RoboJackets Electrical/Firmware Training Week 1 Lab Guide

Stella Fournier, Devaughn Menezes

September 20, 2020
v1.0

Contents

1 Background

Prototyping with an Arduino is a great way to learn electronics. It is relatively easy to connect peripherals and program a controlled output. Its praiseworthy Arduino Software (IDE) is easy for beginners to use, but flexible to create some advanced programs as well. We can write programs to an Arduino by creating ‘sketches’. These have a .ino extension and can best be edited within the Arduino IDE. The IDE consists of an editor and a console which will serve as your primary tools to edit and check your program. You can view the full Arduino IDE Official Guide.

For this distance lab, we will be using the TinkerCAD simulator to create and run our lab experiments. You only require a browser and a stable internet connection to use the simulator properly. Additionally, the canvas/playground works using drag-and-drop functions. To program your Arduino or microcontroller, you can code using blocks or text, although, to improve your proficiency, we would recommend using text in the long run.

2 Objective

2.1 Setting Up AutoDesk Account

TinkerCAD is an AutoDesk product so it is useful to first create and AutoDesk education account first. For those who will do electrical training this is very useful as you will use the same account to install EAGLE.

1. Create an AutoDesk Education Account here using your Georgia Tech email
2. Log in to TinkerCAD with the same account here

2.2 Twinkle, Twinkle, LEDs

The objective of this lab experiment is to create a circuit using an Arduino that will cause LEDs to blink (turn on and off repeatedly). Take a minute to discuss with your partner(s) about a possible strategy to accomplish this. Then, refer to the instructions below.

2.2.1 Building and Running an existing program

Here, we will be using the blinking LED as an example program.

1. On the panel to the right, click the drop-down menu and find Arduino.
2. Drag-and-drop the ‘Blink’ circuit. What do you notice?
3. Click ‘Start Simulation’. What do you notice?
4. Click ‘Stop Simulation’. Then, click ‘Code’. Expand the panel if you wish. Then using the drop-down menu, switch to either ‘Blocks + Text’ or ‘Text’. Try to read the code.
   (a) Change the value of delay
   (b) Add more lines of code so that the delay is varied between each on-off cycle
   (c) Try to write code instead of using the blocks

2.2.2 Blinking an external LED on a breadboard

Now, try to repeat what you saw in the blink program from scratch. We recommend using a breadboard (you can search for this in the components menu).

1. Create a new circuit and place the following components on your canvas:
   - Arduino Uno R3
   - 220 Ohm resistor

- LED
- Breadboard

2. Setup your breadboard and components in a style similar to the image below:
   - Remember to check the polarity of your LED (longer side is +)
   - Connect a resistor in series with the LED to avoid accidentally blowing the LED
   - We recommend using different colored wires to differentiate between connections
3. Modify your code to blink the LED on the breadboard
   - Remember that blinking takes place as a result of alternating between on (`digitalWrite(pin, HIGH)`) and off (`digitalWrite(pin, LOW)`) states of the output pin
   - Include delays between each new state. This will allow for enough time to notice a change in the on/off state of the LED. You may use the `delay()` function which takes milliseconds as a parameter.

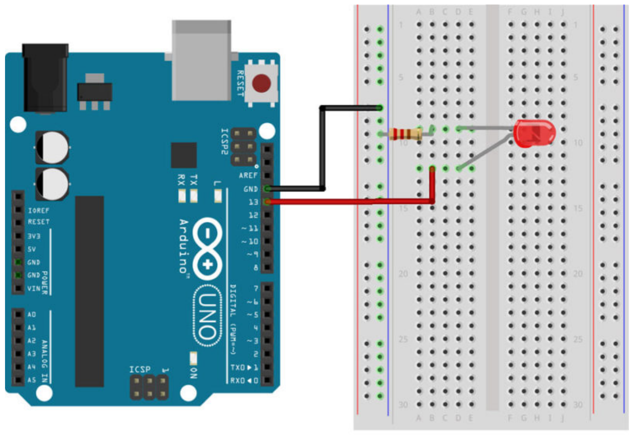


Figure 1: Arduino Uno connected to an LED on a breadboard

2.3 How Hot?

The objective of this lab is to use a temperature sensor and 5 LEDs controlled by an analog temperature sensor reading. The input level of temperature will light up a different number of LEDs.

2.3.1 Temperature Sensor Background Knowledge

- It basically measures the degree of hotness or coolness and converts it into a readable unit
- A temperature sensor has three terminals: Power (connect to the 3.3V / 5V), Vout (connect to one of the analog inputs), GND (connect to the ground pin). In TinkerCAD, if you hover over the pins, you can see which one is which on the part.

2.3.2 Reading an analog input

1. Prepare a new circuit canvas and place the following components on to the canvas:
2. Since this lab requires an analog reading, remember to use the analog-input pins on your Arduino . Likewise, remember to use the `analogRead()` function when programming your Arduino.

3. Before you write a program to light up your LEDs, it may be a good idea to use the `Serial.println()` function. This will create an output on the serial monitor (which can be accessed by clicking the 'Serial Monitor' button on the bottom right of your screen) allowing you to create proper divisions for values to light up each LED.

4. Will your LEDs light up as the temperature sensor readings increase linearly or exponentially? Experiment using different techniques and discuss with your teammates the pros and cons of each method.

5. Try to make your circuit as compact as possible. Use the least number of wires and try to make sure wires don't cross each other a lot (this will be a useful trait when creating PCBs)

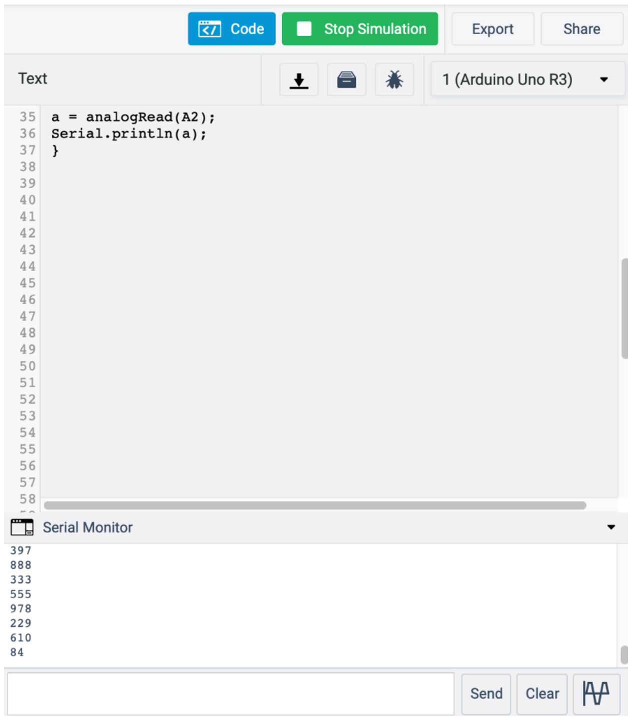
2.3.3 Using the `Serial.println` function and Serial Monitor

This is a very useful function when experimenting with serial analog inputs, but there is a specific way you need to set it up for this lab. Note that you can use Serial.print() as well, but doing so will have everything printed on one line instead of a new line.

1. Inside `void setup()`, write a function that sets up the data rate in bits per second (baud). It's okay if you don't understand what this means right now. Just know that in order to use the serial monitor, you have to establish the data rate for communication. The common rate is 9600 baud. You can set this by writing the function with the following parameter `Serial.begin(9600)`;

2. Inside `void loop()`, include `Serial.println(var)`; directly after your `analogRead()` function. A good idea might be to have the `analogRead()` function update a variable and then have the `Serial.println()` function print the value of the function.

3. To view your output, you will have to open the Serial Monitor. On the Arduino IDE, you would go to Tools → Serial Monitor. Since we are using TinkerCAD, you will have to open the Code panel and click Serial Monitor on the bottom. Note that you will not see any printed values until you Start Simulation.



```
a = analogRead(A2);
Serial.println(a);
}
```

```
397
888
333
555
978
229
610
84
```

Figure 2: Serial Monitor in TinkerCAD

3 Materials

- AutoDesk Education Account
- TinkerCAD for 2.1
- TinkerCAD for 2.2

4 Relevant Information

4.1 TinkerCAD Simulator

This online CAD works by using drag-and-drop features. You can search for components by using the search bar on the panel to the right. Once you place down a component, you can rotate or delete it using the buttons on the top left of your screen (or click on backspace on your keyboard). To connect components to each other, you can use a breadboard or hover and click from one terminal to another to connect a wire.

If you use a component which requires some code, like an Arduino, you can program the component by clicking the Code button on the top right. After you finish building and programming your circuit, click Start Simulation. You may have an option to view the Serial Monitor/Console.

Once you succeed with your build, you can share it with team members for review. Alternatively, you can leave comments on other people's or your own circuit by clicking the Annotation button on the top left.

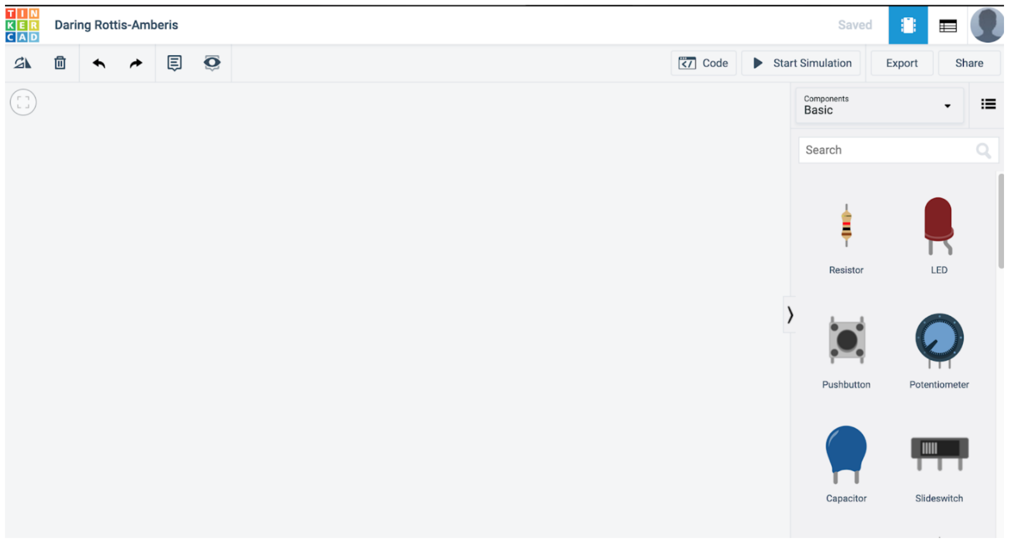


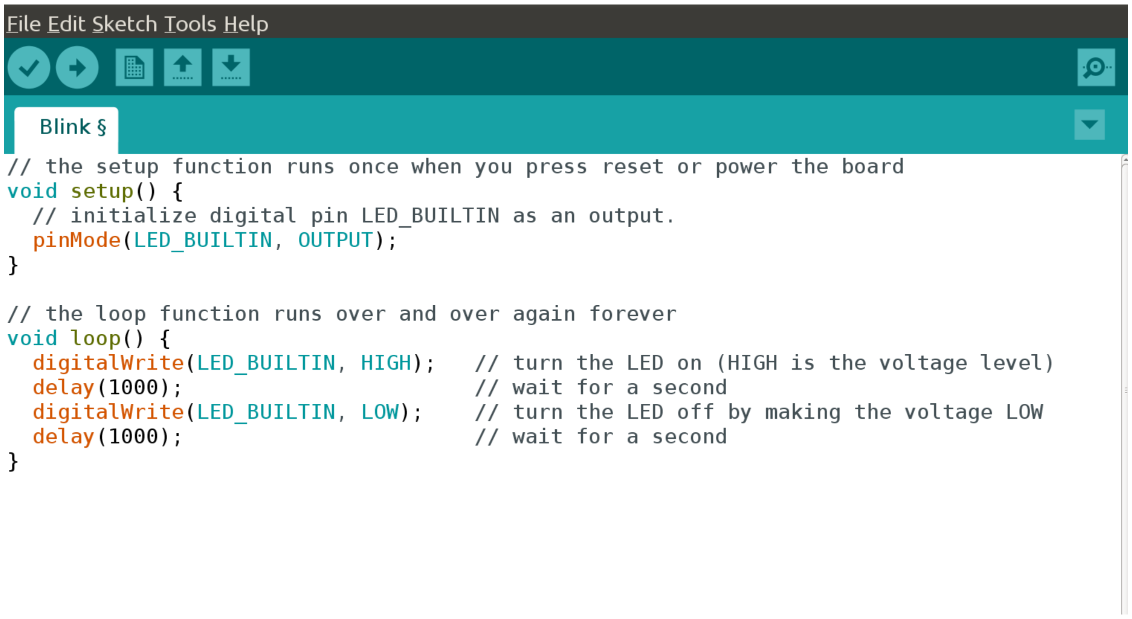
Figure 3: Creating a new circuit in TinkerCAD

4.2 Arduino IDE

The full official guide for the Arduino can be found here

- The checkmark is the “Verify” button which will compile your code
  - Compiling your code means translating your written code from a specific language to a machine executable code
- The arrow pointing to the right is the “Upload” button which will compile and load your code on your Arduino

- Before you compile make sure to select the port on your computer where your Arduino is connected by going to Tools → Port



```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Figure 4: Arduino IDE with a Blink program

The screenshot above is from the Arduino IDE – the application in which you will write your code and upload the code to your Arduino. The program will blink an on-built LED on the Arduino.

A typical program is split into two parts: the `setup()` function and the `loop()` function:

- `setup()` This function is called when the program starts. It'll run when you power on or reset the Arduino board. Here is where you will initialize variables, set pin modes, etc.
- `loop()` This function loops consecutively so the code you place here will constantly run

Some common functions you should know are outlined below:

- `pinMode(pin, mode)`: Configures the specified pin to behave either as an input or an output
  - `pin` is the number which is located next to the pin on the board
  - `mode` determines whether the pin serves as an INPUT or OUTPUT
- `digitalWrite(pin, value)`: Write a high or a low value to a digital pin
  - `pin` references the pin number
  - `value` determines whether the output should be given a HIGH or LOW value
- `digitalRead(pin)`: Read a value (high or low) from a specified pin
  - `pin` references the pin number
- `delay(time)`: Pauses the program for a specified amount of time
  - `time` takes a numerical value to represent milliseconds

Some more advanced functions include:

- `analogWrite(pin, value)`: Write an analog value to a digital pin with PWM (pulse width modulation)

  - `pin` references the pin number
  - `value` determines the output voltage (often based on AREF)

- `analogRead(pin)`: Read an analog value from a specified pin
  - `pin` references the pin number

5 Troubleshooting

5.1 Solutions

We have included the solutions below if you do not complete the lab during the session or if you want to verify your answer. If you need help during the lab ask an instructor!

- TinkerCAD Solution for 2.1
- TinkerCAD Solution for 2.2