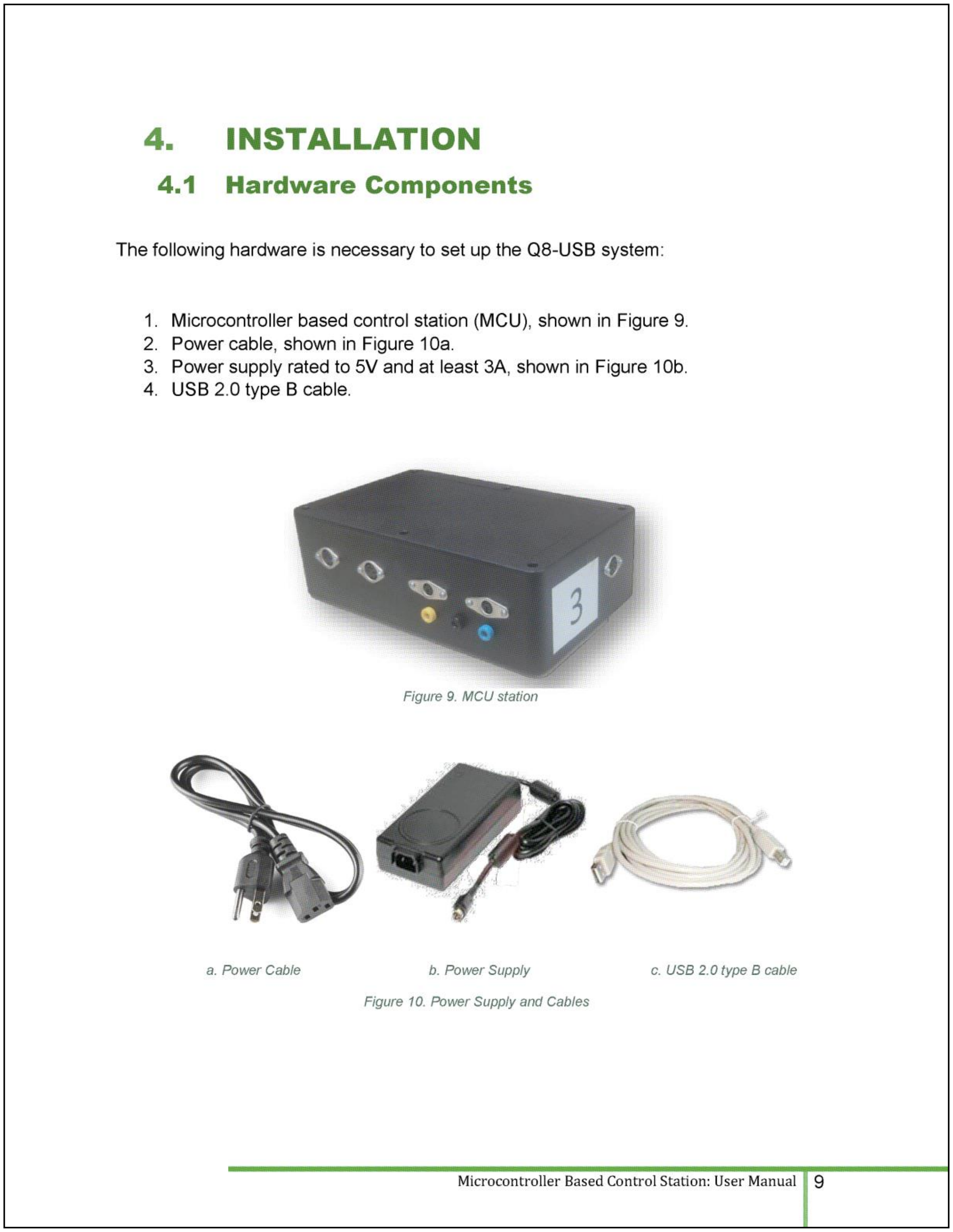


Figure 8. Encoder Input pin-out.

User Manual Page 9



4. INSTALLATION

4.1 Hardware Components

The following hardware is necessary to set up the Q8-USB system:

1. Microcontroller based control station (MCU), shown in Figure 9.
2. Power cable, shown in Figure 10a.
3. Power supply rated to 5V and at least 3A, shown in Figure 10b.
4. USB 2.0 type B cable.

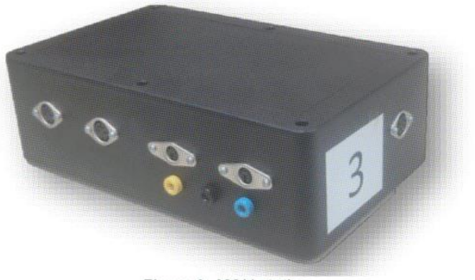
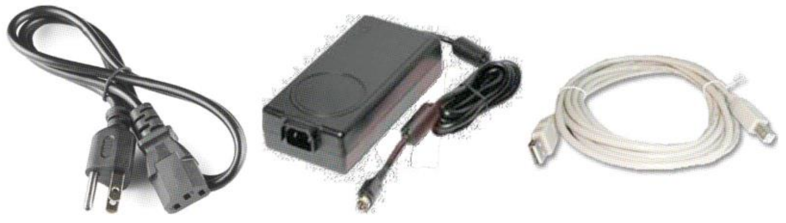


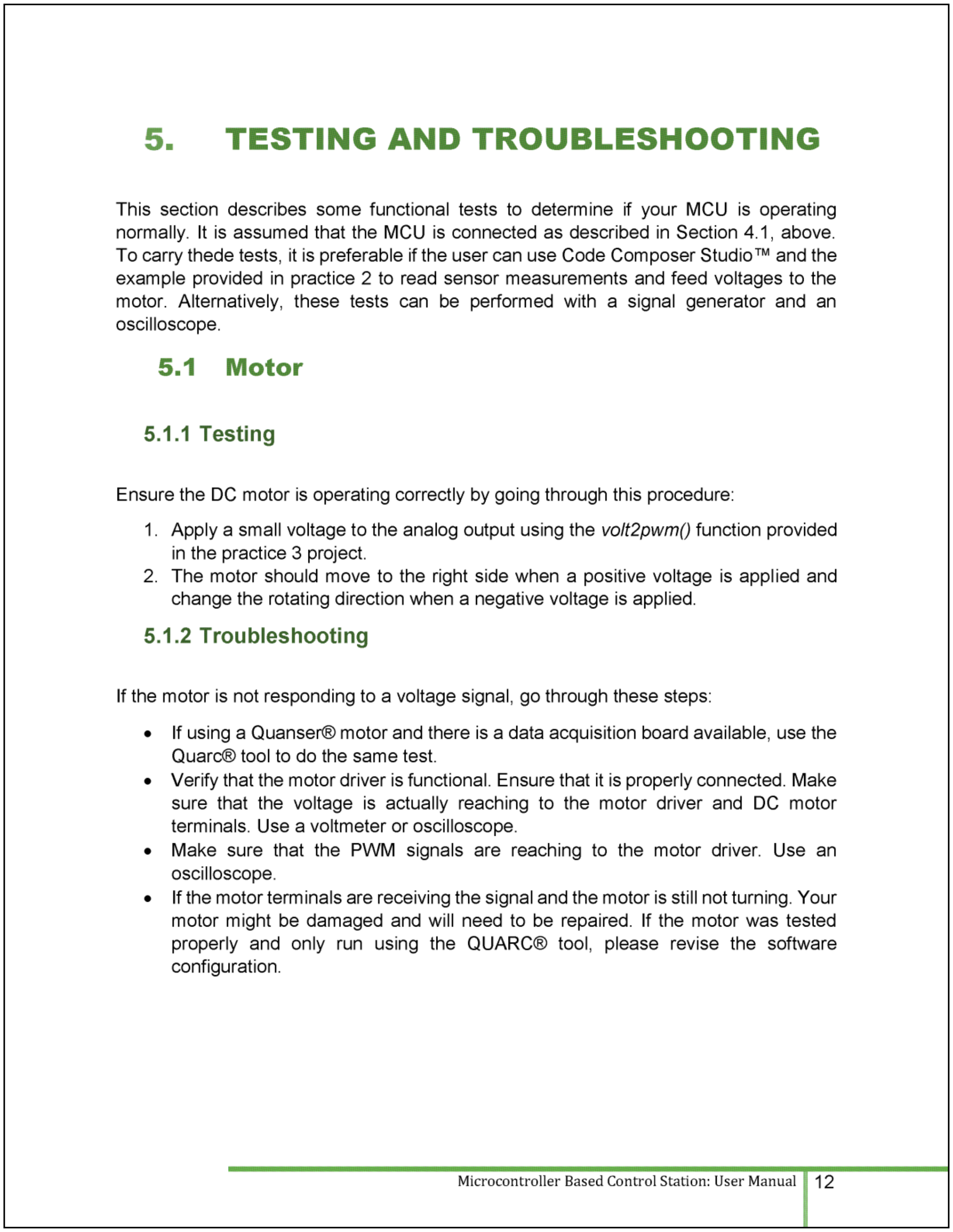
Figure 9. MCU station



a. Power Cable *b. Power Supply* *c. USB 2.0 type B cable*

Figure 10. Power Supply and Cables

User Manual Page 10



5. TESTING AND TROUBLESHOOTING

This section describes some functional tests to determine if your MCU is operating normally. It is assumed that the MCU is connected as described in Section 4.1, above. To carry thede tests, it is preferable if the user can use Code Composer Studio™ and the example provided in practice 2 to read sensor measurements and feed voltages to the motor. Alternatively, these tests can be performed with a signal generator and an oscilloscope.

5.1 Motor

5.1.1 Testing

Ensure the DC motor is operating correctly by going through this procedure:

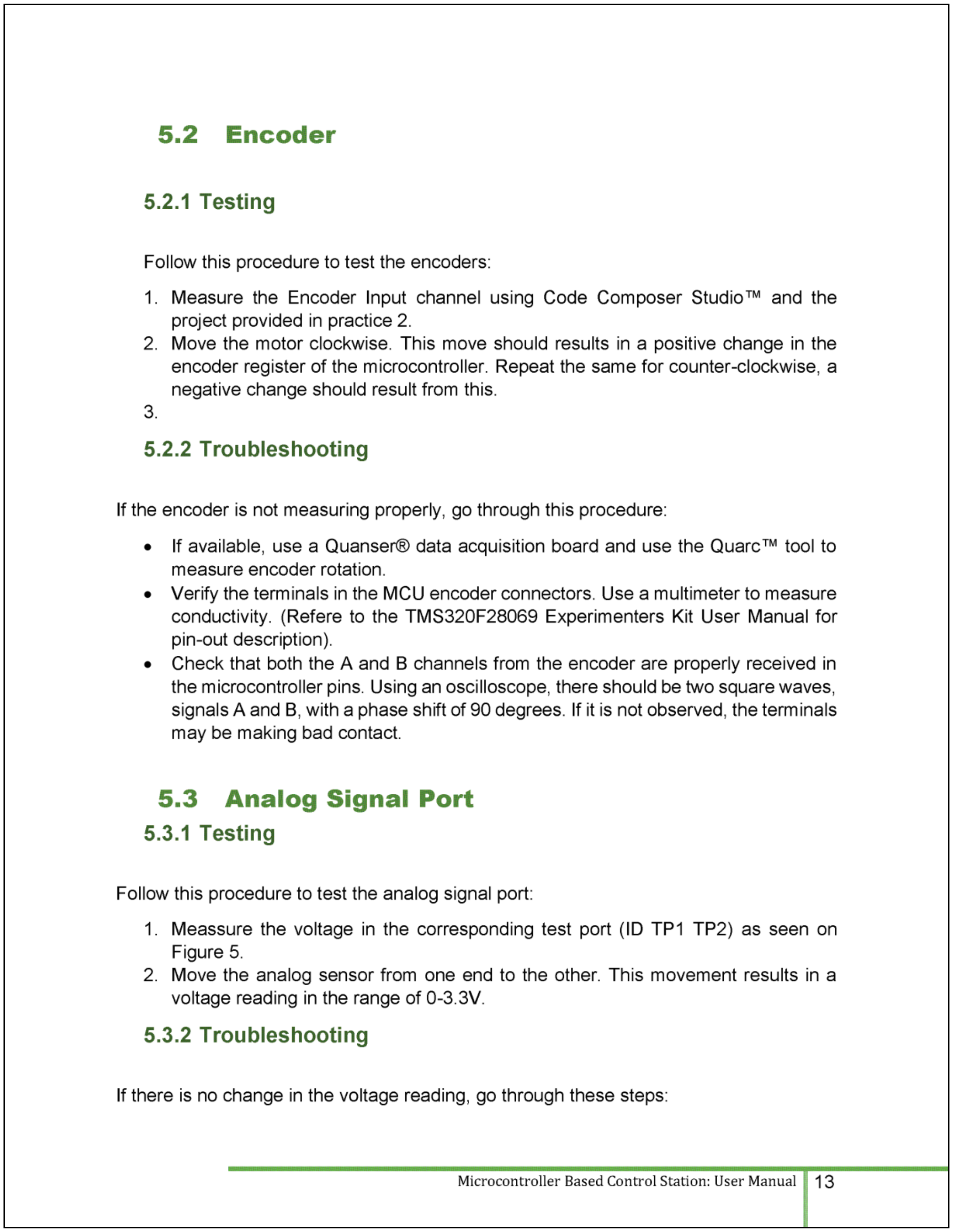
1. Apply a small voltage to the analog output using the *volt2pwm()* function provided in the practice 3 project.
2. The motor should move to the right side when a positive voltage is applied and change the rotating direction when a negative voltage is applied.

5.1.2 Troubleshooting

If the motor is not responding to a voltage signal, go through these steps:

- If using a Quanser® motor and there is a data acquisition board available, use the Quarc® tool to do the same test.
- Verify that the motor driver is functional. Ensure that it is properly connected. Make sure that the voltage is actually reaching to the motor driver and DC motor terminals. Use a voltmeter or oscilloscope.
- Make sure that the PWM signals are reaching to the motor driver. Use an oscilloscope.
- If the motor terminals are receiving the signal and the motor is still not turning. Your motor might be damaged and will need to be repaired. If the motor was tested properly and only run using the QUARC® tool, please revise the software configuration.

User Manual Page 11



5.2 Encoder

5.2.1 Testing

Follow this procedure to test the encoders:

1. Measure the Encoder Input channel using Code Composer Studio™ and the project provided in practice 2.
2. Move the motor clockwise. This move should results in a positive change in the encoder register of the microcontroller. Repeat the same for counter-clockwise, a negative change should result from this.
3.

5.2.2 Troubleshooting

If the encoder is not measuring properly, go through this procedure:

- If available, use a Quanser® data acquisition board and use the Quarc™ tool to measure encoder rotation.
- Verify the terminals in the MCU encoder connectors. Use a multimeter to measure conductivity. (Refere to the TMS320F28069 Experimenters Kit User Manual for pin-out description).
- Check that both the A and B channels from the encoder are properly received in the microcontroller pins. Using an oscilloscope, there should be two square waves, signals A and B, with a phase shift of 90 degrees. If it is not observed, the terminals may be making bad contact.

5.3 Analog Signal Port

5.3.1 Testing

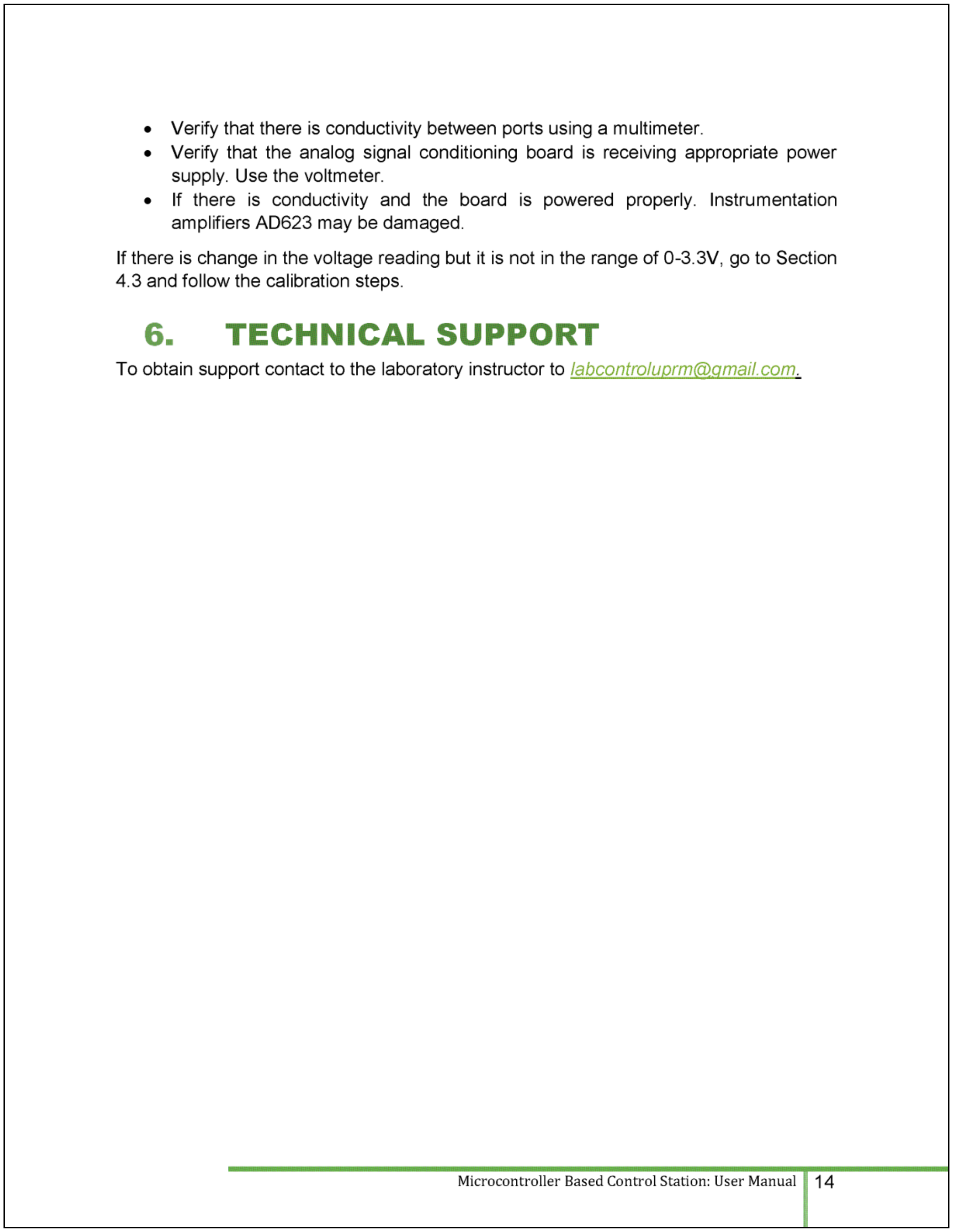
Follow this procedure to test the analog signal port:

1. Meassure the voltage in the corresponding test port (ID TP1 TP2) as seen on Figure 5.
2. Move the analog sensor from one end to the other. This movement results in a voltage reading in the range of 0-3.3V.

5.3.2 Troubleshooting

If there is no change in the voltage reading, go through these steps:

User Manual Page 12



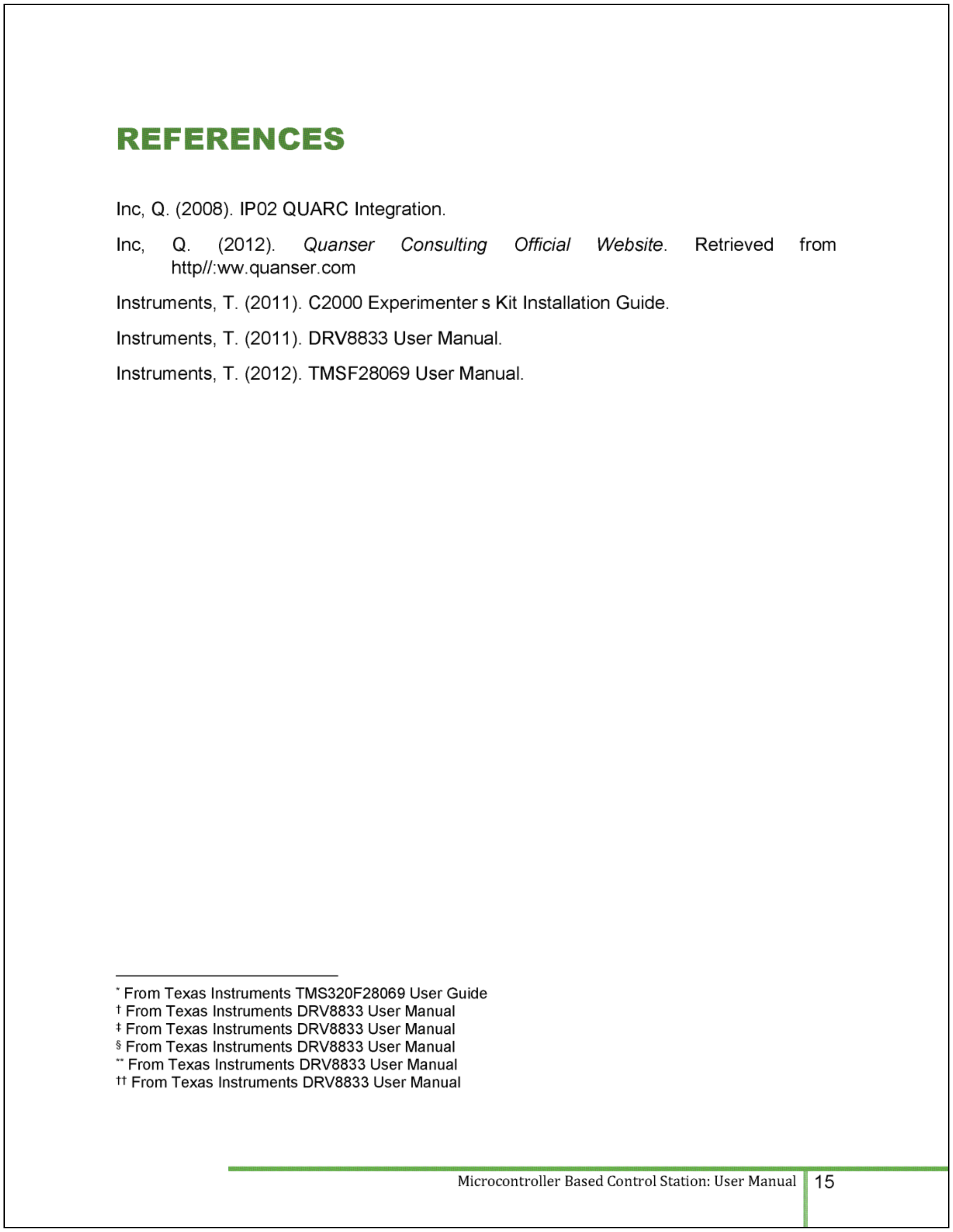
- Verify that there is conductivity between ports using a multimeter.
- Verify that the analog signal conditioning board is receiving appropriate power supply. Use the voltmeter.
- If there is conductivity and the board is powered properly. Instrumentation amplifiers AD623 may be damaged.

If there is change in the voltage reading but it is not in the range of 0-3.3V, go to Section 4.3 and follow the calibration steps.

6. TECHNICAL SUPPORT

To obtain support contact to the laboratory instructor to *labcontroluprm@gmail.com*.

User Manual Page 13



REFERENCES

Inc, Q. (2008). IP02 QUARC Integration.

Inc, Q. (2012). *Quanser Consulting Official Website*. Retrieved from http//:ww.quanser.com

Instruments, T. (2011). C2000 Experimenter s Kit Installation Guide.

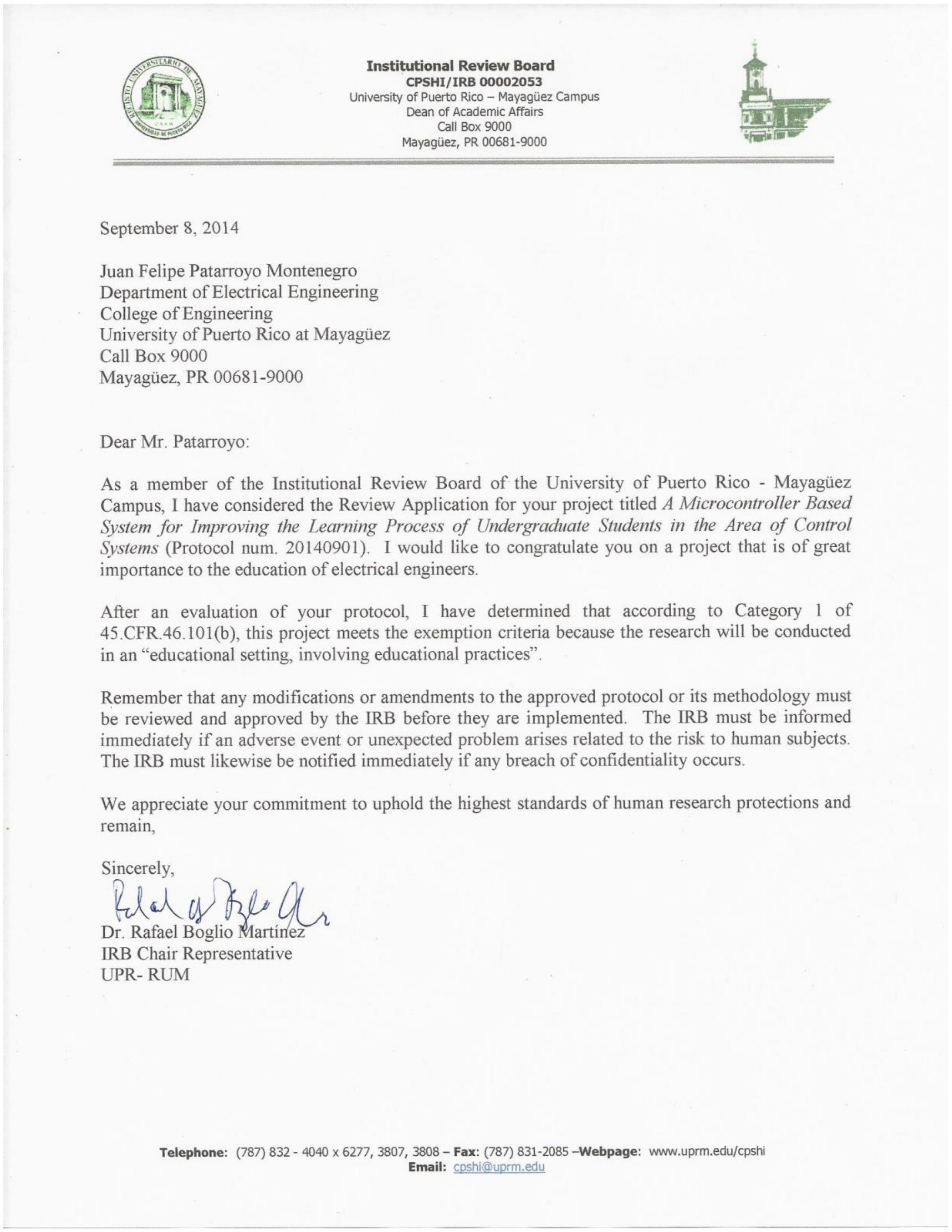
Instruments, T. (2011). DRV8833 User Manual.

Instruments, T. (2012). TMSF28069 User Manual.

[*] From Texas Instruments TMS320F28069 User Guide
[†] From Texas Instruments DRV8833 User Manual
[‡] From Texas Instruments DRV8833 User Manual
[§] From Texas Instruments DRV8833 User Manual
[**] From Texas Instruments DRV8833 User Manual
[††] From Texas Instruments DRV8833 User Manual

User Manual Page 14

Appendix F: IRB Acceptance Letters



Institutional Review Board
CPSHI/IRB 00002053
University of Puerto Rico – Mayagüez Campus
Dean of Academic Affairs
Call Box 9000
Mayagüez, PR 00681-9000

September 8, 2014

Juan Felipe Patarroyo Montenegro
Department of Electrical Engineering
College of Engineering
University of Puerto Rico at Mayagüez
Call Box 9000
Mayagüez, PR 00681-9000

Dear Mr. Patarroyo:

As a member of the Institutional Review Board of the University of Puerto Rico - Mayagüez Campus, I have considered the Review Application for your project titled *A Microcontroller Based System for Improving the Learning Process of Undergraduate Students in the Area of Control Systems* (Protocol num. 20140901). I would like to congratulate you on a project that is of great importance to the education of electrical engineers.

After an evaluation of your protocol, I have determined that according to Category 1 of 45.CFR.46.101(b), this project meets the exemption criteria because the research will be conducted in an "educational setting, involving educational practices".

Remember that any modifications or amendments to the approved protocol or its methodology must be reviewed and approved by the IRB before they are implemented. The IRB must be informed immediately if an adverse event or unexpected problem arises related to the risk to human subjects. The IRB must likewise be notified immediately if any breach of confidentiality occurs.

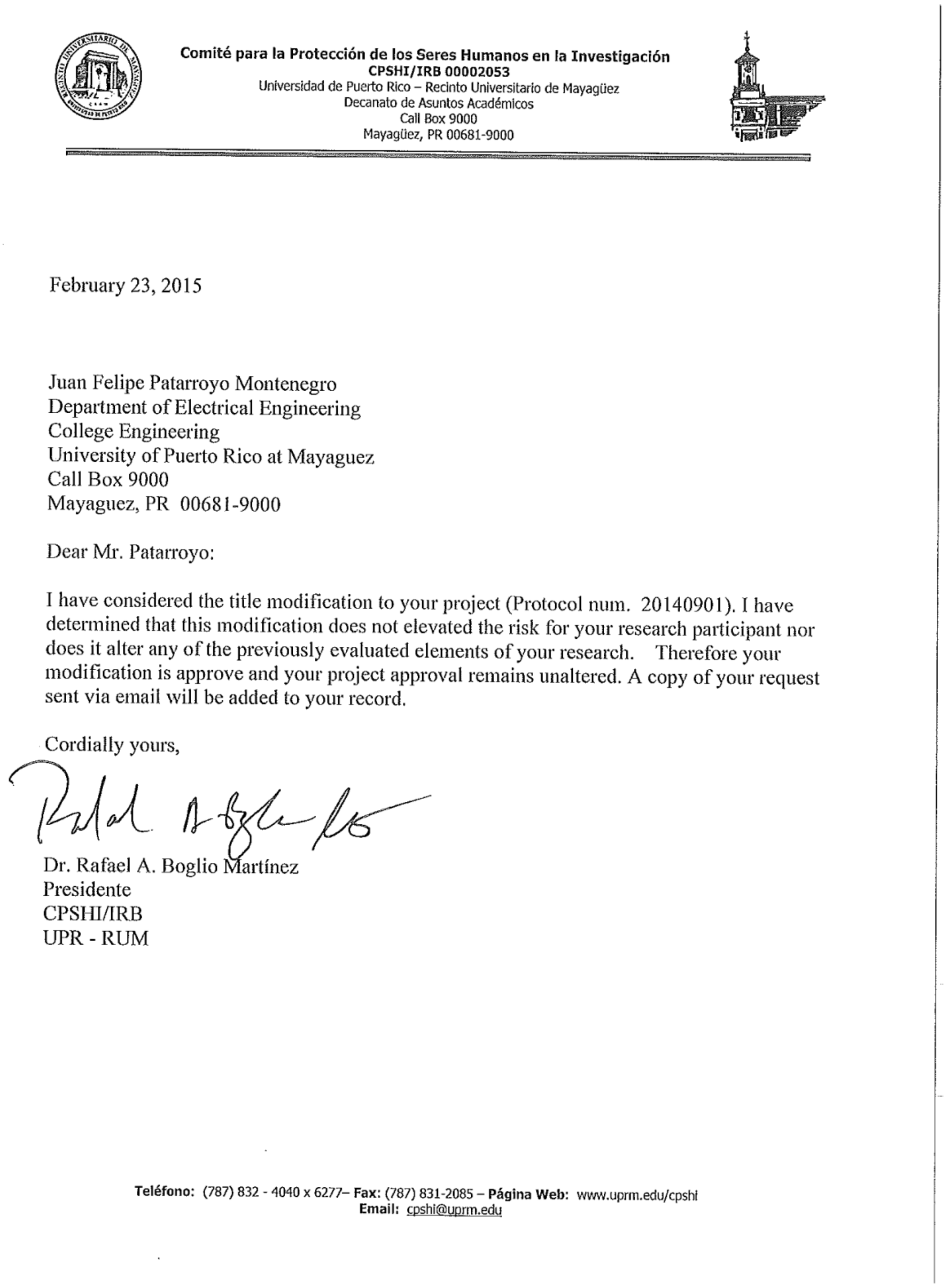
We appreciate your commitment to uphold the highest standards of human research protections and remain,

Sincerely,

[illegible]

Dr. Rafael Boglio Martínez
IRB Chair Representative
UPR- RUM

Telephone: (787) 832 - 4040 x 6277, 3807, 3808 – **Fax:** (787) 831-2085 –**Webpage:** www.uprm.edu/cpshi
Email: cpshi@uprm.edu



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

February 23, 2015

Juan Felipe Patarroyo Montenegro
Department of Electrical Engineering
College Engineering
University of Puerto Rico at Mayaguez
Call Box 9000
Mayaguez, PR 00681-9000

Dear Mr. Patarroyo:

I have considered the title modification to your project (Protocol num. 20140901). I have determined that this modification does not elevated the risk for your research participant nor does it alter any of the previously evaluated elements of your research. Therefore your modification is approve and your project approval remains unaltered. A copy of your request sent via email will be added to your record.

Cordially yours,

[illegible]

Dr. Rafael A. Boglio Martínez
Presidente
CPSHI/IRB
UPR - RUM

Teléfono: (787) 832 - 4040 x 6277– **Fax:** (787) 831-2085 – **Página Web:** www.uprm.edu/cpshi
Email: cpshi@uprm.edu