AC 2012-4006: SO MANY EDUCATIONAL MICROCONTROLLER PLATFORMS, SO LITTLE TIME!

Dr. Antonio Francisco Mondragon, Rochester Institute of Technology

Antonio F. Mondragon-Torres received a B.Sc. degree with honors from Universidad Iberoamericana, Mexico City, Mexico, a M.Sc. degree from Universidad Nacional Autonoma de Mexico, Mexico City, Mexico, and a Ph.D. degree (as a Fullbright-CONACYT scholarship recipient) from Texas A&M University, College Station; all degrees in electrical engineering in 1990, 1996, and 2002, respectively. From 1988 to 1995, he worked in a telecommunications company TVSCOM, Mexico City, Mexico, designing teletext products, first as a Design Engineer and later as a Design Manager. In 1995, he joined the Mechanical and Electrical Department, Universidad Iberoamericana, as an Associate Professor. From 2002 through 2008, he was with the DSPS R&D Center's Mobile Wireless Communications Technology branch, Texas Instruments Dallas, Texas, and in 2008, he moved to the nanoMeter Analog Integration Wireless branch where he worked as Analog IP verification technical lead. In 2009, he worked for Intel Guadalajara, Design Center in Mexico as Front-End/Back-End technical lead. In 2009, he joined the Electrical, Computer and Telecommunications Engineering Technology Department at the Rochester Institute of technology where he currently is a tenure-track Assistant Professor. His research interests are analog and digital integrated circuit implementation of communications systems, and system-on-a-chip methodologies.

Dr. Adriana Becker-Gomez, Rochester Institute of Technology

So Many Educational Microcontroller Platforms, so Little Time!

1 Introduction

Since embedded systems are ubiquitous, we should expose engineering technology students at an early stage in their program with tools that will lead them to ideation, innovation, energy awareness, and problem solving skills, which will enable them to become part of a very competitive workforce in the future.

To teach a basic microcontroller course at early stages of the electrical and computer engineering technology program is very challenging. Many students come with experience already on one or more particular microcontroller development platforms that they have used in high school or that they have used as hobbyists. On the other hand, the majority of students have not been exposed to any microcontroller platform at all. The worst part is that they want to learn how to develop smartphone or tablet like applications right away, and turning on LEDs does not impress them anymore.

If we survey the number of different microcontroller platforms available to teach students, we find a very large number of alternatives: CISC and RISC architectures; platforms that could be programmed very efficiently in assembly language and others using very high level languages; platforms that do not have any open source libraries to perform input/output interfaces and other ones that have a complete set of libraries, and platforms that hide all the microcontroller architecture and just focus on applications.

What is a platform that could offer good exposure to microcontroller architectures and still satisfy these criteria: ease to program in assembly language and high level languages; ease to perform graphical system programming and configuration; and the possibility for students to apply it in project based learning. In addition there are a couple of other very important factors to consider. Is the platform affordable for students to buy and experiment with? Would they be able to use it later in their degree program?

In this paper we will give an overview of some of the educational microcontroller platforms available, and we will give an example of one platform that has been selected to teach an introductory microcontroller based course that has all of the above traits and has been used for the last couple years in an electrical and computer engineering technology department.

2 Course Requirements

The selected course covered in this paper is that of a second year introductory overview of microcontroller concepts and the required learning outcomes are as follows:

1) Explain the differences between microcomputer and microcontroller.
2) Explain the hardware and software features of a microcontroller.
3) Design microcontroller assembly programs.
4) Using a simulator, test and debug assembly language programs.
5) Analyze and design assembly language software delay routines.
6) Write software subroutines; explain their execution and their effects on the system stack and processor registers.
7) Write interrupt service routines; explain their execution and their effects on the system stack and processor registers.
8) Analyze and design decoding logic for memory and I/O subsystems.

9) Write assembly language programs to initialize and utilize the I/O features of a single board computer.

As can be observed in these learning outcomes, assembly language programming is a fundamental part of the course. Some educators and practicing engineers can argue that nobody programs in assembly anymore, which is true in most of cases. To use a microcontroller today does not even require a degree in electrical or computer engineering technology. Platforms such as Arduino[1] and mbed[2] have evolved to a level, where no formal training is required to design applications using these platforms, and a large number of disciplines from art to engineering are using these development systems for rapid prototyping and even for production.

Nevertheless, job descriptions were found recently in sites such as IEEE jobs[3], Monster[4] and Yahoo! Careers[5] having Arduino or mbed as a skill required. Because of this, we need to train our students on a platform that will be easy to use, that is widely used in industry and that will allow students to work at all levels of abstraction, from assembly language, going through a high level language such as C or C++ all the way to graphical programming and automatic code generation.

2.1 *Challenges*

As mentioned previously, the main challenge is to keep students motivated and engaged. Students want to create complex applications immediately and our job as educators is to keep them with their feet on the ground by providing guidance through all the steps required to accomplish the type of applications that they envision and are motivated to do. Students need to learn to appreciate the level of effort required to blink a simple LED as a precursor to use gestures on a touchscreen such as those available in smartphones and tablets.

2.2 *Previous Experience*

The majority of students that are taking this course have not been exposed to any type of microprocessor or microcontroller before; there are just a few that have been introduced in high school through robotics clubs or that are coming as transfer students with associate degrees. By their second year, students have a quarter long basic digital fundamentals course and the vast majority a single programming course in C++.

2.3 *Perception*

This course is perceived as difficult due to the amount of new material to be covered; the learning of assembly language is always intimidating to the newbies. The introduction of these concepts are particularly difficult: memory mapping; peripheral interfacing; using timers and counters; serial and parallel communications; mnemonics, opcodes and operands; addressing modes; polling and interrupts; using stacks and queues; and using pointers.

While it is true that it can be overwhelming, the concepts learned in this course are some of the most important that both electrical and computer engineering technology students will use. It is important to convey to students that no matter what their specialization, one way or another they will be using a programmable device as the interface with their designs. A well rounded graduate should be able to easily add a microcontroller to their projects, since it is now cheaper and more time efficient to use off the shelf solutions such as a microcontroller rather than designing discrete logic, state machines and programmable logic.

2.4 *Engagement*

This is one of the most difficult tasks, to keep students engaged in a course that will be the backbone of their careers, full of new concepts that could become overwhelming and that share very limited scope with what they use in their normal lives.

The current generation of students likes to receive immediate feedback, and in the case of a microcontroller course, they appreciate something that blinks or moves in response to their efforts. A robotic platform can be very engaging, but limited in the applications they can develop early in their programs[6]. The solution that has become the most popular in the use of microcontroller platforms is to use open source concepts where both the hardware and the software are available for anyone to use without restrictions and are based on individual contributions. This is a worldwide movement that has generated a lot of momentum, and it is here to stay. The best example of this is the Arduino platform that is being used by students from all majors from artists to engineers. Students have available to them a hardware platform and a set of libraries, and they either can buy shells to extend their hardware or can build their own.

Another very interesting platform is the mbed, which is based on the same principle of open source, but this platform is more geared to prototyping on breadboards due to the availability of a dual in line package or DIP. This makes it very easy to mount and interface with a large number of devices without having to integrate with other printed circuit boards.

These two platforms have proven to be extremely engaging by allowing students to interface to the internet, to displays, and to control motors, to communicate wirelessly, and to create really complex software interfaces with relatively simple commands. The best of all is that both are community driven.

3 Microcontroller Platforms

When talking about microcontroller platforms, there is a huge variety of options available. It is really difficult to find the ideal microcontroller that has everything in a single package and within reasonable cost. Several years ago the market was driven by 8-bit microcontrollers, then 16 bits started to become popular, but suddenly the 32-bit microcontrollers are now offered at almost the same price as their 8-bit counterparts, with more code density, with equivalent power consumption and with a growth path within the same family.

It is a difficult choice to find a single microcontroller that has the best of all worlds. What would be some of the traits that the authors would like to have on a microcontroller platform? The first will be to have a rich set of general purpose registers that students can use without having resource bottlenecks in their programs. The next would be to have an orthogonal instruction set where all the resources could be used with all the addressing modes. It should also have a rich set of addressing modes that match the most common data structures in high level programming languages such as pointers, easy access, and creation and maintenance of structures such as multidimensional arrays, queues and stacks. Then it would be that instructions will always use the same amount of memory and execute in one clock cycle, so counting instructions will give us the amount of time to execute a particular block of code.

In computer architecture there are several classifications in terms of how to access instructions and data, and the complexity of the instruction sets. In the next subsection, we will talk about the main classifications according to the instructions sets and give some example architectures.

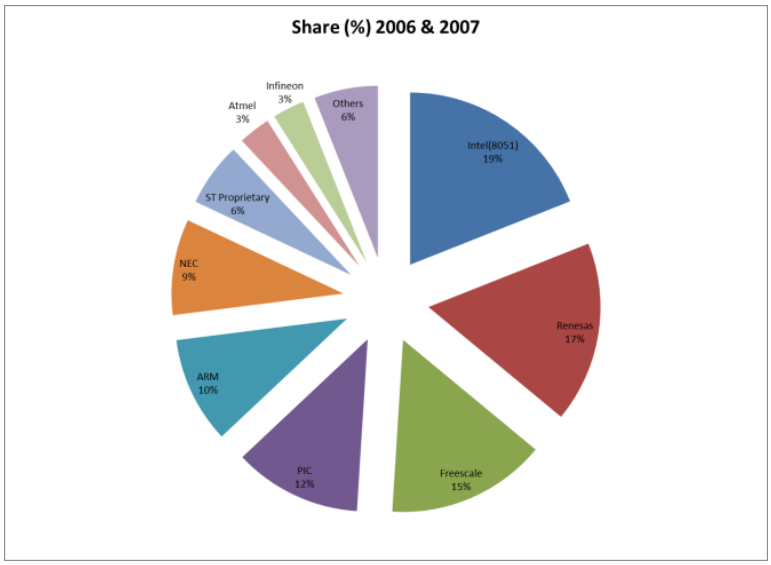

Figure 1. Microcontroller architectures market share for 2006 and 2007.

3.1 *CISC*

Complex Instruction Set Computer or CISC is characterized by having a large set of very specific instructions for different purposes, and the instructions are targeted to solve precise problems in hardware. Usually the instruction set is very large, and each instruction is usually variable in size depending on the operands required. A typical example of CISC architecture is the Intel 8051, which has a Harvard architecture (instruction and data in separate memory spaces) and has been around since 1980. It has been one of the most popular microcontrollers as can be observed in Figure 1 and is still being used today in many products. The question is, "Are new products still being designed with the 8051 core?"

Another very popular family is the Motorola 68HC11/12 (now produced by Freescale Semiconductor) which has been also been very popular for teaching [7, 8] and for product design. The 68HC11 is an inexpensive and easy to use 8-bit microcontroller that has been around since 1985. The Motorola 68HC12 is a 16-bit microprocessor descended from the 68HC11. It is able to access larger memory banks and additional peripherals yet remains code compatible with HC11's architecture and instruction set. It has become popular with users needing more speed and memory than the HC11 provides. The 68HC11/12 has gained momentum because of its simple architecture and instruction set. A large selection of books, sample projects and free software is available for this target. There are readily available kits that bring out all the resources of the HC11/12 as well as support hardware and prototype area.

What the authors consider the best CISC architecture as an educational platform for its time was the Motorola 68000 family. It was one of the first 32-bit general purpose architectures that created a paradigm in consistent design practices, and its assembly language was quite intuitive. It was a unique architecture where important concepts such as general purpose registers, addressing modes, memory addressing and some other important characteristics were able to be demonstrated.

The Intel 80x86 architecture has been extremely popular, but not on the embedded market. It is difficult to program in assembly language, and it is not very structured in terms of register usage. The modern contender for embedded systems is the Atom processor, which is very popular in

notebooks, netbooks and embedded appliances. Many of the CISC architectures are accumulator based, and instructions vary in size from one to as many as 5 or 6 words.

3.2 *RISC*

The Reduced Instruction Set Computer or RISC has the characteristic that the number of instructions is small, and more complex instructions such as the ones available in CISC are executed as a series of simpler instructions. It is based on load and store operations where there are only two instructions to access memory: load an operand from memory into a general purpose register and store an operand from a general purpose register into memory. Since the operands are held in registers, traditionally a large set of registers is provided. Another characteristic of RISC is that all instructions are of the same size, usually one word long, such that instruction execution is predictable and easily pipelined. Some registers in RISC processors are used with specific purposes, such as constant generators, and the concept of pseudo-instructions is widely used. One characteristic of RISC processors is that they traditionally are programmed using a high level language such as C. Part of the instruction execution efficiency is assigned to the compiler in order to match the architecture; this makes the RISC processor simpler.

A good example of RISC architecture is the family of PIC microcontrollers, which also has a Harvard architecture similar to the 8051 CISC architecture. It has been extremely popular with hobbyists, in academia and for product design. Contrary to RISC philosophy, it is an accumulator based architecture.

Another very popular family is the Atmel AVR architecture; it executes most of the instructions in single execution cycle, has a large set of 32 8-bit general purpose registers and also has a Harvard architecture.

Texas Instruments has a family of microcontrollers denominated MSP430 that is RISC but has some unique characteristics that fit into both CISC and RISC definitions. First it does not have a Harvard architecture but instead has a Von Newman architecture, since both instructions and data reside in the same memory space. The instruction set is only 27 instructions and 24 pseudo-instructions. It has a large general purpose register set of 16 16-bit registers. It does not have load and store architecture, meaning that the move operations can be combined with registers and memory. Instruction length varies from one up to three words. It has most of the useful addressing modes; this makes it very simple to translate, for example, a C program into its assembly equivalent, and it is very useful to demonstrate side by side C code and disassembled assembly language instructions.

Recently the ARM architecture has been dominating the market, especially in consumer electronics with smartphone and tablet usage popularity. ARM licenses its cores to a huge number of semiconductor companies, and they are producing microcontrollers based on this architecture. One of the decision criteria for microcontroller platform selection is that it is worthwhile to learn a single architecture that will most likely dominate the embedded market in years to come. One of the most popular ARM cores is the Cortex-M series that ranges from a very simple 8-bit microcontroller replacement with the Cortex-M0, to the standard Cortex-M3, and to the digital signal processing like floating point Cortex-M4. The ARM architecture is more like the traditional load and store architecture descriptions, it has really high performance in terms of Dhrystone million of instructions per second per Megahertz (DMIPS/MHz), it is very C programming friendly, and it has several energy saving modes. All these features have made this

architecture very popular. All Android phones run on ARM platforms, and also the iPad and iPhone have their own version of an ARM core. The upcoming Windows 8 will also run on ARM platforms.

Another contender in the RISC arena is the MIPS architecture, and recently Microchip has launched a family of 32-bit microcontrollers based on this architecture, which is especially popular with research communities.

3.3 *Platform Summary*

As we can observe in the CISC vs. RISC decisions, there is not a single platform that can achieve all the traits on the instructor's criteria All mentioned architectures have their strengths, weakness, opportunities and threats. Choosing a particular architecture will always leave students without the feel for some of the concepts available in another. Thus it is very important to expose students to all design tradeoffs, so they become aware of their choices when selecting a platform and then take a deep dive into a particular architecture. Even if using a single architecture for teaching, we always have to make an effort to contrast how the same processes or operations would be performed on other architecture.

A very important aspect now is energy conservation, and the battle of semiconductor companies is in terms of who can deliver the best performance using the least energy. This is also another dimension that goes beyond the traditional criteria to select a platform when planning a microcontroller course[9].

Last but not least, what are the platforms most used in industry that will give students a competitive advantage when looking for co-ops or permanent jobs? As observed in Figure 1 there are several players that have a large market share. Surprisingly they are the 8051, 68HC11 and PIC architectures, which we can consider legacy platforms, but the installed base of products based on them is still very large. Some of the new product designs now employ 32-bit microcontrollers, and if we plot the equivalent data in some years, we may see that the players have changed. This is because the new architectures bring desirable features: for example, better performance, better code density, energy efficiency and cost. The best that we can do as educators is to follow the trends and choose the best platform for students to get the maximum exposure to important concepts in the most pedagogical manner.

4 Programming Language

This is an interesting topic where people agree and disagree. The vast majority of modern embedded design is being done using a high level language, in particular the embedded subset of the C language. The C++ language has made incursions into the embedded arena, and it is used in higher levels of abstraction where the Java language is also very popular. Traditionally the underlying control of the hardware architecture is being coded in the plain C language and also in assembly language.

People can argue that assembly language is the most efficient way to program a microcontroller, and indeed it is for CISC processors. RISC philosophy assumes that the compiler will also be in charge of the processor performance by selecting multiple simple instructions to be executed and also by having tight control of the pipelines. It is simpler for a compiler to optimize the instruction sequencing such that it can schedule out of order instructions and predict branches, than for a programmer to find the best set of instructions to code. These are some of the few

reasons RISC processors can be more efficient to program in a high level language, in addition to the fact that RISC cores are designed with compilers in mind.

There is a simple rule of thumb that engineers know in different formats and ratios, "90% of the time is spent executing 10% of the code." This is the best reason why our graduates should learn assembly language. What we tell students is that anybody can do simple embedded coding, but only they would be qualified to push a processor to the limits in terms of coding efficiency, cost, performance and power consumption. This is what makes the difference when hiring an engineer to develop a product in the current market situation where there are time-to-market constraints as well as the need to design a very cost efficient product loaded with innovative features to have market differentiation from similar products.

Based on these facts, we are convinced that the solution is to expose students to both languages and to give them the criteria and challenges for when to use each one to make their code more efficient and increase performance.

5 Development Tools

Semiconductor companies make their revenue based on how many chips they sell. This is the reason why recently most of the companies have code size or performance limited versions of their development tools for free in order to motivate designers to use their products. In practical student assignments and laboratory exercises, the size of problems given fit within the restrictions of the free versions available. Students are encouraged to download these versions on their computers, so they can work anywhere rather than to be tied to the laboratory and its available hours, which was the case some years ago when the platforms were rather expensive for students to buy.

Not only are the development tools commonly known as electronic design automation (EDA) tools free, but semiconductor manufacturers also make available a large volume of sample code that can be modified to meet particular requirements. Open source communities have been great in motivating students and hobbyists to share their knowledge so other people can leverage their work without any cost and just for the sake of expanding the resources available. As we mentioned previously, the Arduino community is the best example of this movement.

6 Platform Selection

Now as the title specifies there are "So Many Educational Microcontroller Platforms, so Little Time!" The question is how to choose the most appropriate platform to teach students early in the program those concepts that will be used throughout the curricula as well as in their professional life. In the next subsections we will give a very brief introduction to some of the platforms that have been evaluated or that have been popular with students. It is by no means a comprehensive list, and it may be biased towards certain companies, but a decision had to be made, and it all depends on the support and relations that a particular company provides. In Table 1 we present several platforms that we have been able to evaluate (some more than others). In our opinion this list of platforms has a very high potential to be incorporated to teach microcontroller courses. Some are very basic and intuitive some others are more complex and powerful.

6.1 *Arduino*

As mentioned previously the Arduino platform is the most popular embedded platform where it is easy to use, to interface and to program. Since there is a very large community of users, it is

being used by different practitioners from artists to engineers. The platform is very strong, based on its simple programming language and code structure; it also has a extremely large set of libraries where it is likely that many of the problems faced by students have already been solved by somebody in the on-line communities. There are two main processor architectures used in the Arduino platform: the Atmel AVR and the Microchip PIC32. It was recently announced that an ARM version will be available soon[10].

Their motto is: *"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments."*

Table 1. Examples of microcontroller platforms that have been evaluated†, used by students‡ and used in the program*.

Development platform	Processor Core	Cost
‡Arduino Uno - R3	Atmel 8-bit AVR processor (ATmega328)	$29.95
‡Arduino Mega 2560 R3	Atmel 8-bit AVR processor (ATmega2560)	$59.95
†chipKIT Uno32 - Digilent	Microchip PIC32 32-bit MIPS processor (PIC32MX320F128)	$26.95
†chipKIT Max32 - Digilent	Microchip PIC32 32-bit MIPS processor (PIC32MX795F512)	$49.50
†mbed NXP LPC1768	NXP Cortex-M3 32-bit ARM processor (LPC1768)	$59.95
†mbed NXP LPC11U24	NXP Cortex-M3 32-bit ARM processor (LPC11U24)	$44.95
*LaunchPad - Texas Instruments	TI MSP430G2xx 16 bit MSP430 processor (MSP430G2211/31)	$4.30
*MSP430FG4618/F2013 Experimenter Board – Texas Instruments	MSP430FG4618/F2013	$99.00
†EK-EVALBOT – Texas Instruments	LM3S9B92 ARM® Cortex™-M3	$149.00
†LPCXpresso - NXP	LPC1XXX ARM® Cortex™-M3/M0	$29.95
†STM32VLDISCOVERY – ST Microelectronics	STM32F100RBT6B LPC1XXX ARM® Cortex™-M3	$9.88
†Explorer 16 Starter Kit (DV164037) - Microchip	PIC24FJ128GA010 dsPIC33FJ256GP710	$129.99
†Cerebot MX7cK - Digilent	PIC32MX795F512L 32-bit MIPS processor	$99.00
†TWR-K60N512-KIT - Freescale	MK60N512VMD100 ARM® Cortex™-M4	$139.00
†CY8CKIT-014 PSoC® 5 FirstTouch™ Starter Kit – Cypress Semiconductor	CY8C5588AXI PSoC 5 Device - ARM® Cortex™-M3	$48.99
†Low Cost Demo Board for the Freescale MC9S12DT256	HCS12DT256 STUDENT LEARN	$95.00
*Freescale 68HCS12 Microcontroller Project Board	PROJ BOARD, HCS12DT256 S	$175.00

Now the question: *"If the Arduino platform is so widely used, there is no point on continuing evaluating other platforms?"* Well, we think we are not at that point yet! The problem with the Arduino is that anybody can use it, so there is no point in having a course on a platform that anybody can use on their own. The other point is that the actual microcontroller architecture, memory and peripherals have been hidden such that it can be ported to different processor cores without the need for the user to be aware of the details. Since the platform is designed so that the user does not care about the underlying architecture, memory structure and peripherals available, it cannot be used in a course on microcontrollers where the students are to learn the low level details of a particular architecture.

In addition, there are basically no engineering jobs that would require mastering the Arduino platform since it is so generic that no company will highlight as important the skills acquired through an informal very high level platform. If the course or project just requires simple interfacing and basic capabilities by reusing ideas and code, there is no question that Arduino can deliver results easily and can leverage a large community both local and on-line that could help with projects.

6.2 ***mbed***

Very similar to the Arduino spirit, the mbed platform achieves comparable objectives, but it is somehow more targeted to prototyping on a breadboard. What makes it unique is that uses an ARM Cortex-M3 architecture from NXP, which is quite powerful when compared with other platforms, at a very reasonable price and with a rich set of libraries also supported by a large online community worldwide. There has been recently announced the release of a Cortex-M0 version of the mbed[11]. Another very distinctive characteristic of the mbed platform is that all development tools are online, so there is no need to download software to develop with mbed. Once the software is compiled, the mbed is attached as a thumb drive, the code is just copied to the mbed, the reset button is pressed, and voila! You have a standalone running application. Well, there is a catch! There are no debugging tools for mbed and no way to evaluate the code generated by the compiler. The old "burn and crash" methodology is used to test functionality. This is a similar problem with the Arduino platform where students cannot get access to the automatically generated code and very few optimizations could be done by the final user.

6.3 ***Launchpad***

The Launchpad is based on the Texas Instruments MSP430 16-bit architecture. One of the characteristics of this family is the ultra-low power capability, which is a very important concept in modern electronic design. The MSP430 has a very nice architecture with several of the traits mentioned at the beginning of section 3. It has the following characteristics: a small number of instructions; an adequate number of general purpose registers; powerful addressing modes; orthogonal instruction set; and very clean C to assembly language translation in terms of high level structures and statements.

One of the main advantages of this platform is that it can easily be programmed using assembly language or C, and some of the hard to understand details of peripheral configuration can be accomplished using a graphical user interface. The MSP430 architecture has been used in applications such as medical electronics, industrial automation and sensing applications, networking, power metering, and energy harvesting to mention some. This particular platform is very cost effective since the price is less than $5.00 and the development tools can be obtained for free.

The Launchpad is a very pin limited device, but for the price, performance and tools available it is an ideal tool for teaching microcontroller architecture and programming.

What the Launchpad platform needs at this moment is an equivalent momentum similar to the previously mentioned open source platforms where on-line communities contribute to the development of libraries and applications. If a job search is performed issuing MSP430 as a keyword, it is likely that there will be some job postings requiring knowledge of this particular platform.

6.4 *STM32 Value Line*

ST Microelectronics has also joined the low cost and free software development tools market. There are three discovery kits for the L, VL and F4 STM32 Value Line microcontrollers. These three platforms are under $15, and the L and VL are based on ARM Cortex-M3 and the F4 on ARM Cortex-M4. Similar to the case with the Launchpad, these boards can be programmed in either assembly language or C and could be targeted to more advanced students due to the richness of the peripherals available as well as the extensive set of communication interfaces available, which can support external components.

6.5 *PSoC*

Microcontrollers have been traditionally used to interface analog and digital domains by incorporating analog to digital and digital to analog converters (ADC/DAC). An equivalent function to a DAC can be performed by generating a pulse width modulated (PWM) signal for control: for example, the intensity of an LED, the speed of a direct current (DC) motor and the position of a servo. All the previously mentioned platforms are rich in these types of peripherals and can generate multiple PWM signals by the use of timers or by specialized hardware.

The Cypress PSoC 5 platform goes beyond the typical microcontroller by integrating an ARM Cortex-M3 core, programmable analog components and also programmable logic devices into a single package. This is an excellent architecture not just for a microcontroller course but for electronics in general. There is also a PSoC 3 version that uses an 8051 core architecture with similar capabilities to the PSoC 5.

6.6 *Evalbot*

Robots are very engaging platforms for students; Texas Instruments has a robotic platform called the Evalbot that uses the ARM Cortex-M3 architecture, and it is loaded with interesting peripherals and a very good communications infrastructure. The platform can be used as either a moving robot or a complete development platform for embedded applications. The platform seems to be targeted to more advanced students such as juniors and seniors, due to the amount of setting up and coding required making it work. While it is very attractive to use it in the early years of the program, some more research has to be done to see if it can be brought to the level where it can be easy to use at the same time as intuitive to program. The little experience students have had so far with the platform points this way, but there may be frameworks already in place to make it more intuitive.

6.7 *LPCXpresso*

NXP has available a set of small prototyping boards that are pin compatible with the mbed, which has a wide set of processors from ARM Cortex-M0 to Cortex-M3. Several of these microcontrollers can be used to create USB peripherals very easily. This platform is really

attractive for use in introductory and intermediate courses. The compatibility with mbed is great so students can rapidly prototype their ideas and later substitute the platform for the LPCXpresso to work on optimized code to interface at a much deeper level with the ARM platform.

6.8 *PIC*

Microchip PIC microcontrollers are extremely popular for hobbyists, academia and industry. There is a very large number of development platforms available for 8, 16 and 32 bits. The chipkit Arduino compatible platform can also be used as a proof of concept., because a rapid prototyping tool and then direct access to the controller is possible to explore the microcontroller architecture. By using the MIPS architecture, microchip is aligned with one of the most used 32-bit RISC architectures, especially in academia. It is also very interesting that several platforms such as the Cerebot from Digilent can be expanded by using PMODs, which are programmable modules external to the platform to connect motor drivers, wireless communications, ethernet ports, etc.

6.9 *Freescale*

Freescale has traditionally been a producer of very popular microcontroller platforms such as the 68HC11/12. These platforms are widely used in academia today, and there are great resources available. The trend that we are seeing is a convergence to the ARM architecture, and Freescale offers the Kinetis microcontroller with an ARM Cortex-M3. The platform that is presented here is the Tower System, which is great due to its expandability and the rich set of peripherals and sensors available in each pluggable board.

7 Platform Selected for Introductory Course

While we have described just the surface of multiple educational microcontroller platforms, we have not really spent too much time on their particular characteristics, such as the number of general purpose input/output (GPIO) ports, wired communication interfaces (UART, I2C, SPI, CAN, etc.), hardware control interfaces (number of PWM channels, number of analog to digital converters channels, etc.), user interfaces (pushbuttons, switches, LEDs, displays), or sensors available (temperature, accelerometers, light, proximity, etc.). We will try our best to frame the needs of the platform required for an introductory microcontroller course in the engineering technology program.

7.1 *Simplicity vs. Visibility*

All of the platforms but the Arduino and mbed can be programmed with a combination of assembly language and a high level language such as C. While it is advantageous to keep things simple, a microcontroller course should provide the opportunity for students dive deep into a particular architecture and understand the inner workings of a microcontroller with everything that it implies. While at some point students will be asked to use already made libraries and function calls to application libraries, the first exposure should be to how things work, so students get an appreciation for what knowledge is required to program embedded devices. Nothing is better than coding in assembly language to expose students to the closest level of hardware control that could be achieved. This is also appreciated by employers that consider assembly language programming a real asset in the student toolset.

To be able to emulate a program running on microcontroller hardware has a lot of benefits from the perspective of analysis and optimization. Sometimes we challenge the students to find who could deliver the code with the best performance; this is where motivated students go the extra

mile to understand code optimization, execution speed and the tradeoffs involved. The next challenge could be to find out who consumes the least power while meeting design requirements.

7.2 *Cost vs. Plug & Play vs. Performance*

The title of this subsection suggests interesting tradeoffs where traditionally there have been inverse relationships among them. The faster the platform is, the more expensive; the more intuitive, the more costly; etc. What we mean by these examples is that in the past, an educational embedded system could be really expensive depending on the level of maturity of hardware and software that the particular platform had. Today it may be that the friendliest platforms are the least expensive, and one of the reasons may be due to the open source nature of the platform. We can observe that the Arduino and mbed are Plug & Play platforms that are very cost efficient, but it would be difficult to measure the performance since the underlying hardware is not easily accessible. These platforms are restricted to applications where the real time is in the order of couple milliseconds, while for example the STM32XXDISCOVERY series could potentially be used for high performance audio processing using the Cortex-M4 with floating point operations, and all for under $20. What this platform lacks to be very popular, is a community that develops applications or libraries provided from the manufacturer to an initial community. There are some open source platforms like the Beagleboard and the Beaglebone that have laptop like performance, at the same time they can run a full-fledged operating system, and all for less than $150.

As we can observe, today students have a plethora of options from which to choose, but the question is, "For an introductory course what are the basic requirements?" We would like to have access to an educational platform that is cost effective, easy to use and has a representative good computer architecture that is easy to understand, program and interface. For an introductory course, performance may not be the target since students are struggling first to understand the basic concept rather than pushing the platform to the limits.

7.3 *Assembly vs. C vs. GUI driven*

As mentioned before, the programming language is an integral part of learning the details of microcontroller architectures. Most of the current jobs in embedded systems require C language proficiency, and every engineering technology program should do a very good job in teaching the fundamentals of C and C++. Sometimes the subset of C required by embedded programmers can be obscured by all the features offered by C++, and sometimes it is difficult to descend into the very basic features of C that make the subset required for embedded programming. It is probable that assembly language programming could be one of the deadliest weapons in the engineering technology graduate arsenal, since assembly language could become a language in danger of extinction. Today it is possible to create complete optimized applications in pure embedded C language, but when required, the graduate should be able to change hats and perform as an assembly programmer.

Sometimes one of the most complex tasks in learning how a microcontroller works may lie in how the peripherals work and how are they programmed. Let's consider the example of controlling a servo with a pulse width modulated (PWM) signal of 20-ms period with a pulse width between 1 ms to 2 ms for complete 90° control. In the case of Arduino and mbed there are already libraries that perform the required functions, so the students do not have to deal with the details of PWM generation. If we would like to control the servos with other platforms, depending on the microcontroller used there will be a large number of different methods to

accomplish the same functionality. The challenge to the student is to choose the best way to do it; the language may be irrelevant and could be done in either assembly or C. What is required in this case is a clear understanding of the computer architecture and the peripherals available.

A very interesting method of programming peripherals is by using a graphical user interface (GUI. We will give an example of how the students based on specification can program peripherals in a very easy way, but at the same time with complete control of the underlying hardware. The platform used is the MSP430 Launchpad, and the GUI is denominated Grace, which is part of the tools available from Texas Instruments.

The students are asked to control a servo, and the following specifications are given: PWM with frequency of 50 Hz (20 ms) and a duty cycle from 5% (1 ms) to 10% (2 ms). These are some possible steps that students could follow to accomplish the task as can be observed in Figure 2.

a) The student opens Grace and takes a look at the microcontroller architecture.
b) The student observes that the requirements call for time generation. There is a timer denominated Timer0_A2; he or she double clicks on the time block to open it and get the timer overview window. Here it can be observed that it has a PWM mode, and instructions for configuration appear.
c) The student now follows the instructions and decides to connect the servo on port P1.2.
d) Then the student can open the register control window to find out how all registers and bits within the registers were programmed. By changing the duty cycles in the basic window, he or she can observe the minimum and maximum ranges.
e) The resulting C program is a template that is used to write the actual code for the application. In this case if the student wants to change the PWM duty cycle, he or she can go to the TACCR1 register and change the value, which in turn will move the servo.
f) The student can now add some lines to control the servo, (to move back and forth controlled by a software delay).

Once the student understands the operations and sees the results, he or she can go into power user modes and start playing with advanced configurations to the point where he or she can directly program the component using the control register.

By using Grace, the student is able to visualize microcontroller resources, to understand that there are some timer functions involved. The student learns that there was one timer with PWM capabilities, that it could be configured for a frequency of 50 Hz by programming a value of 239 to TACCR0 and that the percentage duty cycle was programmable. The student finds the ranges corresponding to 5% to be TACCR1=12 and 10% to be TACCR1=24 for a 90° position control. At the end of the assignment, the student has to write a few lines of C code to be able to control the servo with a resolution of 7.5°. The next challenge once the student is proficient could be how to get a 1° resolution with the same platform.

7.4 *Industrial Experience Training*

Out of this very basic introductory course, the students must be motivated and prepared with the decision criteria on how to integrate a microcontroller into their projects. Since the course teaches basic skills available in all architectures, they should have a broad view of the different tradeoffs in embedded systems. The students should be able to talk with recruiters and with engineers in the field about the different options available: how to specialize in their particular architecture to exploit its features; how to interface with external components; and how to structure code to meet the systems requirements. For example, we found that today, having certain keywords in their resume, (e.g., ARM, MSP430, PIC, HC11.HC12, assembly, C, C++, etc.), could potentially help them move their resume to the top of the pile and have a better chance of being hired for co-op or permanent jobs.

8 Platform Selected for the Course

As was mentioned in the instructor's wish list, there were many qualities that a microcontroller must have in order to be representative of a good architecture. The tradeoff here is that this is an introductory course where students basically have not had any previous exposure to any types of microcontroller platforms. As can be seen in Table 1, there were a relatively large number of microcontroller platforms suitable for education. The decision lies from a simple Arduino/mbed to the more advance ARM architectures.

The microcontroller architecture selected for this particular course is the MSP430, which has been proven to have a very simple and understandable architecture. On top of that all tools are free for students, and the Launchpad cost is less than $5.00. These characteristics make it ideal for a first course on microcontrollers, since now all the students are able to afford it and work either at the laboratory or at home. We have taken an approach in which a more complete and expensive platform such as the MSP430FG4618/F2013 Experimenter board is being used for the laboratory exercises, and the Launchpad is being used for homework assignments. The other difference is that the laboratory exercises are based on learning assembly language, while the homework assignments deal with using the Launchpad to control different devices using C language, code examples, graphical design tools.

9 Conclusions

As the title says *"So Many Educational Microcontroller Platforms, so Little Time!"* choosing a single platform is very difficult because there are so many astounding educational development systems that instructors get excited about and would like to be able to present so students could experience the different types of traits each platform brings. While the selection of the teaching platform for an introductory course to second year engineering technology seems somewhat arbitrary among so many options, we started with the MSP430 for sentimental reasons, and we have not been able to find another platform which could deliver the same cleanness in introductory concepts for our students. Since we started using the MSP430FG4618/F2013 experimenter board almost three years ago, the Launchpad was released at less than $5.00, and also Grace was released. So far students are enjoying their experience with a very simple low cost educational device, but at the same time are aware of other architectures, such that they should be able to discern what to use on academic or industrial projects. Electrical engineering technology students take at least two more courses on embedded design, and computer engineering technology students take at least five more. These latter students get to play with one or more educational platforms in advanced courses and also for their senior capstone course.

10 Acknowledgments

We would like to thank Dr. Roy Melton and Dr. Lisa Hermsen for helping us proof reading the final document.

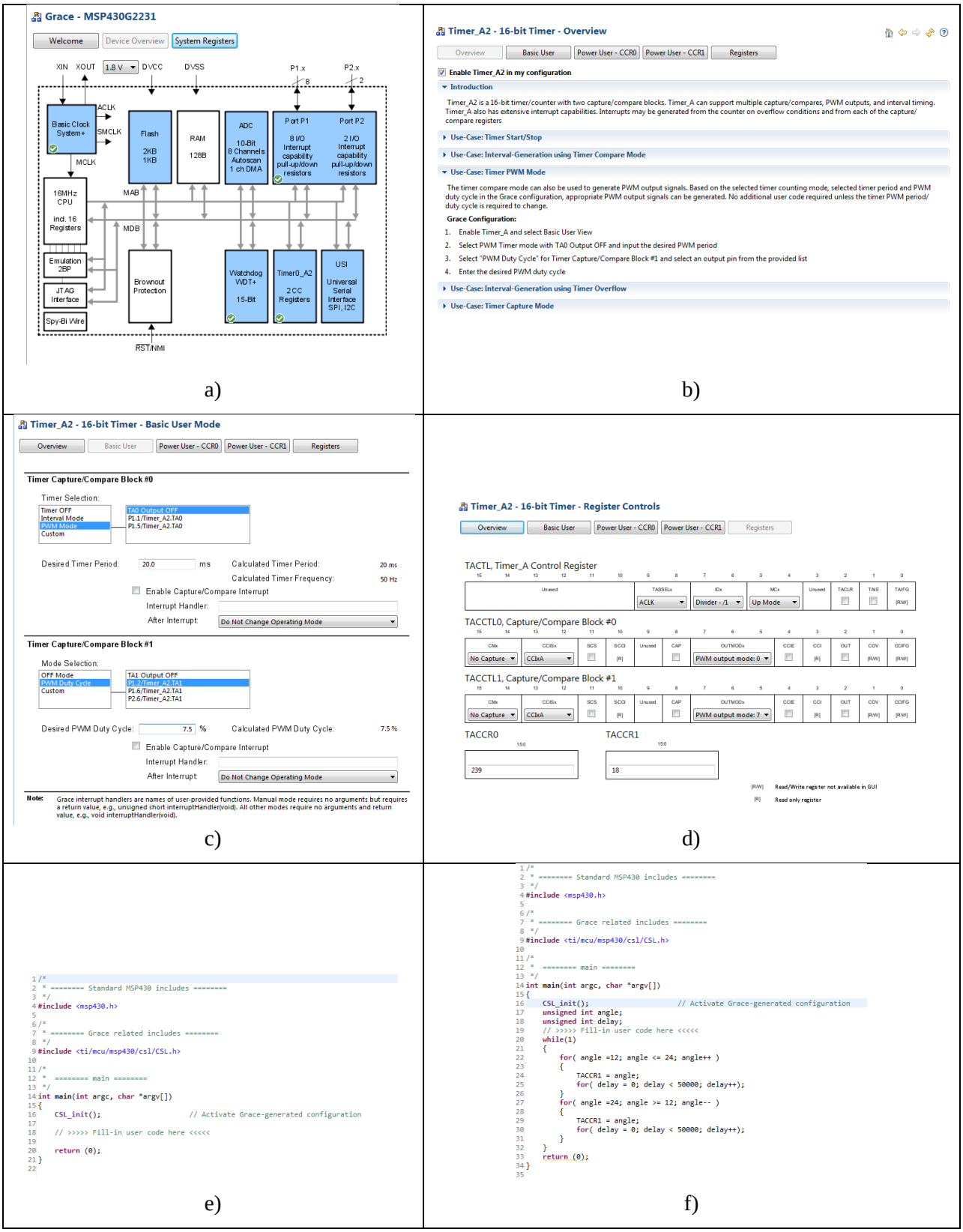

Figure 2. Grace exploration flow to control a servo.

11 References

[1] (2012). *Arduino - HomePage*. Available: http://www.arduino.cc/

[2] (2012). *Rapid Prototyping for Microcontrollers | mbed*. Available: http://mbed.org/

[3] (2012). *Engineering and Technology Jobs - IEEE Job Site*. Available: http://careers.ieee.org/

[4] (Posted). *Find Jobs. Build a Better Career. Find Your Calling. | Monster.com*. Available: http://www.monster.com/

[5] (2012). *Yahoo! Careers*. Available: http://us.careers.yahoo.com/

[6] D. Heer, R. Traylor, T. Thompson, and T. Fiez, "Integrating computer engineering education with a platform for learning," in *Frontiers in Education, 2003. FIE 2003. 33rd Annual*, 2003, pp. F2F-17-22 Vol.2.

[7] T. Stoltz, M. Paulik, and N. Al-Holou, "A Microcontroller Laboratory Hardware Platform for the Academic Environment: The UDM-EVB," in *Frontiers in Education, 2005. FIE '05. Proceedings 35th Annual Conference*, 2005, pp. S2G-S2G.

[8] W. A. Stapleton, "Microcomputer Fundamentals for Embedded Systems Education," in *Frontiers in Education Conference, 36th Annual*, 2006, pp. 6-10.

[9] A. F. Mondragon-Torres and IEEE, "Work in Progress - Ultra-Low Power and the Millennium Generation," *2010 Ieee Frontiers in Education Conference (Fie)*, 2010 2010.

[10] C. Bonnington. (2012, 01/16/2012). *ARM-ed to the Teeth, Arduino Hardware Grows Up | Gadget Lab | Wired.com*. Available: http://www.wired.com/gadgetlab/2011/09/arduino-arm-products/

[11] mbed.org. (2012). *m0 release - Handbook | mbed*. Available: http://mbed.org/handbook/m0-release