CSUS EEE 174 Lab - Section 4 - Tuesday

Laboratory Experiment Number 2: Lab Report

**Multicore Propeller Microcontroller**



**Introduction**

The purpose of this lab is to expose you to the Parallax Propeller Microcontroller. The Parallax Propeller is a multi-core chip controller that has an 8 32-bit RISC CPU cores; the cores in the microcontrollers are called cogs. One advantages of the cogs is the ability of to work together to perform tasks that can execute their own special tasks. The programming language allows the developer to program in Spin originally and now C. In this lab, a multiple of exercises will be covered on this microcontroller to understand what the Propeller can do, then a final design will be created as a combination of all the lessons learned.

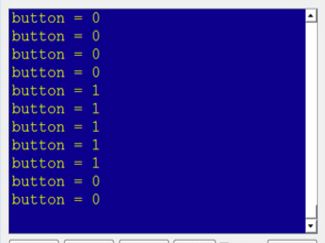
1. **Software, Language, Digital I/O**

In this section, we will set up the Propeller microcontroller software and hardware and also familiarize yourself with common programming and circuit tasks. The Propeller Activity Board uses the SimpleIDE software for programming the microcontroller. The original language that SimpleIDE software uses is Spin language developed by the manufacturer of the propeller, Parallax Incorporated. Recently, in order to achieve the ease of use and more popularity from the microcontroller hobbyists, C language has been added to the software’s libraries.

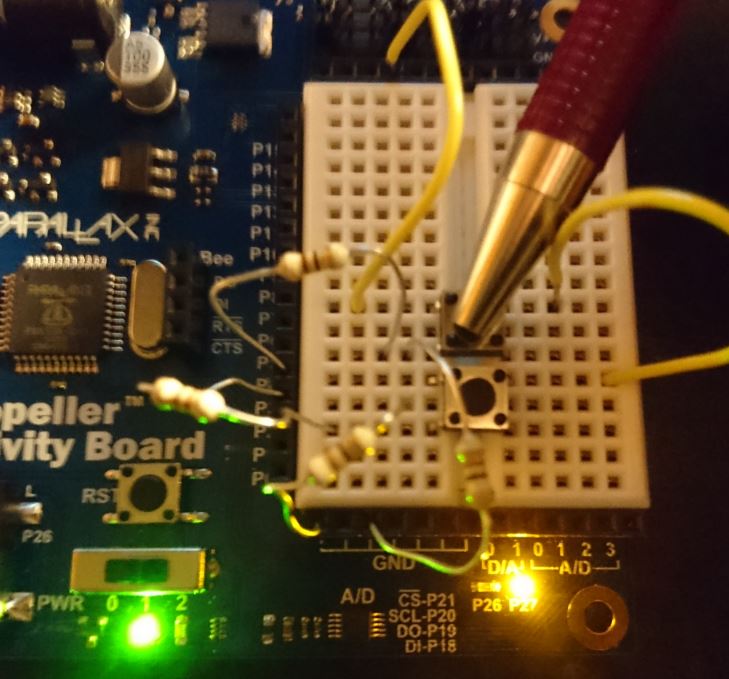
After installing SimpleIDE, the first program that was ran through the instruction and suggestion is to blink a light program. It takes LED output 26 and stays on from our given time then stays off for another amount of time.



Another program that was ran was the “Check Pushbuttons”. By adding two physical push buttons to the circuit and modifying the code to display the push buttons’ statuses as shown below when 0 indicates LOW and 1 indicates HIGH:



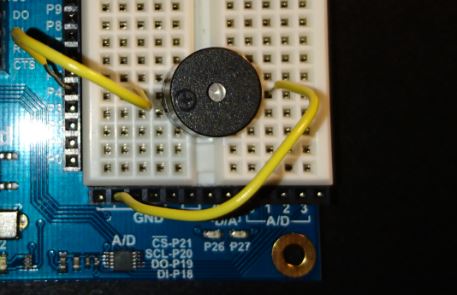
After trying lightning up the LED and displaying the press button, now both were combined so that when button was pressed, the light was up as the picture below:



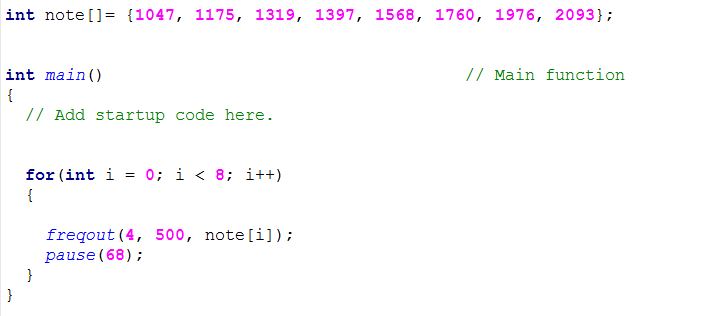
1. **Analog and Timed I/O**

In this section, we will familiarize yourself analog to digital (A/D) and timed input/output (I/O) techniques for monitoring sensors. we also use digital to analog (D/A) and timed I/O techniques for sending outputs to transducers and actuators.

First program that was done in this part was the Piezo Beep as shown below:



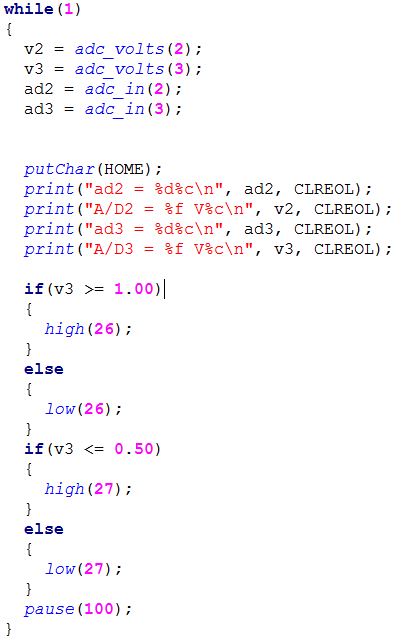
By activating the piezo, the melody of desired sound notes is made because of the preset duration and frequency of each notes. The code for this piezo beep is shown below:



The next program that was worked on in this part is to measure the voltages across provided circuit elements and observe them through the SimpleIDE’s display as shown below:

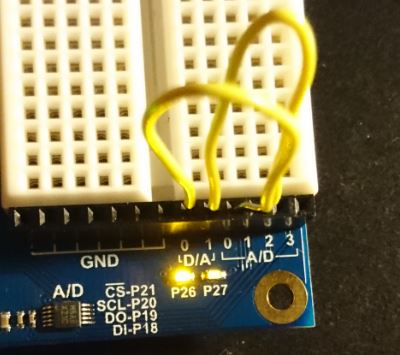


Conditional operation is also examined here by setting certain conditions to set the LED on or off. For example, in the code below, if v3 is greater than or equal to 1.00 V, light will go high, whereas it will go low if v3 is less than 1.00V.

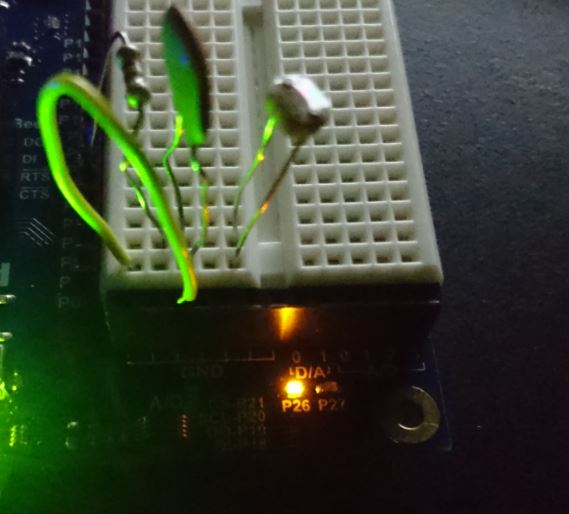


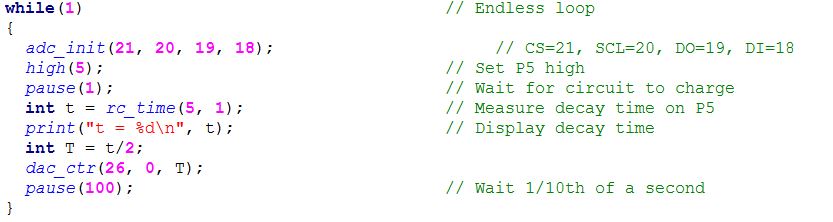
Another function of the propeller is to set voltage level for the circuit. In this program, we manipulate the analog and digital inputs to obtain desired voltage values like below:



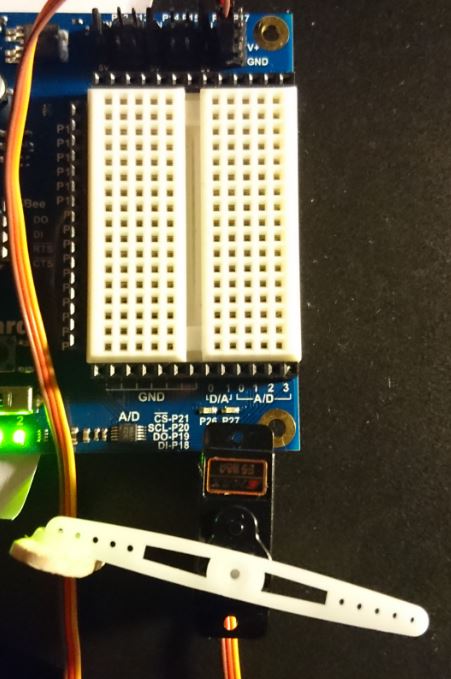


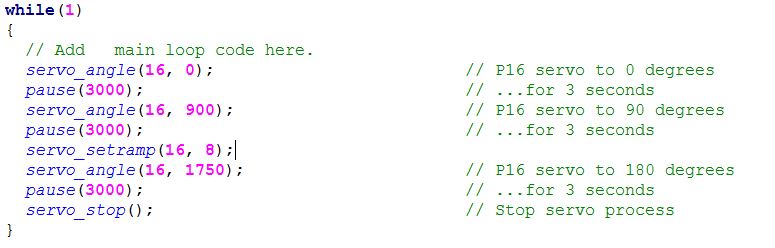
Another program in part 2 is the light sensor. By using phototransistor, capacitor and resistor, a system’s light level will be measured and displayed to LED output number 26 as shown below:



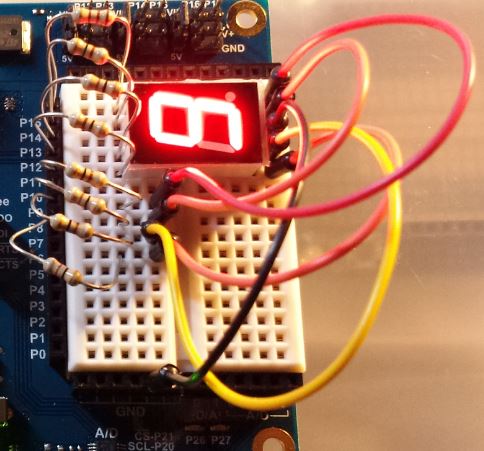


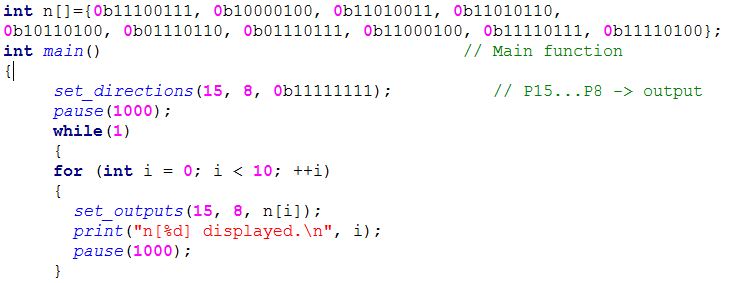
The next program to be examined in this part is the servo control. A servo, in this lab, is made to rotate to certain desired degrees and stop.





The next program in this lab is the seven-segment display LED that is able to countdown from 9 to 1. Using array concept in programming, we were able to control different segments of the device. Also, by delaying different segments with appropriate time, we were able to display numbers and low it down 1 unit at a time. The code and capture of this program are shown below:

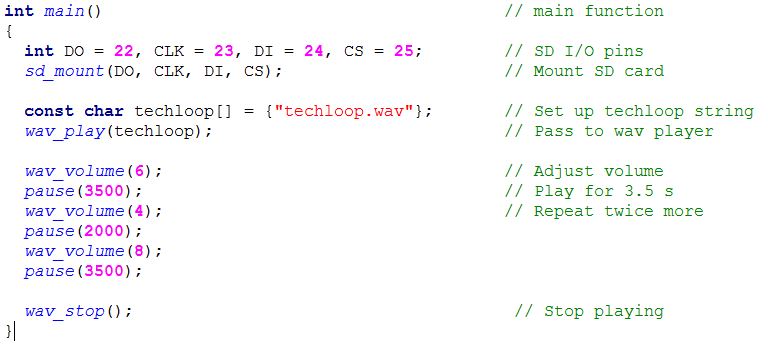




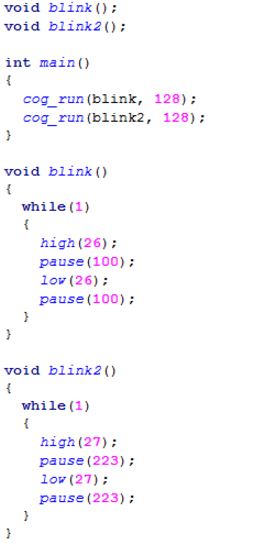
1. **Multicore Approaches, Libraries, and Counter Modules**

In this section, we will familiarize yourself with how to make Propeller microcontroller cores execute function code, how Propeller C libraries are created, and how a core can use counter modules to make processes execute in Parallel with the program thread.

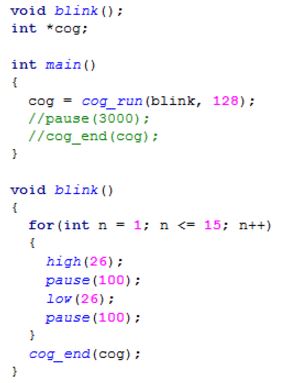
First program that we did in this part is to play sound WAV Files from a micro SD card. By providing the propeller a micro SD card with some WAV files and doing the command lines followed from the instruction, we were able to listen to the files by plug in the earphones or speaker. The code of this program is provided below:



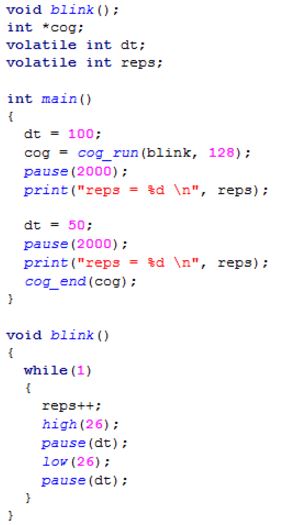
Another interesting function of the propeller is the multicore approach. Multicore helps multiple tasks can be done simultaneously and saves a lot of waiting time. In this exercise, two LEDs were lighted up from their own cores through command “cog\_run().” The source code is shown below:



Besides running multiple cores at the same time, the propeller also provides a function to stop the running cores as the user want. By using the code below, we were able to run a core, make a light blink then turn it off.

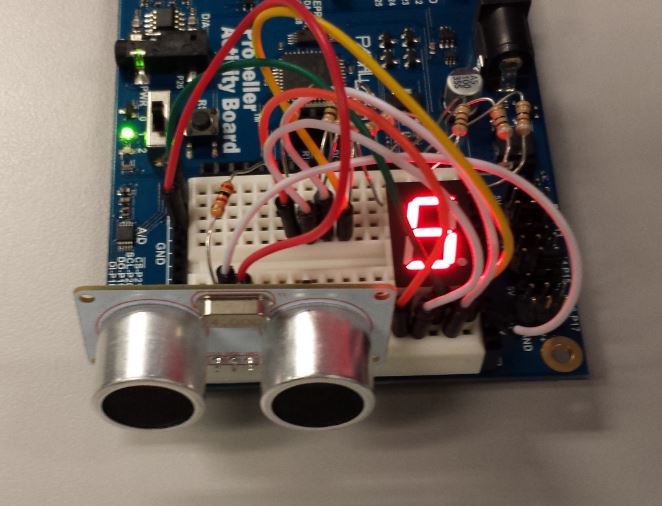


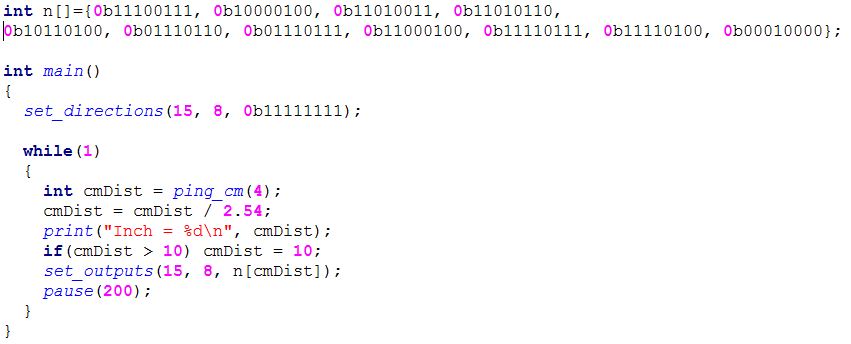
Also, in multicore functions, the data can also be shared between them in real time. By doing that, we were able to make the the other core to perform same task as the current core is doing, such as blinking the light in this example.



1. **Design Project**

In this section, we will combine parts and techniques from the previous sections to make an application. Our project is the combination between the distance sensor and seven-segment display to display how far the object is from the sensor. The capture of the project and its code are shown below:





**Conclusion**

The lab is a great exploration in the Propeller Activity Board microcontroller. By trying different programs and modifying them to serve our projects, we can tell the Propeller is very powerful and is able to as many tasks as the users want. By understanding it thoroughly, the user can take a huge advantage of the Propeller’s multicore function to speed up their work and make it more simply.