**Introduction**

The purpose of this lap is to gain familiarity to the Parallax Propeller Microcontroller. Parallax Propeller is a multicore processor parallel architecture microcontroller chip with 8 32-bit reduced instruction set computer CPUs. First introduced in 2006 by Parallax Inc. Each of the eight 32-bit cores has a CPU with 512 32 bit of data. The Propeller can be clocked using either an internal, on-chip oscillator. By using programming, we can use it to our advantage in many applications. Programing language used are Spin and C.

In this lab, we are going to work on some of its applications and it will be demonstrated in the next few sections.

**Part 1: Software, Language, Digital I/O**

In this part, we are going to install some of the key software and hardware for the Propeller microcontroller. The first program we are going to install is SimpleIDE, which is the early IDE that was used to write and compile code in Spin language, in order to control the hardware. Later, C language was enabled in the software for convenience. C language was much easier and popular between most people.

For this section we are going to use SimpleIDE to build to programs:

1. **Blink a Light** : The following code will make the LED on port 26 turn on then pause then turns off based on our code.

/\*

Blink Light.c

Blink light circuit connected to P26.

\*/

#include "simpletools.h" // Include simpletools

int main() // main function

{

while(1) // Endless loop

{

high(26); // Set P26 I/O pin high

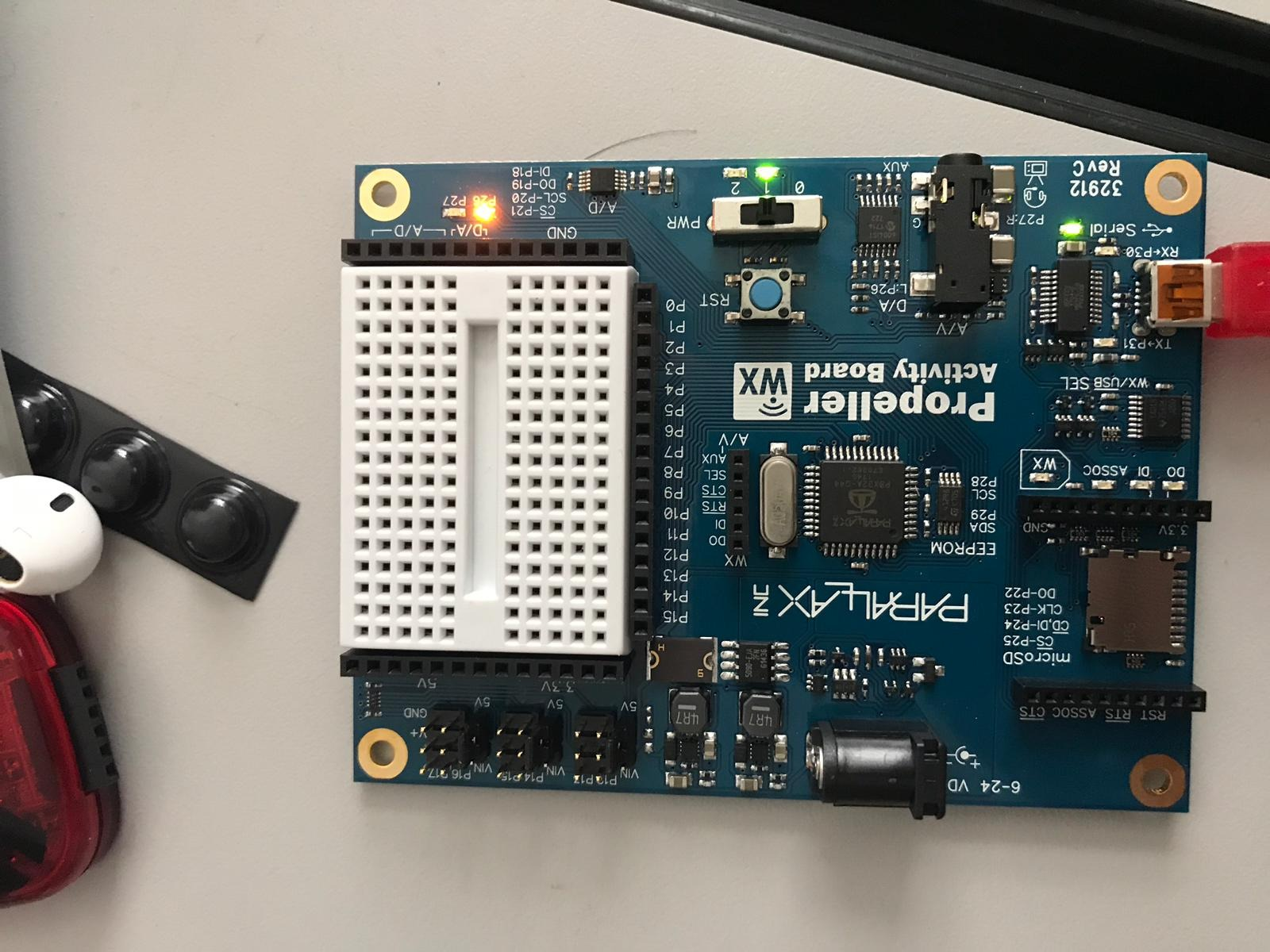
pause(100); // Wait 1/10 second

low(26); // Set P26 I/O pin low

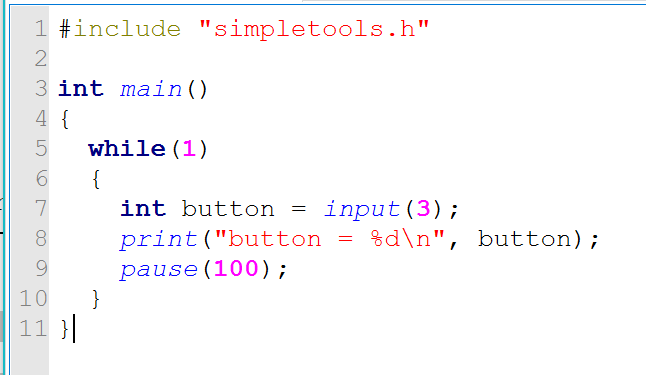
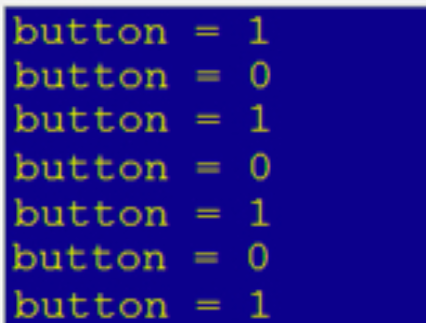
pause(100); // Wait another 1/10 second

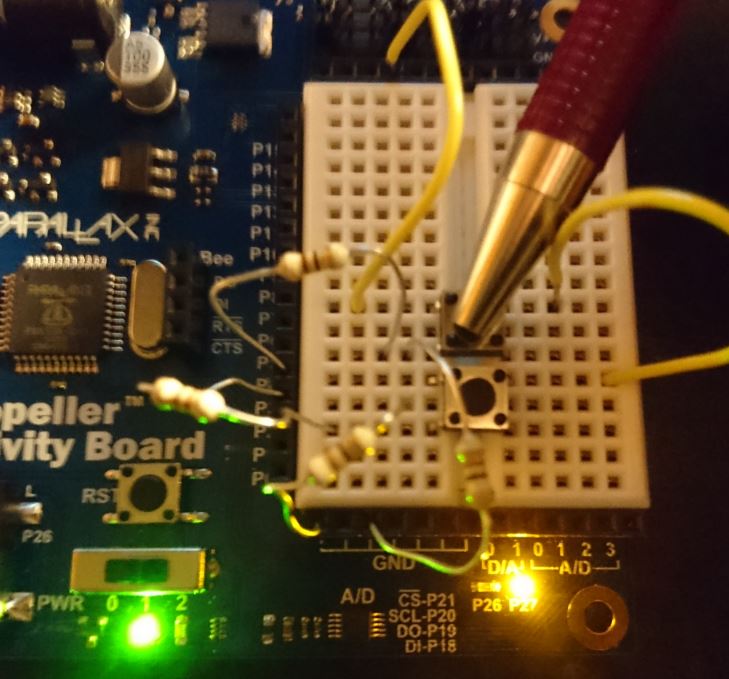
}

}



1. **Check Pushbuttons** : This time we will control the circuits by connecting 2 pushbuttons to it. This example will display the state of the button connected to P3 in the SimpleIDE Terminal.  It displays 1 if the button is pressed, or 0 if it is not pressed.





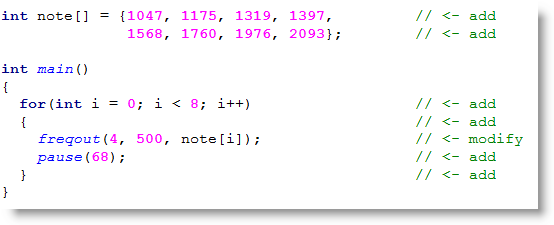
#### Part 2: Analog and Timed I/O

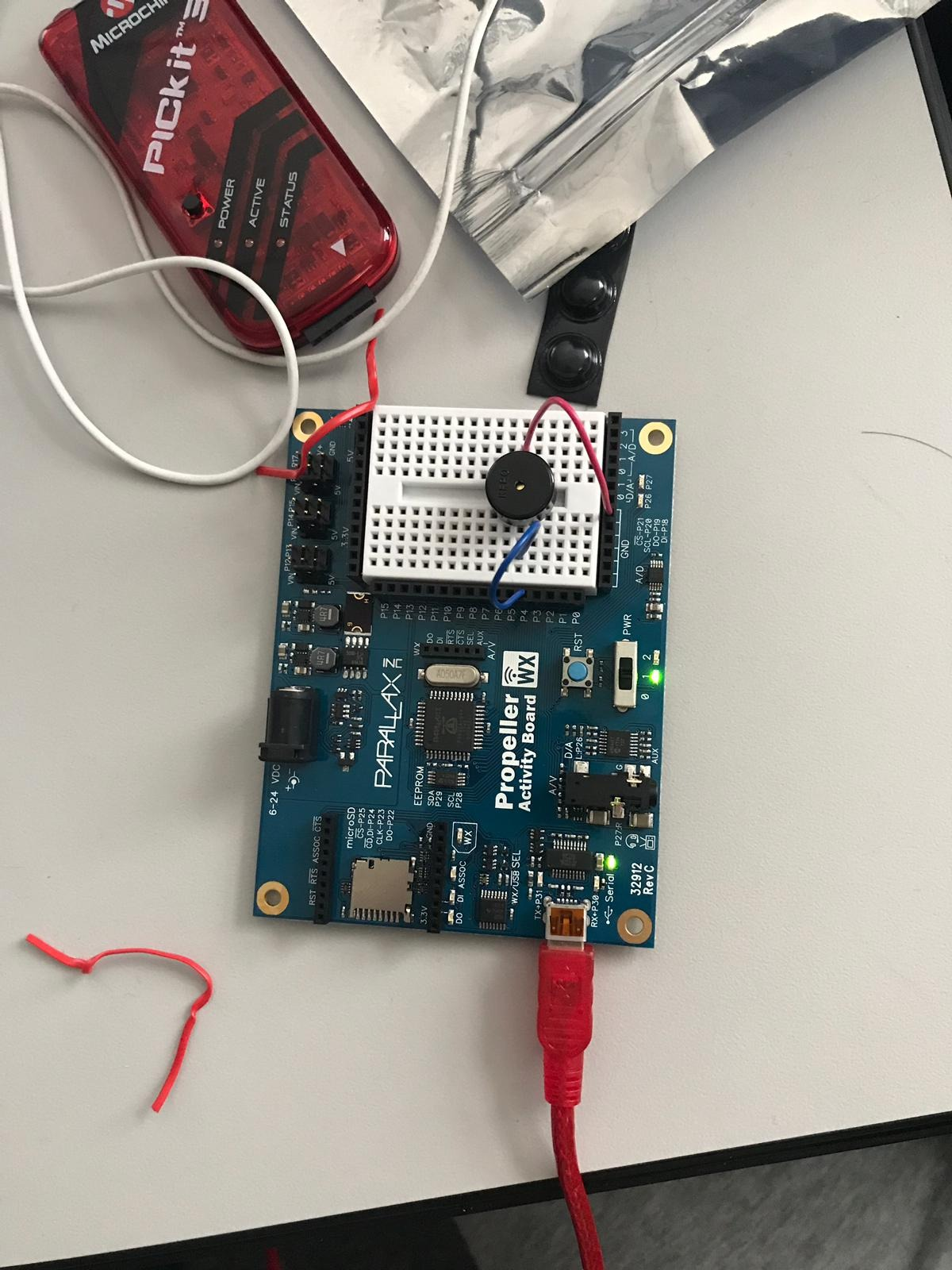
#### In this section of the lab, we ill be working analog to digital (A/D) and input/output (I/O). The purpose of such concepts is to monitor and sent outputs the board.

For this section we are going to use SimpleIDE to build to programs:

1. Piezo Beep: this will play the sound notes that are in the array note. The note is played because of the frequencies that are defined below.

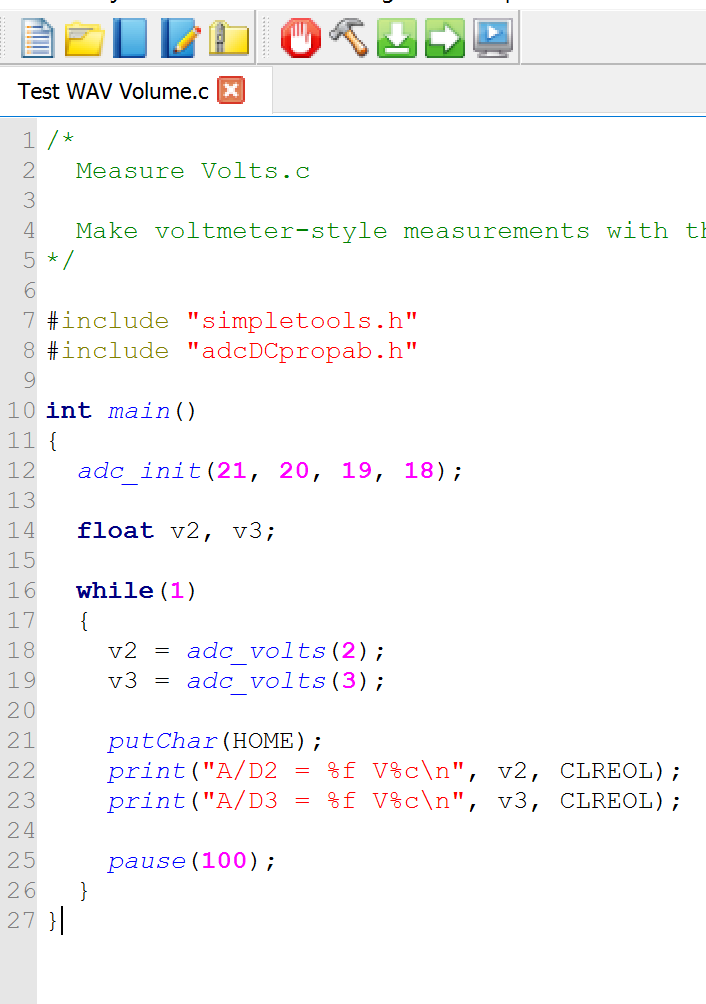
The code is:





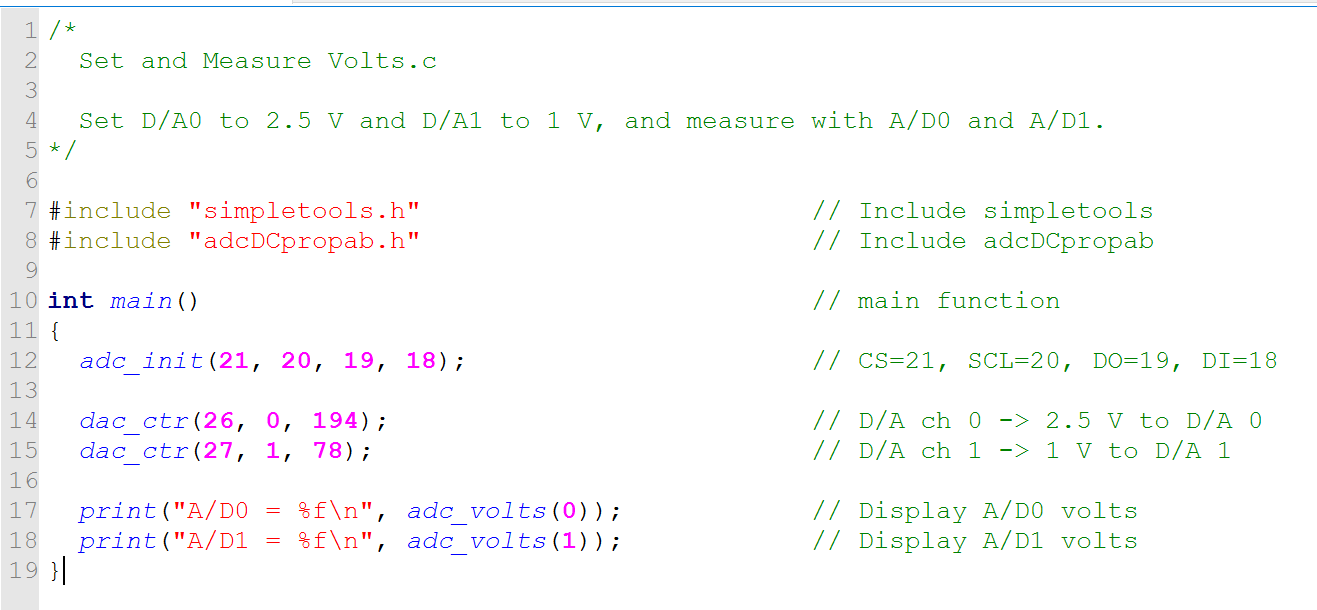
1. Measure Volts: This program will measure the voltages across provided circuits elements and show them on SimpleIDE terminal.

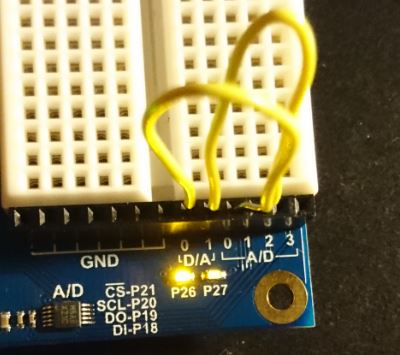




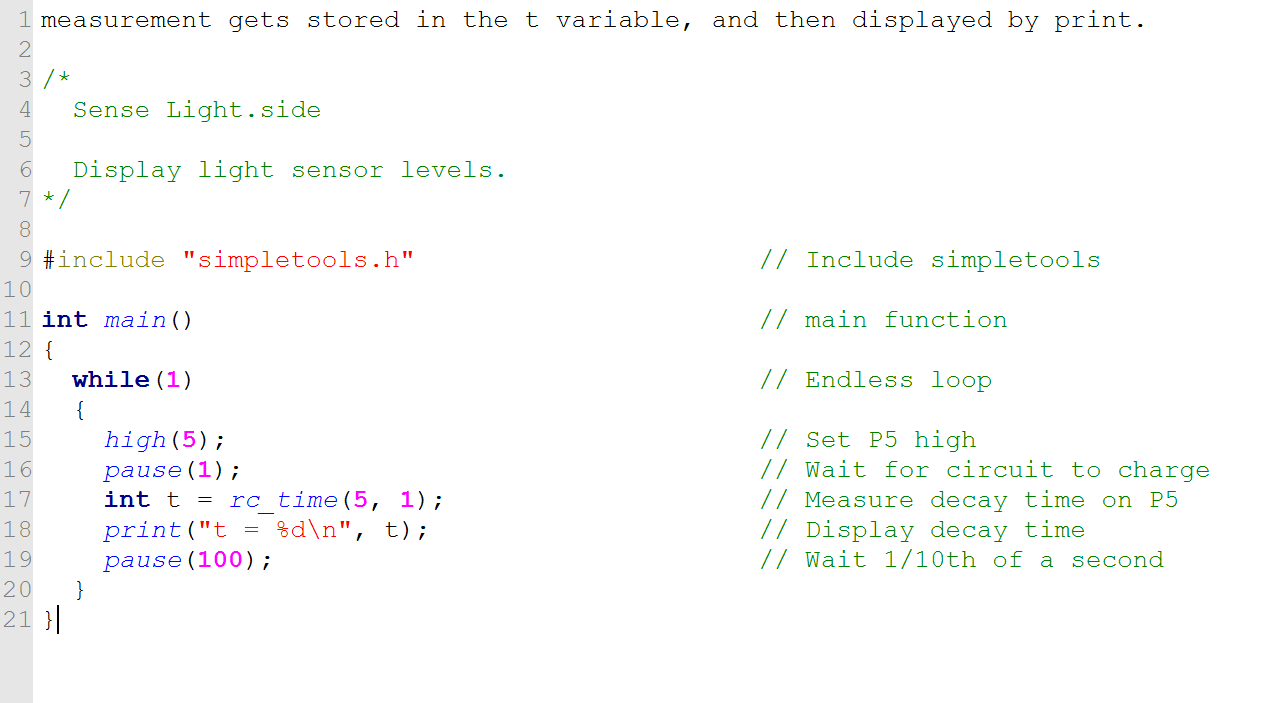
The sample panel will show voltages at A / D2, which should be null volts, and A / D3, which will differ as you press the button of the potentiometer. Here's an instance with the A / D2 properly reading GND as 0.00 V and evaluating the panel at about 1.62 V. That's about half way through the spectrum of movement of the panel and about quarter of the complete voltage spectrum of 0.0 to 3.3 V.

1. Sets Volts: In this one I used the board to set voltage level of the Propeller board by connecting D/A0 to A/D0 and D/A1 to A/D1. P26 send a 2.5 V signal to D/A0, and P27 send a 1 V signal to D/A1 so P26 LED will brighter than P27.



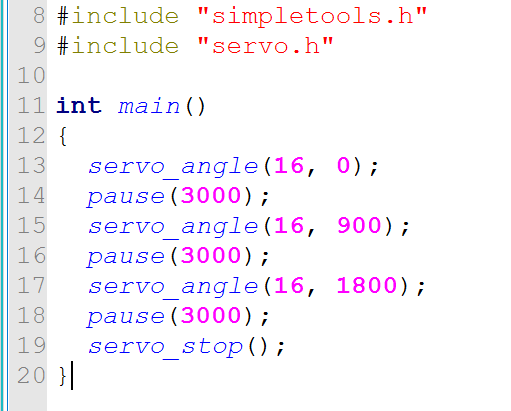


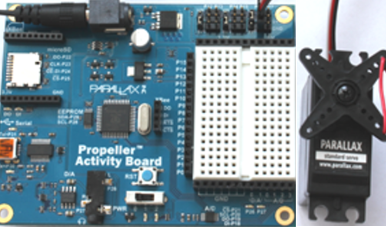
1. Sense Light: The Sense Light program will a number that equal to the light level the QT circuit detects.



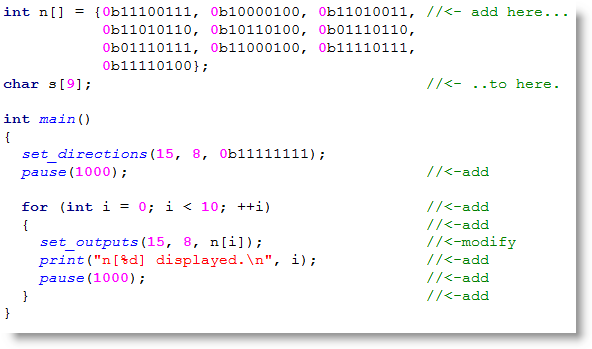
by increasing the levels of shade over the phototransistor, the darker it gets and the larger the measurement gets.

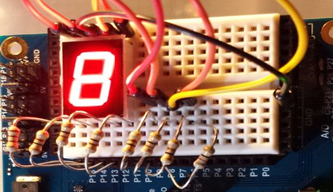
1. Standard Servo: In this part we sued the Parallax Standard Servo and the Propeller board to program application to control the Servo. A wire was attached to the Servo to identify the direction.





1. Seven Segment LED: This part we used a seven segment display and SimpleIDE to program countdown and display it on the seven segment. We can control the wait between each count by pausing at the end of each loop period.

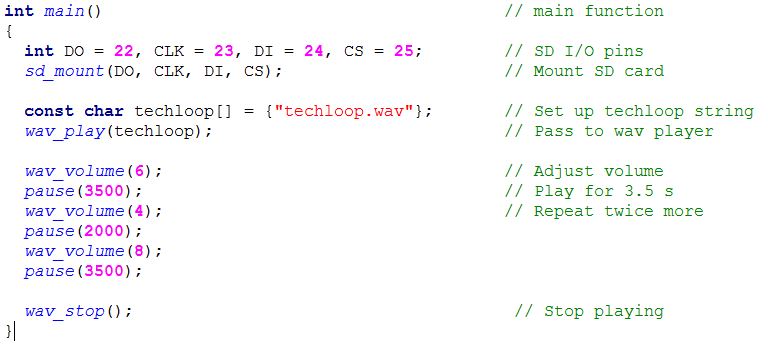




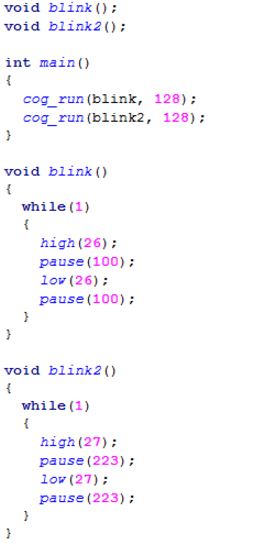
**Part 3: Multicore Approaches, Libraries, and Counter Modules**

In this section, we will dive deeper into the logic of the board and how some of it functionality actually work. We are going to learn about Propeller C libraries, Multicore Approaches, etc.

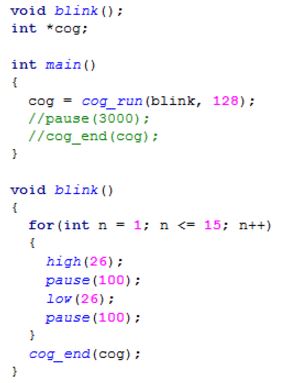
1. SD Card Data and Play WAV Files: For this program we used an SD card to download a dot wav file to play it on the board. After the download we run the following code and by using a headphone, we were able to hear the wav file that is on the SD card.

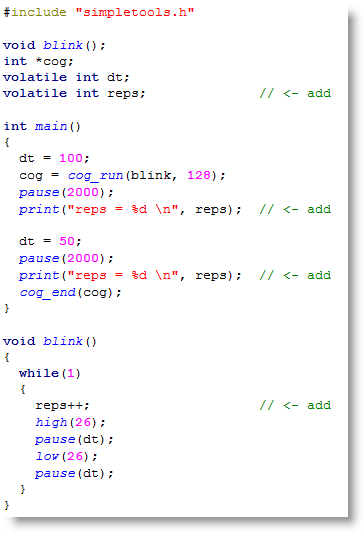
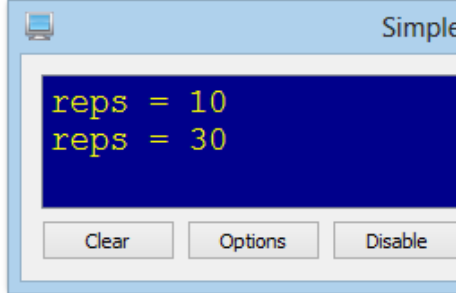


#### Multi-Core: After running the above application, I tried doing the Blink light in part one again. However, this time, I used Cog\_run(function name, 128) twice to tell the board I want to run those simultaneously.

in here we are calling the two functions together  
  
the first one will make the LED on port 26 run then pause foo 100 then turn of and pause for 100 infinitely.

the second one will well make the LED on port 27 run then pause foo 223 then turn of and pause for 233 infinitely.

1. STOP or cog\_end(cog) :The propeller has a function that can stop the running cores. the code below demonstrates how it works.  
     
   
2. Also, the board support sharing data between the cores. We can declare a global volatile variable that the main function can use to control a function. The main function sets the value of the variable, and then the all other functions can use global and use it in that specific function.

Example:  
   
  


**Part 4: Design Project**In this secion we are going to combine two of the projects we did in the previous parts to make an application that includes an input and output and multi-core application.  
  
**The program will prompt a question to the user asking to pick choice from one to three.  
 1- This option will run a single core function which is blink on port 26**

**2- This option will run a single core function which is wav**

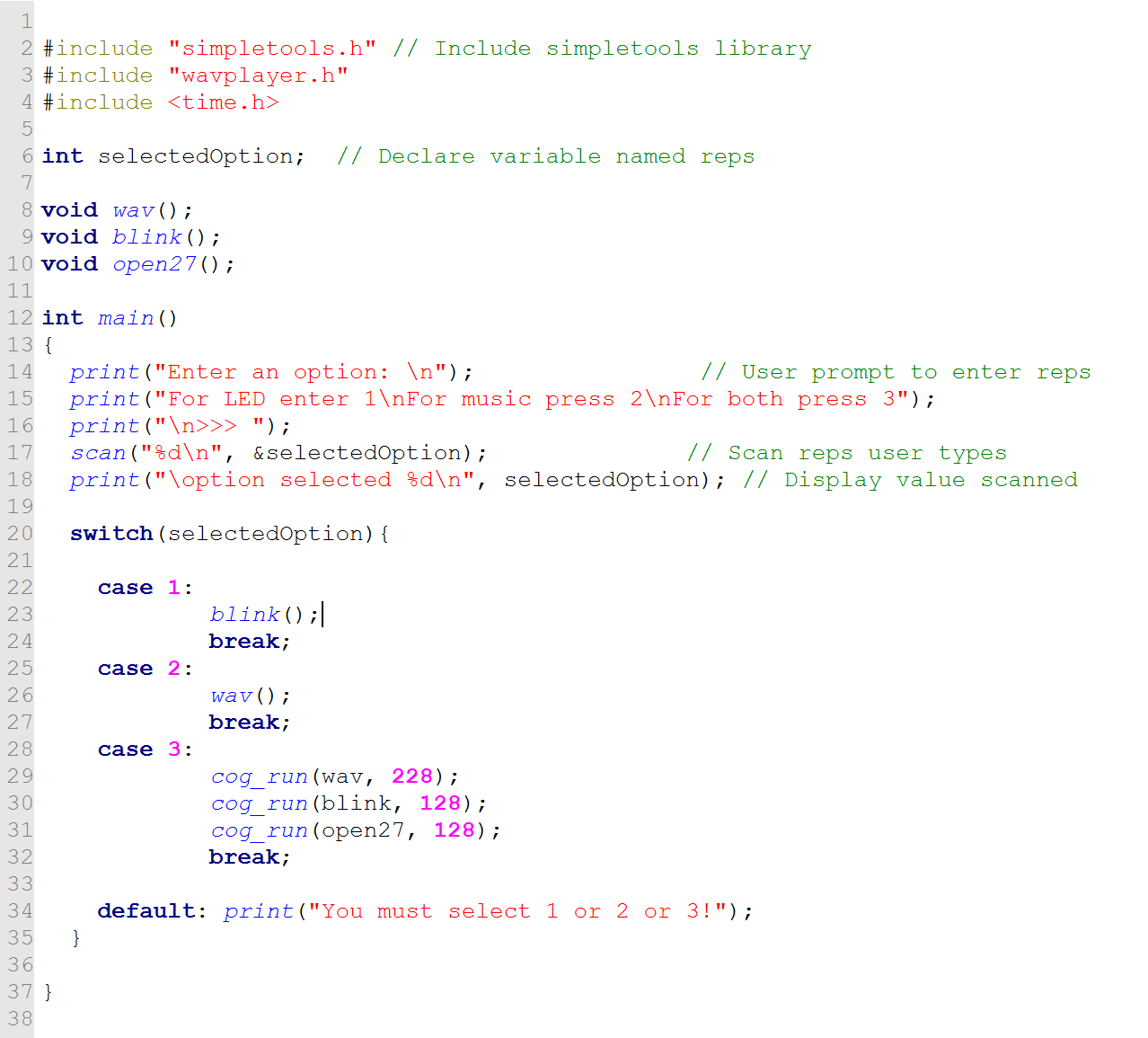
**3- This option will call three functions using cog\_run and using multi-core approach**

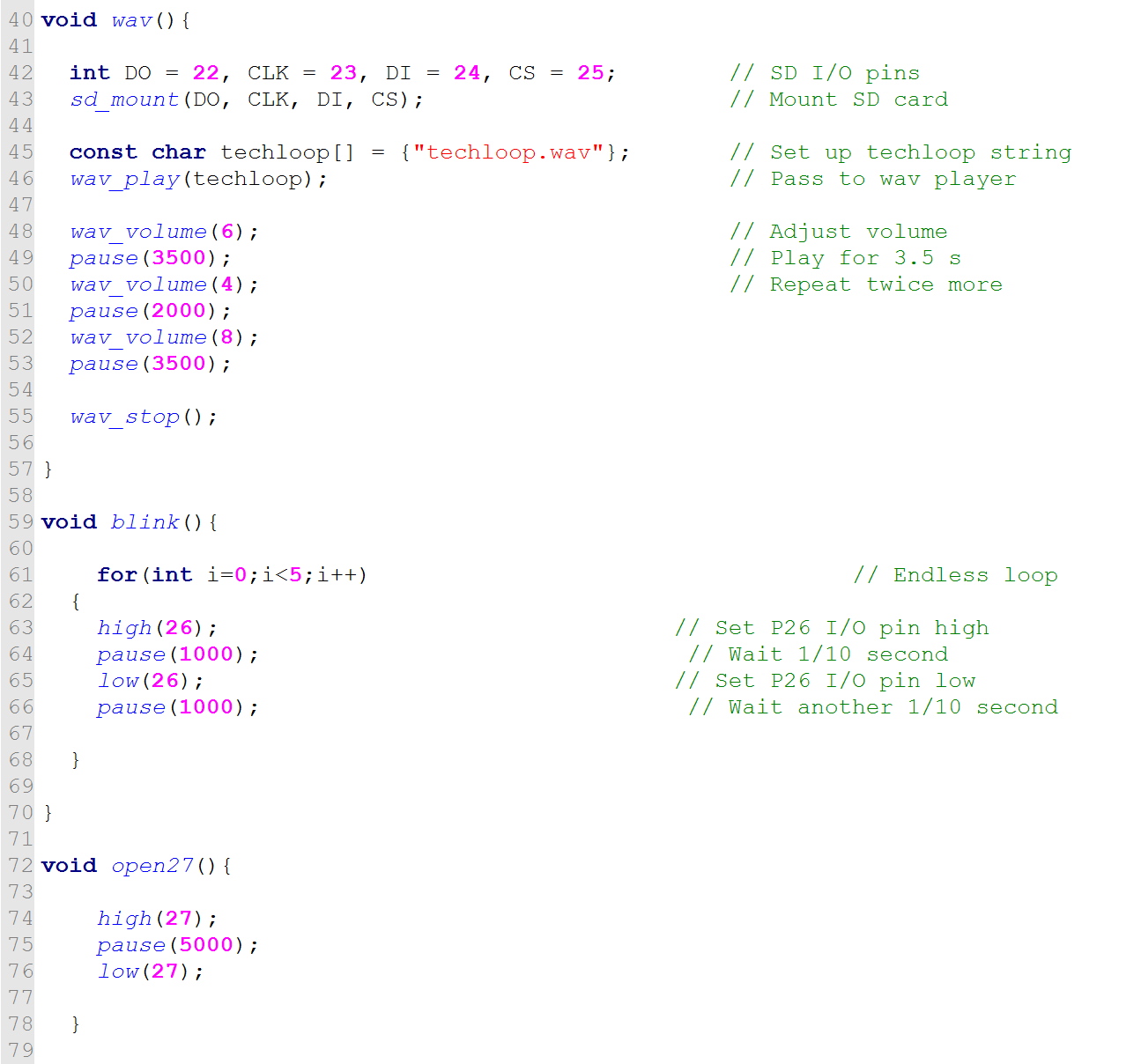
**To run blink on port 26 and wav sound and another blink on port 27.**

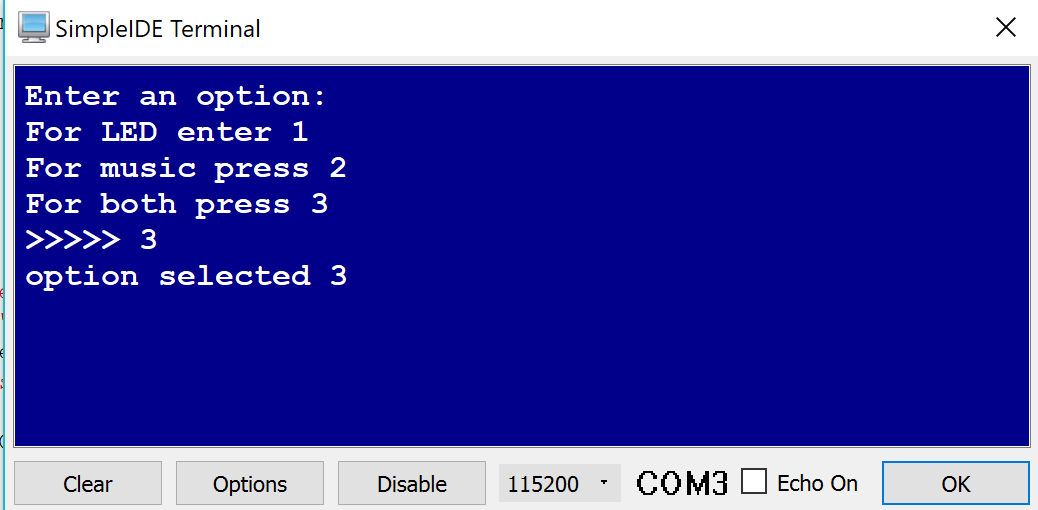
Circuits used are >>  
- 2 blink light functions using port 27 and 26,  
- Wav File with MicroSD Card,

-C Programing,

CODE:





**Output:  
**