**Address:**

To: Dr. Christopher Peters and Mr. David Cinciruk

From: Philip Mak and Sancheet Hoque

Date 4/19/2019

Re: Lab 2 - Week 1 (Prelab)

**Introduction:**

The purpose of the first week of Lab 2 was to familiarize students with Arduino programming along with software integration with MATLAB and lab equipment (DMM,AWG,Oscilloscope.) For the first part, 2A, students wrote an Arduino code that takes DC signal inputs and outputs a PWM signal, with a linearly increasing duty cycle. In the second part, 2B, students wrote an Arduino code that counts pulses from a signal. To finish the prelab in part 2C, students wrote an Arduino sketch alongside a MATLAB code that interact with each other. All of this work in the prelab would be later used in the Lab 2 - Week 2 (Lab.)

**Method/Analysis**

For part 2A, the students were required to have the Arduino output a PWM signal at 980.3Hz with a changing duty cycle which is linear to the DC input from 0 to 5V. In the Arduino, 5V is 1023 and 100% duty cycle is 255. In lines 4 and 6 of Figure 1, these were initialized and used in the loop. To relate the input to the duty cycle, a simple ratio was calculated: input voltage value divided by max voltage. This value was then multiple by the max duty cycle to use as the output duty cycle for the PWM. Serial Print was added for debugging.

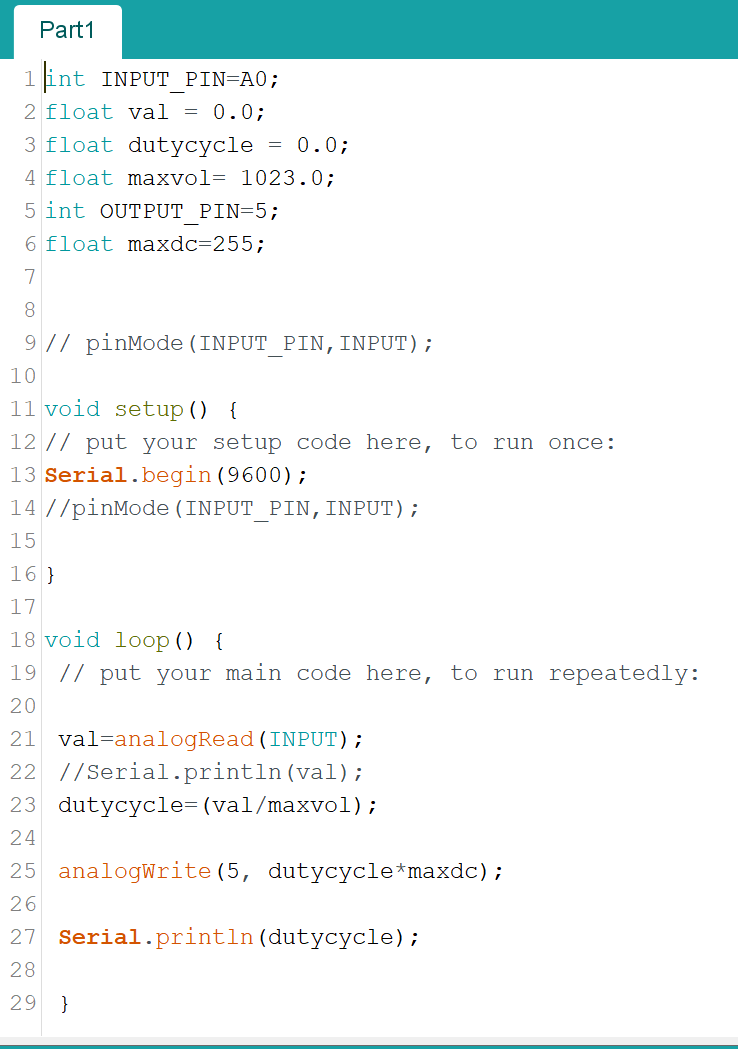


Figure I : Arduino sketch of part 2A - DC input to PWM output

For part 2B, the students had the Arduino count the number of pulses of a signal which is displayed on the serial com each second. Since this is no output signal, only a input pin was initialized. In the setup function, the serial and interrupt were implemented. The interrupt is triggered when there is a rising pulse and the function pulse state is run. Pulse state counts the number of rising edges in the current pulse. In the main loop, a 1000ms delay is placed to account for the counting only per 1 second. The serial print is also put right after in the main loop.



Figure II : Arduino sketch of part 2B - Pulse Counting

For part 2C, the students had to read the values from their Arduino code. The built in serial commands are used to simply read the output in the Com port. Fscanf was used to convert into a readable value.

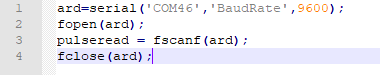


Figure III : Part 2C - MATLAB code interacting with the code from Part 2B

**Discussion and Conclusion**

The introduction of the microcontrollers into this project is important as the Arduino generates a PWM output signal. The Arduino has an odd “translation” of volts to bits, as 5V became 1023, and the sketch was different than the usual MATLAB code. Students had some trouble with the Arduino toolbox values mismatching and opted to use serial and fscanf() instead. Flushinput was used before the fscanf so the scan doesn’t read in multiple values but rather only the most recent one.