ECE 303-Lab Package-Fall 2020

Dr. Christopher Peters

October 2, 2020

Contents

List of Figures

Listings

1 Week 1-Introduction to Arduino

This week's lab is to introduce students to the basics of the Arduino. While many students already have experience with the Arduino set of microcontrollers, many have not. This lab puts students at least to a basic level of understanding and operating an Arduino.

1.1 Goals of this Lab

The main goals of this lab is for students to learn:

- Basic sketch construct
- Digital writing to a pin
- Analog reading of a pin
- Analog writing of a pin
- Delay operation
- LED operation

1.2 Preliminary Steps

- Install the Arduino IDE software - you can find this at Arduino IDE Website.
- Set up the Arduino - you can find the instruction at Getting Started with Arduino MEGA2560 Website
- Read the following Wikipedia site on pulse width modulation Wikipedia site

1.3 Parts Required

- Breadboard
- Arduino with power source (usually wall socket source)
- Resistors (you will have to figure out which ones are important-you need to find the maximum current each pin can handle)
- An LED - doesn't matter which color

1.4 Experiment 1-Digital Pin Operation

In this example, you will be turning the LED on and off via the digital pin. There will be a 1 second delay between the on and off operations. There are two parts: one hardware and the other software.

1.4.1 Hardware Part

In this part, you need to build the simple circuit. Figure 1 shows the basic circuit setup. Follow these steps:

1. Connect digital pin 13 to the blue rail.
2. Connect a 330 Ω resistor, with one end to the blue rail, and the other end to one of the holes in the main part of the board.
3. Connect one end of the LED to one other end of the resistor.
4. Connect the proper end of the LED to the resistor end, and the other to another row NOT on the same line as the resistor.
5. Connect the downsrteam end of the LED to the ground (GND) terminal of the Arduino.

That should be it!

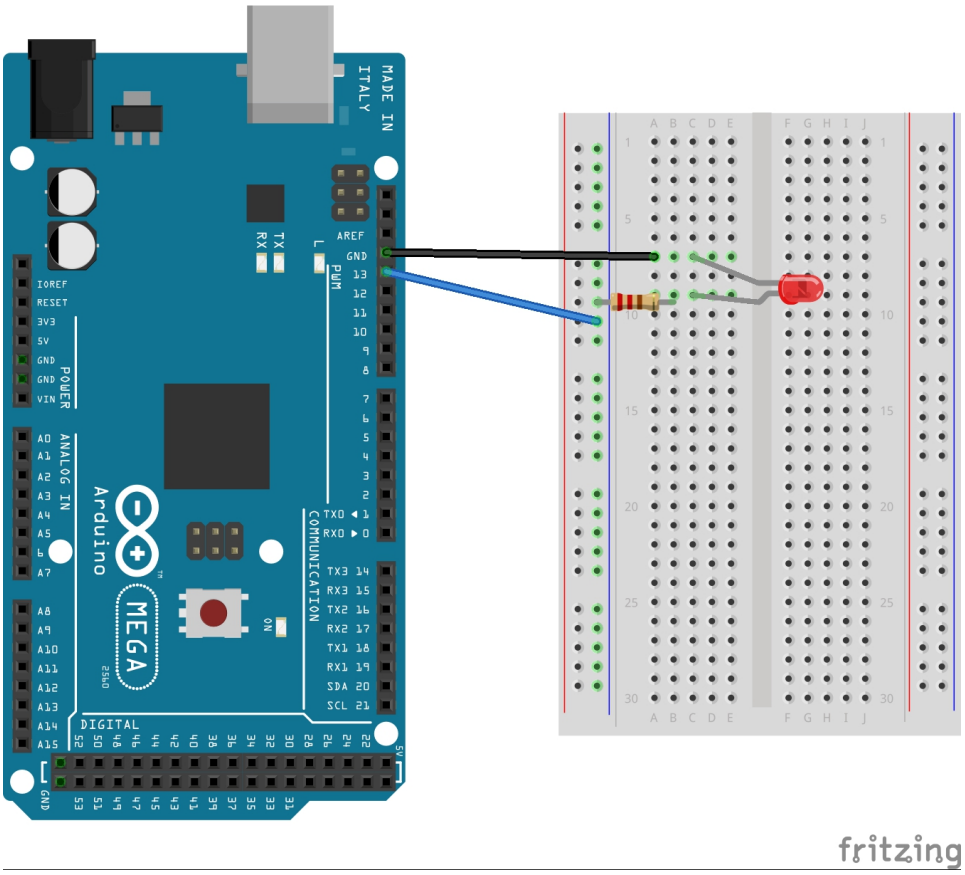

Figure 1: Basic LED circuit setup

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Listing 1: Basic Arduino blink code

1.4.2 Software Part

Now, you'll be working with entering the code, and then uploading it to the Arduino. First connect the Arduino to the laptop. Listing 1 is the basic code you need to input into the Arduino IDE and compile/upload to the Arduino. After you have entered the code, save it and upload it to the Arduino. The LED should start blinking.

1.5 Experiment 2-Analog Pin Operation

In this experiment, you will be keeping the circuit configuration the same as in Experiment 1. However, instead of using digital, you will be using the **analogWrite()** command to change the intensity of the LED. The program must have the following features:

1. Take user input from the Serial Monitor, accepting integer values from 0 to 255
2. Change the LED intensity by using the **analogWrite** command and the integer value entered
3. The LED intensity must not change until the user enters another integer

You will also need to use the **Serial.begin()** command, setting it to 9600 baud. Additionally, it might be beneficial to use the **Serial.available()** command.

2 Week 2-Timers and Interrupts Project-Codebreaker Project

2.1 Introduction

Imagine you are trying to crack a safe with some 4-digit combination. You have a limited number of attempts to find the code. As you enter a possible combination, you gain information about whether or not any of the numbers input match the safe's code. After you reach the maximum number of guesses, the safe will lock. This week, you will attempt to replicate this process using the Arduino Mega 2560. This lab provides experience in programming Arduino, understanding timers and interrupts, and using input from the serial monitor.

2.2 Additional Resources

- If not familiar with programming an Arduino, review this tutorial site, review the tutorial site. Start with the **Program Structure** part and continue until you are comfortable.
- Familiarize yourself with Arduino commands Web Site
- On attaching interrupts Web Site
- On timer interrupts Web Site
- on the Serial command Web Site
- How to write a technical memo Web Site

2.3 Functionality

The general functionality of the code breaking project should be as follows:

- The program should generate a random integer between 0000 and 9999.
- Each position of the generated number corresponds to an LED. When the program begins, each LED should be blinking. The LED blinking is controlled by a timer, and should initially be blinking slowly.
- The user enters a four-digit number. That number is broken down into its 4 integers and is compared to the respective number in the same position as the computer-generated number.
- At each iteration (for each positoin):
  - If the input number is correct, the corresponding LED should turn off.
  - If the input number is incorrect, the corresponding LED should speed up.
  - After 5 tries, if the user has failed to guess the code, all integers of the number which have not been guessed stay solid.

As a simple example, here is how the program should respond if the user has 3 attempts to guess the code. Assume between subsequent iterations, LED blink frequency will increase by 100 Hz for each time that position is guessed incorrectly. HIGH will indicate the LED is constantly on while LOW means it is constantly off.

- Generated Code (what you are trying to guess): 1000-

LED Behavior: Input Guess Attempt 0				
Integer Position	1	2	3	4
User Input Position Value	-	-	-	-
LED Blink Frequency (Hz)	100	100	100	100

- User Input (Attempt 1): 1234-

LED Behavior: Input Guess Attempt 0				
Integer Position	1	2	3	4
User Input Position Value	-	-	-	-
LED Blink Frequency (Hz)	LOW	200	200	200

- User Input (Attempt 2): 1709-

LED Behavior: Input Guess Attempt 0				
Integer Position	1	2	3	4
User Input Position Value	-	-	-	-
LED Blink Frequency (Hz)	LOW	300	LOW	300

- User Input (Attempt 3) (Final Attempt): 1500-

LED Behavior: Input Guess Attempt 0				
Integer Position	1	2	3	4
User Input Position Value	-	-	-	-
LED Blink Frequency (Hz)	LOW	HIGH	LOW	LOW

2.4 Breadboarded Circuit

Four LEDs are needed for each integer position. Each LED is connected in series with a resistor to limit the LED current (R=1 kΩ will work). Each LED is connected to a digital output pin corresponding to a different timer. Keep track of which digital pins are being used. Each timer is associated with 2 or 3 pins. You do not want multiple LEDs connected to the same timer.

2.5 Sketch Requirements

Below are the main components you need to include in your sketch. There are a number of ways of programming this; do whichever you find best.

- Method of generating an initial combination (the number you are trying to guess between 0000 and 9999)
- Read input from the Serial Monitor to be used in your program-remember to properly set your baud rate
- Four timers, one corresponding to each LED. You have both 8-bit and 16-bit timers available. We would like you to use at least one of each.
- Four interrupts (output compare)-You can use the output compare interrupt to toggle the LED status. By changing the OCR value of each timer, you can change the frequency of the signal entering the timer counter register. This will allow you to change the blink frequency of each LED.
- Method of comparing each position of the input value with the computer-generated number. This will be used to determine what change will be made to each LED.

2.6 Deliverables

Write a technical memo summarizing the lab. Given this lab is not having you collect data, to verify you were able to complete the lab, we ask that you submit the following as well:

- Append your Arduino sketch to the technical memo
- Show the TA your results in the lab, or make a video and upload it. You should input guesses in the serial monitor, show your LED response, and show what happens when you reach the maximum number of attempts.

3 Week 3-Pulse Width Modulation

3.1 "Light" Reading—hahaha–get it?

- Adafruit website on LEDs: https://learn.adafruit.com/all-about-leds
- Adafruit website on photocells: https://learn.adafruit.com/photocells

3.2 Motivation

Photoresistors are very simple yet useful pieces of electronic circuits. They are used in several equipment, such as cameras and slow RPM counters.

In this project, you will be analyzing a photoresistor by controlling the intensity of an LED. As the PWM signal increases the duty cycle, the LED should get brighter, up until a point. Additionally, the photocell should decrease in resistance as more light is incident on it.

3.3 Baseline Experiment Configuration

The baseline experiment is displayed below in Figure 2. The LED is a red LED in series with a 1 kΩ resistor(either one can be connected to ground). On a different circuit, the photocell will be connected to the 5V pin, and in series with a 10 kΩ resistor(resistor to ground).

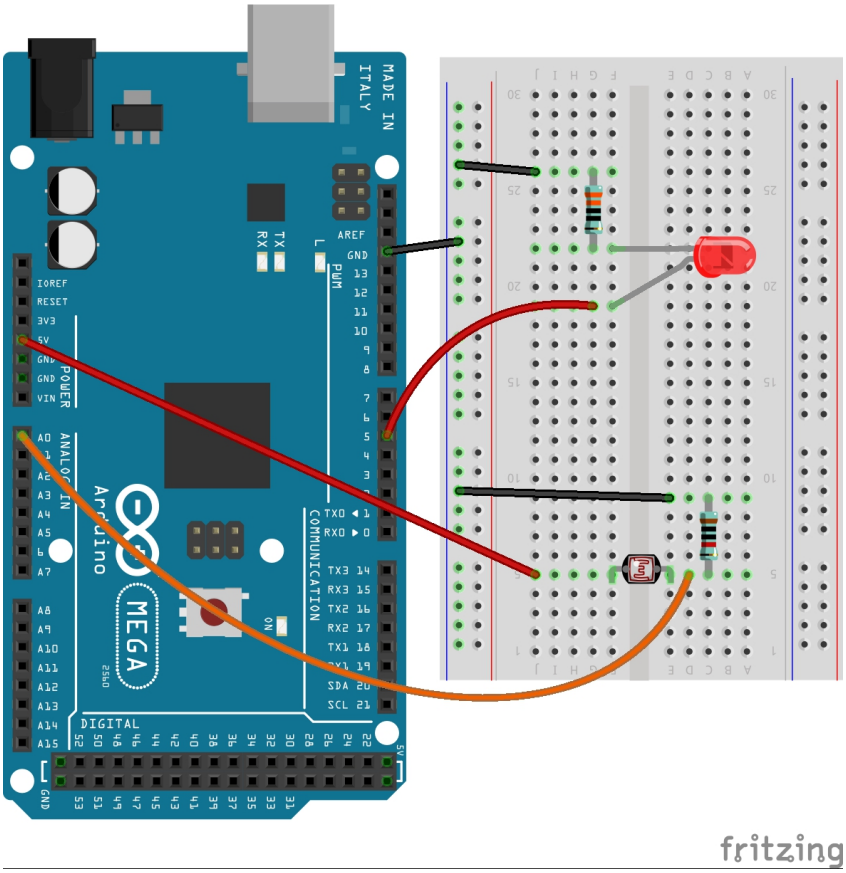

Figure 2: Base experiment

NOTE: Figure 2 is a generic setup. You will have to place your LED near the photocell head, and bend the photocell wiring such that the photocell head is in direct line with the LED. DO NOT wire the experiment with the distance in Figure 2. You will also need to monitor the voltage across the resistor in series with the LED.

NOTE: This experiment needs to be done in a dark environment so external light sources do not interfere.

3.4 Experiment 1-Varying LED Intensity

In this experiment, you will be varying the LED intensity by sweeping the LED input voltage. You can do this easily by using the **analogWrite()** command. However, DO NOT use this command. Instead, you need to implement PWM using timers.

Once the proper circuit is set up, you need to write some Arduino code to change the duty cycle from 5% to 100% in 5% increments. The time between readings should be at least two seconds to ensure all transients are minimized.

Remember, you are going to characterize the photocell behavior. This does NOT mean you are just going to look at photocell. You will also need to analyze the LED. Why? You might see there is some kind of a relationship between the LED circuit and the photocell circuit.

You need to output to the Serial Monitor the following information:

- Duty Cycle (%)
- Voltage across resistor in LED circuit (V)
- Voltage across the resistor in the photocell circuit (you may opt to measure the voltage across the photocell if you want-it can make calcuations easier).

If you are clever, you can copy and paste this information into Excel (or whatever spreadsheet program you use). From this data, you need to make the following calculations for each case:

- Effective source DC voltage powering the LED
- Current through LED circuit
- Current through photocell circuit
- Photocell resistance

3.5 Experiment 2- Various LED colors

This experiment is the exact same experiment as Experiment 1, only you need to repeat Experiment 1 for each color LED you have in your kit (DO NOT use the three-pronged LED).

3.6 Deliverables

In addition to delivering your code, you need to produce the following plots, with each **properly labelled** plot displaying all of the LED colors:

- LED circuit current versus duty cycle
- LED resistance versus duty cycle
- Photocell current versus duty cycle
- Photocell resistance versus duty cycle
- Photocell resistance versus LED current

3.7 Hints and Advice

Other common notes across all experiments:

- Conduct all experiments in a space that is as dark as possible, whether all lights in the room are out, or the system is contained in something where light can minimally penetrate.
- Save your data. It might make it easier for you to compare and contrast in your explanations.
- Power your Arduino using a wall socket to ensure your reference voltage for the ADC is 5V.

3.8 Questions to Ponder

1. Is it better to take voltage readings across the photocell or resistor?
2. Is it better to have the experiment be conducted in a totally dark room or in a container that light can't penetrate?
3. Should the photocell get the same voltage readings from different color LEDs? Why or why not?
4. Is it better to conduct averaging on the Arduino, or post-process it on a computer?
5. Is it better to save the data in memory and then transfer the data to the computer, or send the data every sample?
6. Would it be better or worse if each iteration is conducted faster?

4 Week 4-Serial Communications

In this experiment, you will be keeping your LED and photocell circuit intact, and implement automatic measuring and data analysis through the serial port. You can write you code in either MATLAB or Python. If you use MATLAB, DO NOT use the Arduino Toolkit. If you choose to use Python, DO NOT use any toolkit/library/package focused on the Arduino.

4.1 Requirements

- Increment the duty cycle from 1% to 100% in 1% increments
- For each iteration, take 5-10 points and perform averaging.
- Implement a very basic error correction algorithm
- Implement a simple method to reject a data point if it is not acceptable compared to the other measurements at the same duty cycle
- Write the measurements and calculations to a file
- Do this for each color LED in your kit
- Create the same plots as you did in Week 3. Make your properly labeled plots in either MATLAB or Python. This should be part of your code, not separate.

Attach both your Arduino and MATLAB/Python codes to the back of your memo.

4.2 Questions to Ponder

In your design, you should consider:

- Do you perform the averaging on the Arduino or the computer code? What is the tradeoff?
- How do you identify if a measurement is not similar to the other measurements of the same duty cycle?
- If you have to reject a sample, do you perform averaging on the smaller sample, or do you take another data point?

5 Week 5-Graphical User Interfaces

In this experiment, you will continue with the circuit you built and used in Weeks 4 an 5. You will now be extending this project by adding a graphical interface.

5.1 Project Requirements

You need to create a graphical interface, using either MATLAB or Python, which should have the following characteristics:

- A start button to start the process
- A text area to display the current duty cycle
- Plots of the following:
  - LED current versus duty cycle
  - LED voltage versus duty cycle
  - Photocell current versus duty cycle
  - Photocell voltage versus duty cycle
  - Photocell resistance versus duty cycle

NOTE: All operations from Week 4 must still be included.

5.2 Deliverables

- Technical memo in PDF form with plots and code attached in the back