120th ASEE Annual Conference & Exposition
FRANKLY, WE DO GIVE A D*MN
June 23-26, 2013

Paper ID #7393

Using the Parallax Propeller for Mechatronics Education

Dr. Hugh Jack, Grand Valley State University

Hugh Jack is a Professor of Product Design and Manufacturing Engineering at Grand Valley State University in Grand Rapids, Michigan. His interests include manufacturing education, design, project management, automation, and control systems.

Using the Parallax Propeller for Mechatronics Education

Abstract

Microcontrollers have become a mainstay of mechatronics laboratories. For example, the Arduino boards, and shields, are low cost flexible hardware that can provide substantial capabilities. At Grand Valley State University all engineering students learn to program microcontrollers using Atmel ATMega processors, the same processors used on the Arduino boards. In the mechatronics course, EGR 345 - Dynamic System Modeling and Control, the students use Parallax Propeller based hardware. The alternate, Parallax Propeller, hardware platform broadens the students' knowledge and gives them access to a multiprocessing environment.

The paper objectively outlines the hardware/software platform and how it can be used in a mechatronics course for Manufacturing Engineering students. The supported topics include data collection, feedback control, various sensors, networking, and human interfaces. Educational activities include laboratory work and small projects.

Introduction

A computation platform is the backbone of any introductory course focused on mechatronics and/or modern controls. The number of available platforms easily reaches into the thousands. However for the purposes of education there are a few wise alternatives. The typical selection criteria for these systems are:

- Cost for hardware and software;
- Programming knowledge and student prerequisites;
- Capabilities;
- Electronic interfacing complexity and options;
- Built in capabilities and functions; and
- Adoption by industry.

Naturally each platform has strengths depending on the focus of the academic program and the students. The two major divisions for control systems are industrial and embedded. Industrial control systems normally include Programmable Logic Controllers (PLCs), Programmable

Automation Controllers (PACs), System Control and Data Acquisition (SCADA), Numerical Control (NC), etc. Programs with an industrial focus will have at least one course in industrial controls[1]. Embedded controls use some sort of microcontroller, or full computer, interfaced to sensors and actuators. Embedded control system topics are an essential part of every electrical and computer engineering (ECE) program. Embedded control systems courses are increasingly available for many other programs, such as Mechanical Engineering (ME) and Chemical Engineering[2]. For the purposes of discussion the term Mechatronics will be used to describe a mechanical engineering approach to industrial and embedded control.

Student prerequisite knowledge varies by curriculum. At the lower end of the scale, students have not had a programming course, no digital circuits course, and little procedural problem solving[3]. When the Mechatronics course does not require much prerequisite knowledge the instructor often selects a system with a very shallow learning curve. There are a few approaches that have been used including the Lego Mindstorms NXT[4] and Parallax Basic Stamp 2[5]. Mindstorms based experiences have been used successfully by educators working with high school students, and college freshman. Students construct the system using the ubiquitous blocks and then they construct simple graphical control programs. This approach provides exposure to the programming concepts and the context for the system components[6,7].

The systems with shallow learning curves are not a replacement for embedded system controls. However, they have proven very effective when dealing with students who are reluctant to program and/or do not have the right prerequisite knowledge. When self-motivated students have already learned the fundamentals of programming, they are much better off to focus on a much more mature platform. An example of the mature platform is the Arduino controller[8] which uses the Atmel ATMega processor[9]. The Arduino platform is designed to be modular for the hobby market, but the core processor is industrial grade and used in many commercial products. The processor can be programmed in languages such as C and assembly language. The basic Arduino system can be expanded using *shields*[10] produced by hundred of third party vendors and open source advocates. These shields range from basic input-output boards[11], to touch screen displays[12], and cell phone SMS/Data capabilities[13]. Given the variety of hardware and large number of support websites[14] these are one of the best choices for a mechatronics curriculum. The Raspberry Pi computer[15] is also showing potential for this role. However, these platforms do require some knowledge of microcontroller level hardware interfacing.

The Parallax Propeller[16] occupies a middle ground between the Mindstorm and Arduino platforms. It is possible for a new programmer to produce a functional control system with inputs and outputs using the Spin language. The hardware has eight processors, called *cogs*, using 32K shared RAM and 32 IO pins. Each processor has 2K of RAM and can run up to 80MHz each. Instead of using interrupts, programmers write subroutines that run on separate cogs. For example, on a typical microcontroller a pulse width modulated (PWM) output is generated using

a system counter. To generate a PWM output in SPIN, one cog would run an endless loop that counts on and off steps, forcing the pin manually on and off. Interestingly, the Propeller processor also has a built in capability to generate regular VGA output signals.

Surprisingly, the only ASEE paper to reference the Parallax Propeller hardware was written by Fraser[17]. He switched to the platform in 2010 from a PIC processor. His assessment was "The Parallax environment is far superior to the PIC development environment making for easier learning by non-computer science students." This authors experience is similar, by replacing interrupt driven programming with the polled nature of SPIN, students were able to understand and apply concepts in less time. Some of the advantages and disadvantages of the processor include:

Advantages

- 8 processing 'cogs' make multitasking simple (i.e. no interrupts).
- A small learning curve for the 'spin' language.
- Numerous code examples are available on the Internet including the manufacturer's Object Exchange[18].
- There are some built in functions, such as driving video.

Disadvantages

- Limited abilities for high-level tasks.
- Limited memory.
- No in circuit debugging.
- Options for registers and hardware are limited. e.g., no A/D.

An example of a SPIN program is shown below, to pulse an output pin with a 50% duty cycle. The first three lines of the program define the constants that set the clock speed to 80MHz. Although it is a bit cryptic, beginners accept this quickly. The `VAR` definition includes a few bytes for the stack of the `pulse()` thread. The `X` value becomes a global variable. The `Main` routine sets the flag value, `X`, to 0, and sets pin 16 as an output. The `cognew` function starts another processor cog running the `Pulse` function. After this there are two threads of execution running in parallel. The following is an endless `repeat` loop. `outa` is used to set an output pin value, and the `waitcnt` delays until the system clock, `cnt`, matches the given value.

```
CON
     _clkmode = xtal1 + pll16x ' These two lines set
     xinfreq = 5_000_000       ' the clock to 80MHz
VAR
     long SqStack[6] ' Stack space for Pulse cog
     long X          ' timer flag
```

```
PUB Main ‘ A program to set output 16 with a counter X
     X := 0              ‘ Initialize X
     dira[16] := 1       ‘ Pin 16 is set as an output

     cognew(Pulse(), @SqStack) 'Launch Pulse cog

     repeat              ‘ and endless loop
          if X == 1 ‘ Switch the output pin to match X
               outa[16] := 1 ‘ Turn on pin 16
          else
               outa[16] := 0 ‘ Turn off pin 16

PUB Pulse()  ‘ Switches X on/off at 50% duty cycle
     repeat ‘ another endless loop – on another cog
          waitcnt(1_000_000 + cnt) ‘ Wait 2 million cycles
          X := X + 1     ‘ Cycle the flag
          If X > 1       ‘ Keep the value either 0 or 1
               X := 0;
```

The Course

The course in question is offered to junior level students in the Product Design and Manufacturing Engineering program at Grand Valley State University. The students have had sophomore level experience programming in C on Atmel processors. This course is entitled Dynamic System Modeling and Control[19]. The course topics[20] begin with developing differential equations for mechanical and electrical systems, solving the differential equations explicitly, and numerical methods. The course moves into feedback control, motion control, sensors, actuators, and LaPlace transforms. The course is a combination of 3 hours of lecture and 3 hours of lab per week for a semester. The laboratories cover a variety of issues from basic modeling to system control.

In the past the course has used a number of platforms including the Parallax Basic Stamp 2, Motorola 6811, and Atmel ATMega 16/32. In 2010, the Parallax Propeller system was tested in one laboratory experience. The outcome was very encouraging and this was expended to cover more laboratory experiences in 2011, and was fully deployed in 2012.

Hardware

Students buy their own controller for the course, a P8X32A QuickStart board from Parallax for $25, or from Radio Shack for $40. The reason for student purchased hardware is twofold. When

they own their hardware they are likely to use it for other projects in the future. And, experience from previous years showed that students are much more careful with hardware they own.

Although the students own the main control board, various sensors and actuators are provided. The standard items available from the lab supplies are listed with costs.

User Inputs and Outputs:	
Trackball Module	$15.99
2-Axis Joystick	$3.99
2x16 Parallel LCD	$29.99
uOLED-128-G1 -128x128 color pixel display	$49.99
Sound PAL - Polyphonic audio output	$14.99
Input Sensors:	
4-Directional Tilt Sensor - measures tilt angles in two axes	$9.99
ColorPAL - detects colors in RGB	$19.99
External GPS Antenna	$9.99
RXM-SG GPS Module	$49.99
Memsic 2125 Dual-axis Accelerometer	$29.99
MMA7455 3-Axis Accelerometer Module	$29.99
Piezo Film Vibra Tab - voltage is proportional to deflection	$2.50
Ping Ultrasonic Distance Sensor	$29.99
PIR Sensor - triggers on a change in room temperature	$7.99
QTI Sensor - optical proximity sensor	$9.99
Sensirion Temp/Humid. Sensor	$42.99
Sharp IR Distance Sensor	$12.99
Actuators:	
4-Phase / 12 Volt Unipolar Stepper Motor	$14.99
Brushless Outrunner Motor 1000 Kv	$21.99
HB-25 Motor Controller - H-bridge module	$54.99
Little Step-U Motor Controller	$69.99
Propeller Servo Controller USB	$39.99
Futaba Continuous Rotation Servo	$12.99
Futaba Standard Servo	$12.99
Special:	
RFID Tag	$1.00
RFID Card Reader Serial	$42.99
XBee PRO 60mW transceiver	$36.99
2-channel 12-bit A/D converter with SPI interface	$3.50
4-channel 12-bit A/D converter with SPI interface	$4.75
Netburner Kit - adds web server and email capabilities	$129.99

Students normally worked in pairs in the laboratory. Simpler exercises or tutorials were offered early in the semester to familiarize the students with the hardware and programming approaches. They were encouraged to create and maintain source code for reuse later. As the semester progressed, weekly laboratory work was assigned, along with some outside projects. These provided a useful mix of creative work on projects, and disciplined work on system modeling and data collection. Weekly exercises included:

Week 1: Propeller tutorial
http://sites.google.com/site/rototypeit/home/microcontrollers/parallax-p8x32a-quickstart-board-tutorial
Week 2: Students given higher-level sensors and actuators. They do mini-projects due in one week.
Week 3: Project demonstration and assignment of higher-level devices. Mini-projects due in three weeks.
Week 4: Students create a mini data collection device.
https://docs.google.com/document/d/1jhQUr94gun_I_a_EWBdmjFYP4dRcbvqwfWzlLlTDbus/edit
Week 5: Students use the data collection device to analyze a torsional pendulum.
https://docs.google.com/document/d/1lb1b05qEjIymNQKzA4XZQ--yPvfAUAR5wCPBtyNDdh0/edit
Week 6: Motor Speed Control -
https://docs.google.com/document/d/1Nv9MCpJCWwl3rfAf-n6ReBoWmQuKnbBhW1kCQGbBRoo/edit
Week 7: Feedback Control -
https://docs.google.com/document/d/1Nv9MCpJCWwl3rfAf-n6ReBoWmQuKnbBhW1kCQGbBRoo/edit
Week 9: Nonlinear Feedback Control
Week 10-13: Students create a quad-copter exercise for the labs

The outcome of the course was very positive. Although students had enough knowledge to use the Arduino hardware, they often opted to use the Propeller hardware for fast system prototyping.

Some of the lessons learned along the way were:

- Mixing 5V with 3.3V was a problem at times.
- Students using a USB cable to power more devices hit the 0.5A limit quickly.
- Windows is officially supported, there is a less capable tool for OSx.

Conclusion

The natural parallel structure of the Propeller processor makes it easier for students to break down tasks and write programs. In particular, the cog structure was simpler than interrupts, and easier to debug. Although the author would recommend the use of the Arduino platform for students continuing on to additional Mechatronics courses, the Propeller hardware is very well suited for a mid-level terminal Mechatronics experience.

References

1. Foster, M., Hammerquist, C., Melendy, R., "A Review of Programmable Logic Controllers in Control Systems Education", ASEE Annual Meeting, June 2010.
2. Lodge, K., "The programming of a microcontroller as the laboratory component in process control for undergraduates in chemical engineering", ASEE Annual Meeting, June 2006.
3. Gammon, T., "Introductory Mechatronics Course Created to Fulfill Freshman-Level Engineering Requirement", ASEE Annual Meeting, June 2003.
4. Lego Mindstorms web page, http://mindstorms.lego.com/en-us/Default.aspx, viewed Jan. 2013.
5. Parallax Inc., Basic Stamp web page, http://www.parallax.com/basicstamp, viewed Jan. 2013.
6. Jaksic, N., Spencer, D., "An Introduction to Mechatronics Experiment: LEGO Mindstorms NXT Urban Challenge", ASEE Annual Meeting, June 2007.
7. Whiteman, L., Steck, J., Koert, D., Paarmann, L., "A Class for Undergraduate Technical Literacy Using Lego Mindstorms", ASEE Annual Meeting, June 2007.
8. Arduino website, http://www.arduino.cc/, viewed Jan. 2013.
9. Atmel website, http://www.atmel.com/, viewed Jan. 2013.
10. Arduino Shield List, http://shieldlist.org/, viewed Jan. 2013.
11. Adafruit Proto Shield for Arduino Kit, http://www.adafruit.com/products/51, viewed Jan. 2013.
12. 2.8" TFT Touch Shield for Arduino, http://www.adafruit.com/products/376, viewed Jan. 2013.
13. GPS/GPRS/GSM Arduino Shield, http://www.robotshop.com/gps-gprs-gsm-arduino-shield.html?utm_source=google&utm_medium=base&utm_campaign=jos, viewed Jan. 2013.
14. Arduino Wikipedia page, http://en.wikipedia.org/wiki/Arduino, viewed Jan. 2013.
15. Raspberry Pi, http://www.raspberrypi.org/, viewed Jan. 2013.
16. Propeller General Information, http://www.parallax.com/propeller, viewed Jan. 2013.
17. Fraser, S., "A Community College Perspective of How Ocean Applications Can Enhance Technical Program Course Offerings and Expand Student Opportunities", ASEE Annual Meeting, June 2011.
18. Propeller Object Exchange Library. http://obex.parallax.com/
19. EGR 345 Dynamic System Modeling and Control course web site, https://sites.google.com/site/engineeronadisk/home/egr345, viewed Jan. 2013
20. Jack, H., "Dynamic System Modeling and Control", Lulu Press, 2012. http://www.lulu.com/shop/hugh-jack/dynamic-system-modeling-and-control/paperback/product-20321449.html;jsessionid=0FE494FEB3D31021BE63B452B9EAE983