RSA Lab #3: Introduction to the Arduino

# Part 0: Virtual Reality

For this section of the lab, I simulated the blinking of an LED with a frequency of 1 Hz using the online software TinkerCad.

1. Calculate the value of the resistor assuming an input voltage of 5V, a forward voltage of 2.2V, and a desired current of 20Ma.

The voltage on the cathode should be equivalent to the difference between the input voltage and the voltage drop. Thus, we can use Ohm’s Law to find the resistance required for those specifications:

1. Open the Code Editor window and delete the example code. Write your own sketch that will blink the LED at 1Hz (i.e. one full on/off cycle takes 1s). Submit a screen shot of the simulation with the circuit and the code.

See attached screenshot. The only difference between this code and the one described in Part 2.4 of this lab report is the change in output pin number, from 7 (2.4) to 2 (this problem), representing a change in the pin to which the LED is connected.

A picture containing calendar

Description automatically generated

# Part 1: Hello World

For this section of the lab, I simply replicated the “Hello World” code provided by my instructors and verified its functionality in the Arduino IDE.

Text

Description automatically generated

(This is just a snippet of the results, the messages appeared in the serial monitor indefinitely).

# Part 2: Digital Output

For this section of the lab, I was able to construct a simple LED circuit connected to the Arduino UNO board in order to test the functionality of a PWM signal on the LED.

1. Write a program that uses the delay() function, the pinMode() function, and the digitalWrite() function, to generate a 1 Hz square wave output on digital pin 2 of the Arduino. Run it, and verify that it accomplishes this objective with the oscilloscope. Hand in a printout of your program.

See attached code. This was a relatively simple program to implement, as I just had to reverse the state of the output with half-a-second delays.

Graphical user interface, text, application

Description automatically generated

1. Annotate a scope plot of your 1Hz output. Be sure to annotate voltage level, peak-to-peak “amplitude”, and period of the signal.

See attached scope. I was able to verify the frequency and period of the waveform matched that was programed in the Arduino code.

1. Use your multimeter to measure the diode voltage drop of your LED

I measured the voltage difference across the two terminals of the LED with the DMM to be approximately:

1. Design a circuit that attaches an LED to the digital output pin 2 of the Arduino. Design your circuit to ensure that the diode current does not exceed the current capability of the Arduino and the diode – have LED draw about 30mA. Sketch your full circuit and show the design resistor value. Label all external connections where they connect to the Arduino.

See attached sketch. I was able to find the necessary resistor values by deriving the quotient from the measured voltage and the required current of 0.03 A.

# Part 3: Using Variables

For this section of the lab, I was able to make a program that compares the incrementing speed between integer-type variables and float-type variables in the Arduino IDE.

1. Write a program that runs a race between integer and floating-point variables in a simple calculation. Be sure to place adequate comments in your program. Hand in a copy of your commented program and a screen shot of the Serial Monitor output.

See attached program and serial monitor.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

1. How many bits comprise a variable of type int in this development environment? How about an unsigned long variable?

In this development environment an integer variable is stored as a 16-bit value. For an unsigned long variable, this number doubles to 32 bits.

1. How many bits comprise a variable of type float in this development environment?

This development stores a float type variable as a 32-bit value.

1. Compare the speed of incrementing an integer variable versus incrementing a floating-point variable.

As seen in the attached serial monitor screenshot, the int-type counter produced many iterations of count while the float-type counter only produced a few. This means the integer variable is faster when incrementing compared to a floating-point variable.

# Part 4: Basic Pulse-Width Modulation:

In this section I made a program to further simulate the PWM signal fed into the LED by varying the time when the LED is high while maintaining its frequency, thus decreasing the amount of time it’s low. This was done both manually and with a random number generator, which ranges from 0% high to 100% high.

1. Write a program that powers the external LED from step 2) above at random levels of intensity. Do this by generating a square-wave signal with a fixed frequency, but with a varying amount of time on versus off. This technique is called Pulse Width Modulation, since you are changing the width of the “on” pulses in a square wave with a constant frequency.

See attached program. I was able to construct this by initializing variables to control the percentage at which the LED is high/low. The millis() function was used to decrease wait-time, instead of using the delay() function.

Graphical user interface, text, application, email

Description automatically generated

1. Run your program with a duty cycle of 33%. Use your oscilloscope to measure the frequency and period of the square wave. Print your waveform and annotate the plot.

See attached scope plot. The LED’s time in the HIGH state corresponding to the LOW state verifies the 33% duty cycle that was expected from the program.

1. Now vary the square wave frequency and re-upload your program. What is the minimum frequency that you can pulse the LED without seeing it flicker noticeably?

By constantly varying the period, I was able to just barely notice the LED rapidly blinking at a period of 20ms. Thus, the minimum frequency for visibly pulsing the LED would be the inverse of this value.

1. Now modify your program so that it chooses a random duty cycle, changing once each second. Hand in a copy of your commented program.

See attached code. This implementation wasn’t too difficult to implement, as I simply had to make the scale variable type “long” in order to use a random number generator to get a percentile value. This was moved into the main loop in order to repeat the process after each period, which remains fixed at 1000ms (1 second).

Graphical user interface, text, application, email

Description automatically generated

# Part 5: Batteries: Voltage, Current, and Resistance

1. Take a 9V battery with an open-circuit voltage between 7.5V and 9V. Measure and record its open-circuit voltage VOC.

I was able to use my DMM to measure the open-circuit voltage of this 9V battery.

1. An Alkaline battery’s internal resistance is usually in the range of 1Ω to 10Ω. Can your multimeter handle this? Which current range should you use?

Since the voltage of the battery is 7.25 V and the internal resistance is in the range of , the expected current should be in the range:

Since the maximum amperage of this battery should be 7.25 A, the DMM should be able to handle this when operating in the 10A setting.

1. Measure and record the maximum short-circuit current ISC. DO NOT leave the circuit connected for more than a second or two; leaving it connected will heat it up and could start a fire.

I was able to find this current using the “max” setting on the DMM and recording the first value that appeared on the screen.

1. Calculate the total resistance based on the measured voltage and current.

I was able to use the measured current and measured voltage of the battery to derive its internal resistance.

1. Measure the resistance of your multimeter leads with your multimeter by shorting the leads to each other.
2. Calculate the battery’s internal resistance by subtracting the lead resistance.

# Part 6: Question

This section was comprised of a single question.

1. If a weak 9V battery (VOC = 7.8V, Rbatt = 5Ω) is used to power an Arduino, how much current would the Arduino need to draw in order for the voltage supplying the Arduino (VIN) to drop to the minimum of 7V?

In order for the input voltage to drop to 7V from 7.8V, a voltage difference of 0.8V is required. We can use this, along with the internal resistance of the battery, to calculate the current from the Arduino.

# Part 7: Use an IR distance sensor to measure distance and display the distance in cm

The final section of the lab was dedicated to measuring the distance values predicted from the distance sensor through code written on the Arduino with the voltage given by the sensor. A linear model of distance versus voltage was used from the datasheet provided to calculate this.

1. Use a linear fit to the plot of V vs 1/d in the datasheet for the IR sensor. Actually, I would recommend an equation for 1/d vs V, as in (1/d) = slope \* V + intercept where d is the distance in cm.

This linear fit was derived from the inverted plot of voltage versus 1/d graph provided by the Sharp IR distance sensor datasheet.

A similar equation was also found by not inverting the initial plot, although both equations work for this experiment:

Thus, the slope of both graphs is .

1. Submit your commented code, with a screen shot of the Serial Plotter window.

See attached serial monitor and respective code. This was relatively simple to implement after deriving the linear fit for the relationship, although it was noted that the “voltage” read by the distance sensor had to be scaled by given that 0V corresponds to 0 for the sensor, yet 5V corresponds to 1023.

Graphical user interface, text

Description automatically generated

A picture containing graphical user interface

Description automatically generated