



Innovative Embedded Systems Project

Mr. Stephen A. Strom, Pennsylvania State University, Erie

Stephen Strom is a lecturer in the Electrical and Computer Engineering Technology department of Penn State Behrend, and holds a B.S. in electrical engineering from Carnegie Mellon University. His career includes over thirty years experience in designing and programming embedded systems and has multiple patents for both hardware designs and software algorithms

Innovative Embedded Systems Project

Abstract

The goal of an Embedded Design course is to teach programming, embedded circuits and software algorithms. There are also related goals, such as learning to use development tools, lab equipment and proper debugging techniques. There are tangential goals such as working in teams and preparation for senior/capstone projects. And while there are a variety of approaches to teaching, the common theme is that you want the students to be successful and understand the material as thoroughly as possible.

Each year, there are new peripherals, new processors and most engineering departments do not take a static approach to education and try to include more information in their classes. Technology students (in general) require a more hands on approach in that learning is accomplished via lab projects as opposed to lectures and homework. Thus an Embedded Design course with the proper lab work is the best way to achieve all of the associated goals.

This paper details an approach that I have used in my (junior level) microprocessor course in which the students (some with a strong interest in programming, some with less) all learn to program in 'C', and interface the PIC processor to a wide variety of peripherals using a development board that they build themselves. What is unique is that the PCB was designed in-house using Eagle Cad and contained a wide variety of components not present in typical off-the-shelf development products. This allows us to teach a wide range of sensors (temp, light, range, etc.) and a variety of protocols (RS232, SPI) as well as software debouncing and filtering algorithms.

The final part of the course was to let the students come up with their own ideas for a final lab project and each student took a unique approach in applying their hardware and software skills. With the success of this course, it is hoped that the information provided here could be used as a framework for other classes with similar goals and objectives.

Overview

Engineering students love to design. Technology students love to build. In most engineering courses, greater attention is placed on application and circuit design as opposed to construction and debugging. However, if you can marry these two concepts together, then you have hit a home run.

In our school, the Computer Engineering Technology curriculum is set up so that programming and embedded circuit design is taught via a series of courses: (a) Introduction to 'C' programming, (b) Digital Design and Embedded Systems, (c) C++ and object orientated programming and (d) an Intermediate Embedded Systems course. After completing this series, the students can continue on with an Embedded Linux course, as well as a Software Engineering (process and agile development) course.

This arrangement of courses is a bit unusual in that Technology departments do not always offer embedded systems designs to this depth; however, it is one that really prepares the student to succeed as an embedded programmer/designer. But with this said, success in a curriculum does not always relate to the number of courses available, but its content. To optimize the learning process (motivating a student to learn as much as possible), the courses include Lab's in which the students can build and create embedded applications. The goal of this paper is to show the methodology that was used in the Intermediate Embedded Systems course.

Introduction to the Intermediate Embedded Systems Course

The Intermediate Embedded Systems course is a required junior level course that includes both lecture classes and a lab. It has the following objectives:

- Learn additional 'C' programming and as well as knowledge of embedded programming algorithms.
- Be able to create, and debug a 'C' program using Microsoft Visual Studio.
- Understand hardware and software capabilities of the PIC 18F2520 microcontroller and compile/debugging using MPLAB.
- Understand component specifications using manufacturer data sheets.
- Learn/implement designs with a variety of peripherals.
- Develop embedded applications using the PIC microcontroller.
- To extend and augment topics covered in the earlier computer and circuit courses.

A large portion of the course is spent on embedded application programming and interfacing a PIC microcontroller to a variety of components using the general purpose IO ports, A-to-D converter, Pulse Width Modulator, RS-232 and SPI ports.

In the past few years, I start the students off with a Visual Studio (lab) application. This helps to refresh the students on 'C' programming and gives them time to purchase the components, assemble the board and run the start-up test program.

Course Structure / Lab Kit

The best way to think of a (lab based) embedded design course is to come up with a series of labs, each one sequentially more difficult and includes components (both software and hardware) from the previous lab. Each week, the lecture revolves around introducing new components to the class that support the upcoming lab. This includes new hardware concepts (RS-232, SPI, PWM, A-to-D, general purpose I/O) and their associated peripheral circuits. Also, software (C programming) concepts (such as #defines, MACRO's, functions, arrays, floating point, etc.), and software algorithms (switch debouncing, analog averaging, rotary encoder interface, etc.)

And while this is occurring, the class is also covering operation of various development tools, structured programming techniques and discussing hardware components and software architecture. When you look at all of the material, it appears to be an overwhelming task.

In past years, most of the labs required the students to use a solderless breadboard to construct and debug their circuits. At that time, the students initially place their components onto the breadboard and would work to cut their wires to length and place them in rows/channels. If you were willing to spend time laying out the wires you can get a clean layout as seen in figure 1.

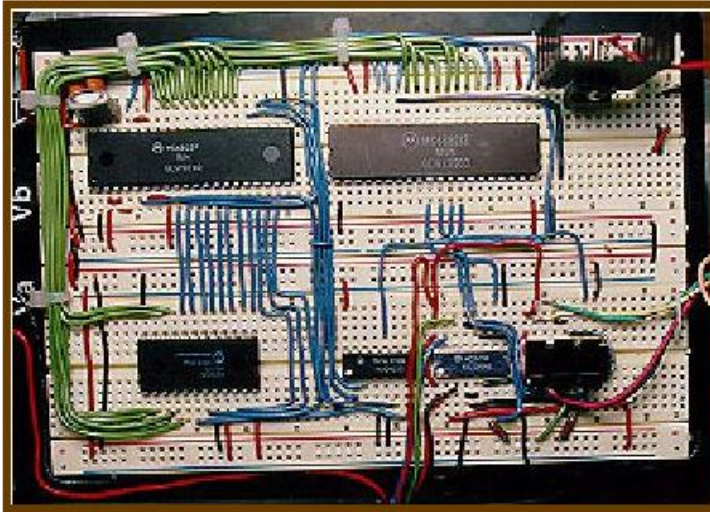


Figure 1. Solderless breadboard with wires

As the weeks progress, some of the wires get removed and others inserted, and little by little the boards grows to a complicated arranged of components and inter-connections as seen in figure 2.

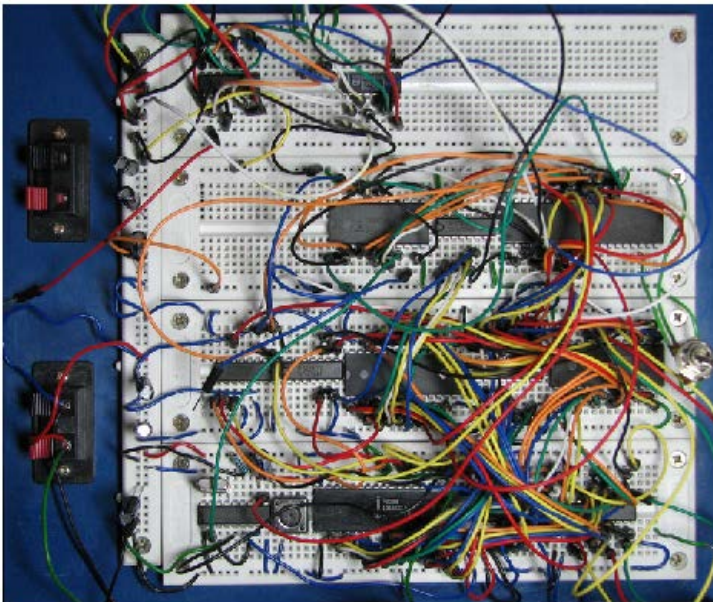


Figure 2. Solderless breadboard with ratsnest

The end result is that the students spend more time debugging loose wires and each time they place scope probe on the board, another wire wiggles loose. The end goal of the lab changes from learning about circuit design, to one of getting something to work and to get out of the lab as quickly as possible.



Figure 5. Assembled board

The end result is that a student with no soldering experience has a high probability of assembling the board without any problems. Adding to the objectives of the course, is the teaching goal to learn about hand soldering and surface mount component soldering.

Course Development

Over the years, the course had been set up so that either (a) the students purchased their own components, or (b) purchased a completed board or (c) purchased nothing and had to “check-out” the school’s development system. In looking at the three options, one observation is that it is better to have students purchase their own boards rather than to borrow one. This gives them a sense of ownership and they take better care of the components.

Another characteristic of owning the board is that they bring the board home with them and work on them at night. Some are inspired to create their own programs (outside of school) as well as use them as a basis for their Senior design projects.

And while it is possible to find an existing off-the-shelf board, there are several factors to consider. In looking at the variety of embedded development boards on the market, you should consider:

- Cost – The target cost should be no more than \$100 (cost of a book)
- Peripherals – The components on the board must satisfy the objectives of the course
- Development tools – The cost of the compiler/programmer/debugger is also part of the overall component cost.

There are a variety of existing microcontroller boards on the market, figures 6 and 7 have examples of low cost embedded development boards, but individually, neither of them meet the objectives of the course. There are more expensive choices (such as figure 8), but the cost of it exceeds the \$100 target range.

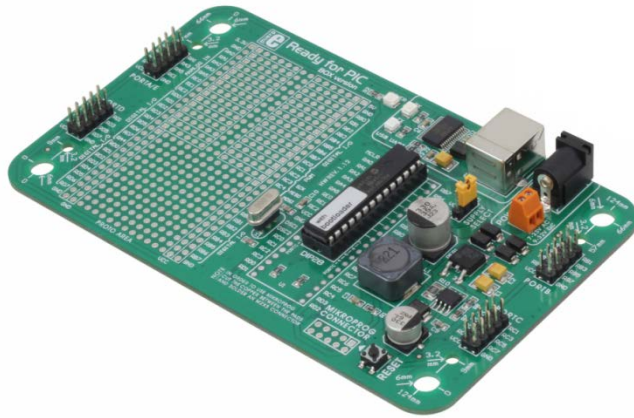


Figure 6. Pic Development Board \$30 ⁽¹⁾
Lacks enough peripheral components to be used for the course

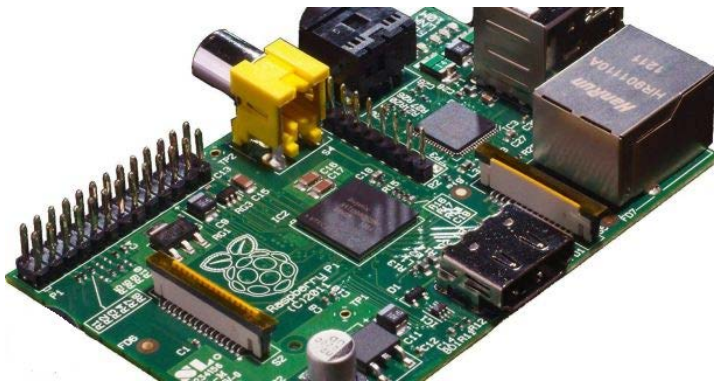


Figure 7. Raspberry PI \$40 ⁽²⁾
Accessing the peripherals via the Linux O/S is not one of the objectives for the course



Figure 8. Embeded SBC with touch display \$250 ⁽³⁾

The board that was developed for the course costs less than \$45 and has everything that is needed to support the course outcomes, plus it gives the students the opportunity to solder and build.

Example Labs

The board that was developed for the course included the following components:

- PIC18F2520 microcontroller
- Tri-Color LED
- 2 momentary push buttons
- Rotary Encoder (2 bits)
- LCD Display (2x16 character, over RS-232)
- Piezo-Buzzer (connected to the PWM output)
- Thermistor
- Photocell
- ADXL345 3-Axis Accelerometer (over SPI)
- Socket for XBee / Bluetooth / GPS (students can optionally purchase these)
- Expansion connector to bring out:
 - 2 A-to-D inputs, 1 Digital I/O, PWM, and RS-232 signals

With these components, a series of labs were developed, where after the board was built and tested, they would spend a week on each lab project. Once a (peripheral interface) function gets created, it could then be used in the next lab. The intent of the labs is to build a library of functions that can access each peripheral device and to use this library in the final project. This meets the goal of teaching students the concept of modular programming – the ability to port code from one application to another.

- Tri-color LED and two input push buttons (general purpose I/O)
This is the first lab and it is used as an “intro” to MPLAB as well as PIC programming. My suggestion is to expanded this to multiple labs, where the end goal is to teach digital outputs and inputs. The second lab can include debouncing of push buttons, and software PWM to adjust the brightness of the LED’s.
- LCD display (interfaces over RS-232)
This lab includes operation of the LCD display, as well as the basics of RS-232 communication. Once the LCD is up and running, the display can be used for all of the remaining labs.
- Hardware Timers (and timer interrupts)
Learning interrupts is essential for all embedded systems designs. There are a lot of choices to demonstrate this; my suggestion is to use the printed circuit board as a stop watch. This will encompass all of the previous components (display, push buttons, etc.).
- Thermistor and Photocell (and Analog to Digital conversion)
This is the first use of an analog input. The end goal of the lab should also include learning about averaging, low-pass filtering, using hysteresis in the control equation (for a thermostat), as well as differences between integer and floating point conversions.

- Pulse Width Modulation (and piezo-speaker)
The basic lab is to use the PWM signal and generate audio from the piezo-buzzer. Integrating in with the photocell, you can have the students build a simple theremin.
- Rotary Encoder
For some reason, students have always had trouble writing code to interface to a rotary encoder. You really need to focus the lab around this component in order for them to have enough time to write/debug the interface routines.
- Accelerometer (interfaces over SPI)
This lab should encompass methods of filtering and analyzing the acceleration data, and can also include references to using SPI as a network.

Final Lab Project

The last lab of the course is one that the students can pick for themselves. It must use the microcontroller, but does not have to use the printed circuit board. It can be completed individually, or in teams. The students can add in new circuitry, and/or connect to an existing program such as LabView⁽⁵⁾ or Matlab⁽⁶⁾. In the past, over half of the students needed to do some hardware design as well as purchase additional components to complete their project. They were also asked to create flowcharts and schematics for their design, as well as give a presentation (example flowcharts and schematics shown in figures 9 and 10).

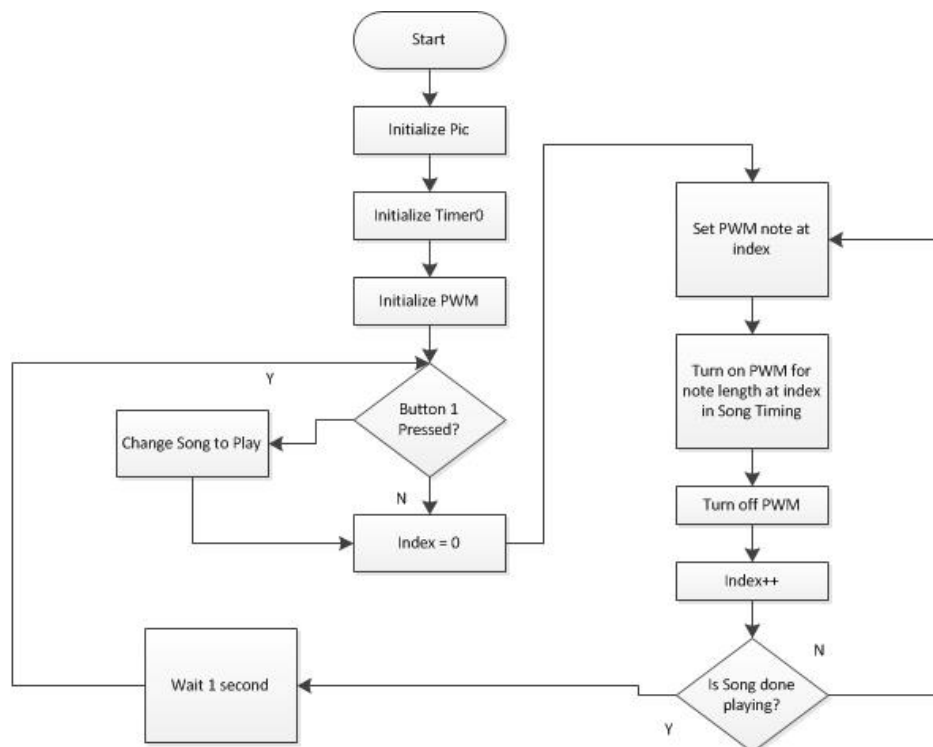


Figure 9. Example Flowchart for an audio application

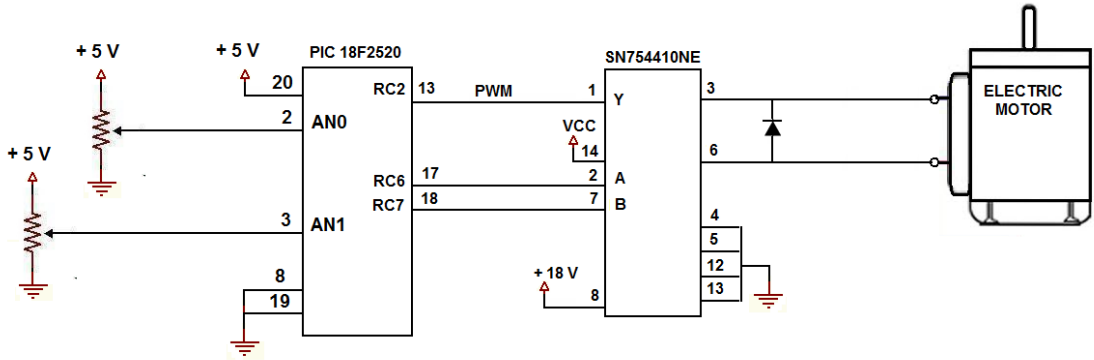


Figure 10. Example schematic for a motor controller application

Some examples of the final project include:

- Laser pointer targeting game (uses just the on board components)
You would use a laser pointer to “shoot” at the photocell on the board and this would count as a hit. You need to “hit” the target as many times as possible within 30 seconds.
- Motor speed control using a joystick (the PIC18F2520 was removed and a custom interface circuit was designed)
The joystick would be used to set the speed of the motor and the A-to-D monitors (rotational) feedback regarding the actual speed.
- Portable GPS (connected the pcb board to a GPS device over RS-232)
Created a portable GPS system that can be used for geo-caching⁽⁴⁾.
- Temperature monitoring/control (interfaced to LabView over RS-232)
The pcb monitored the temperature and LabView provided the front panel display (figure 11).

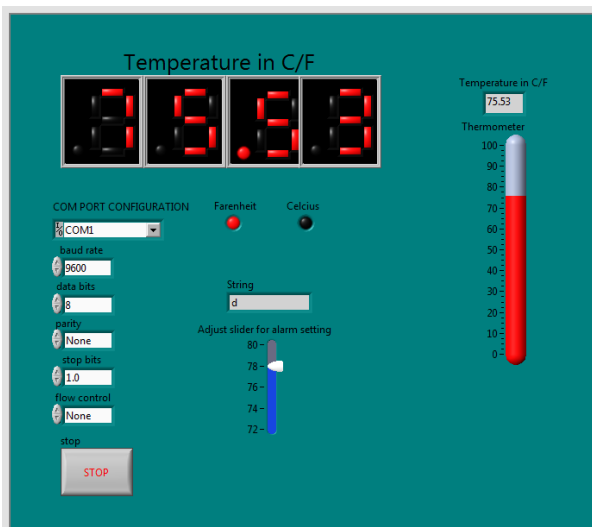


Figure 11, LabView front panel example

Conclusions /Recommendations

I have taught this class where the students only used breadboards, as well as a combination of breadboard's and pcb's and currently, I now teach the class where the material is centered around the pcb. In speaking with the students (each year), this was one of the few classes where every student liked the "pcb-only" approach.

There is no question that they would not have accomplished as much using just breadboards for their designs. The focus of the labs could be on their software development, as opposed to connecting (and debugging) of wires. I noticed that the students that normally struggle with computer programming could now focus on learning the lab content as opposed to going off on tangents debugging loose wires. It was more rewarding and a lot less frustrating for them.

And while there was a loss of some hardware design, it was greatly offset by depth in the software projects. There was still some hardware design, as they needed to design a power supply, and design peripheral circuits for their final lab, plus there was a variety of homework questions on peripheral interfaces.

The biggest change was that there was an increase in motivation to design their final lab project and to show off their designs. Compare to previous years, the attitude of the students changed to one of willing to show off their design, as opposed to (previous years) where they were developing the "minimum" to complete the course. An extension to the course would be to add a writing component to the final design that would include a power-point presentation, similar to what is done with senior projects.

I suggest that these same results could be replicated in other Computer Engineering Technology courses using a similar approach.

Bibliography

1. Ready for PIC, <http://www.mikroe.com/pic/development-boards/>
2. Raspberry PI, <http://www.raspberrypi.org/made-in-the-uk/>
3. Embest SBC8018, <http://www.embeddedworks.net/>
4. Geo-Caching, <https://www.geocaching.com/>
5. LabView (National Instruments), <http://www.ni.com/labview/>
6. MatLab (MathWorks), <http://www.mathworks.com/products/matlab/>