RoboJackets Electrical/Firmware Training Week 1 Lab Guide

Andrew Rocco

September 28, 2021
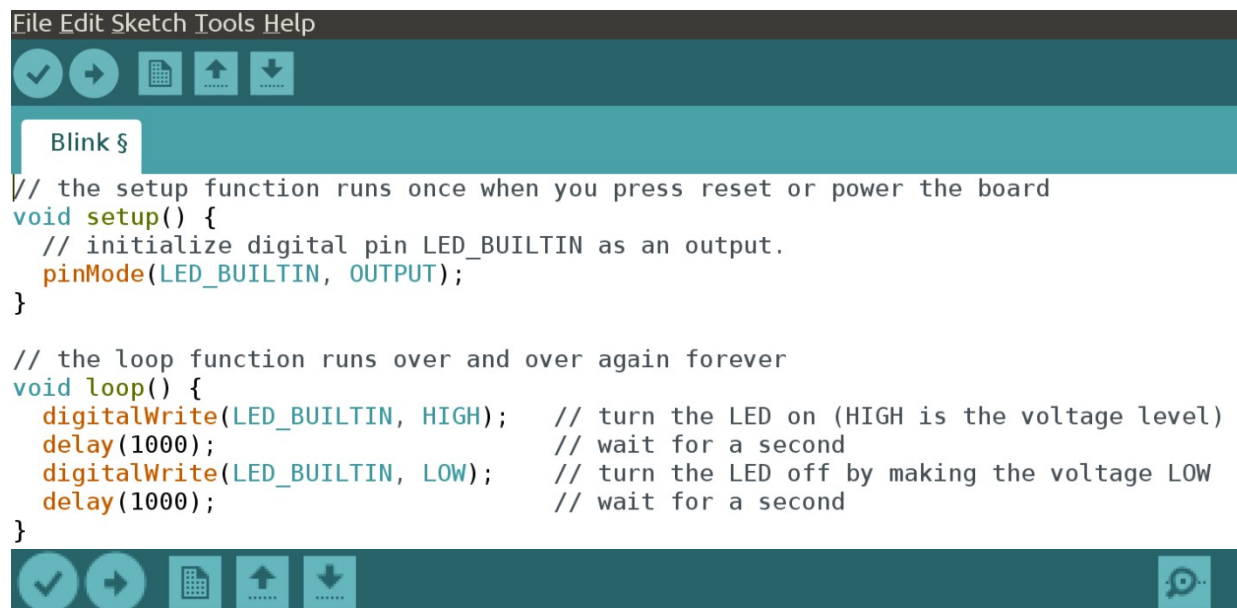v1.0

Contents

1 Background

We can write programs to the Arduino by creating 'sketches' which are written in the C++ programming language. These have a .ino extension and can best be edited within the Arduino IDE. The IDE consists of an editor and console which will serve as your primary tools to edit and check your program.

You can view the full Arduino IDE Official Guide here. You can download Arduino IDE here. Also, for the Arduino Nano, you may need to install a serial port driver to be able to communicate with it. You can find that driver here. If your code is still failing to upload, go to Tools and select the following: Board - Arduino Nano, Processor - ATmega328p (Old Bootloader), Programmer - AVRISP mkII.

```
File Edit Sketch Tools Help

  Blink §

// 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
}
```

- The checkmark is the "Verify" button which will compile your 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 and making sure the correct port is selected

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 once 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. You can have other functions in your program but they will not run unless they are called in setup() or loop().

Some common built-in 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

2 Objective

2.1 Uploading and compiling an existing program

- Here, we will be using the blinking LED example program that Arduino provides. This will blink an LED that exists on the Arduino itself. This is the same code that is pictured above:
  - Accessing program: Files → Examples → Basics → Blink
  - Upload the program

2.2 Blinking an external LED on a breadboard

- Setup your breadboard in a similar style to the picture shown below (note that we are using an Arduino Nano which is directly connected to the breadboard and has a different pin configuration):
  - 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
- Modify your code to blink the LED on the breadboard
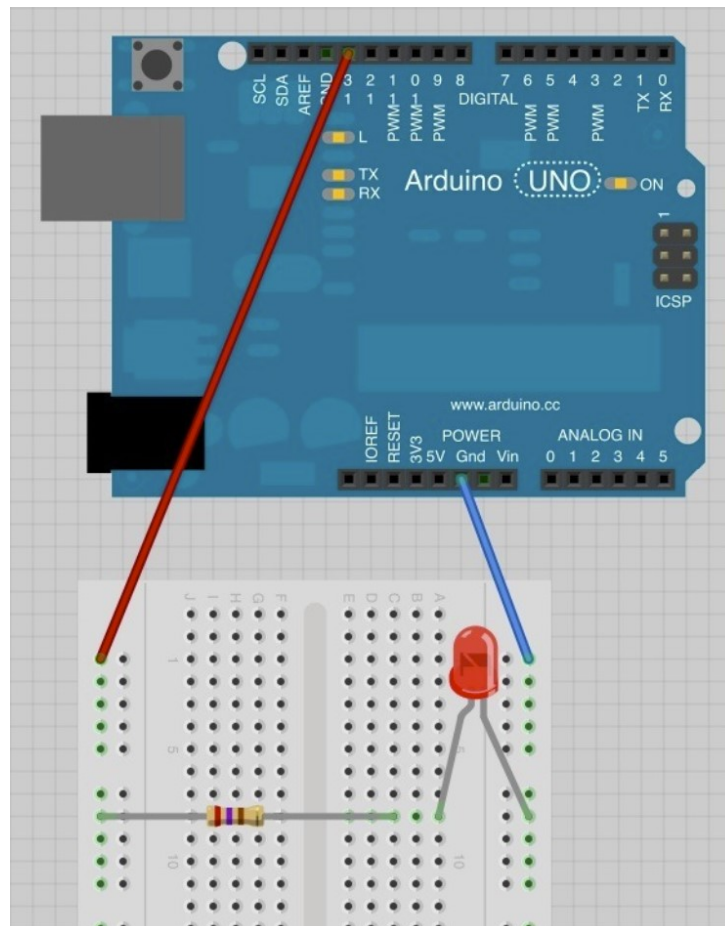
2.3   Something simple

- Use the provided button as a digital input to the Arduino to control the LED

  - We'll utilize the internal pull-up feature of the Arduino Nano here (and most other MCUs), although an external one would work just fine. Check the Relevant Information section below for more information.

- Create a circuit and write code to control 2 LEDs independently using the digital input of one button

  - For example, when the button in pressed, one LED turns on and the other turns off (v.v.)

2.4   Something challenging (Optional)

- Use the button to select between different blink frequencies for the LED

  - No button press results in a slower frequency blink, a button press results in a higher frequency blink

2.5   Something hardcore (Optional)

- Create a binary counter using the button as an input and the LEDs as an output

  - A button press increments the count by 1
  - Things to consider:
    * Binary numbers and bit-masking
    * Debouncing (see Relevant Information below)

3   Materials

- Ardunio Nano

- Breadboard

- Jumper wires

- Resistors

- Pushbuttons

- LEDs

# 4 Relevant Information

4.1 Pull-up Resistors

Pull-up resistors (and pull-down resistors) are used to ensure that a pin is not left floating when a connection is opened (by a button in this case). We connect the output side of the circuit to power through a resistor to accomplish this. Check out the circuit below.
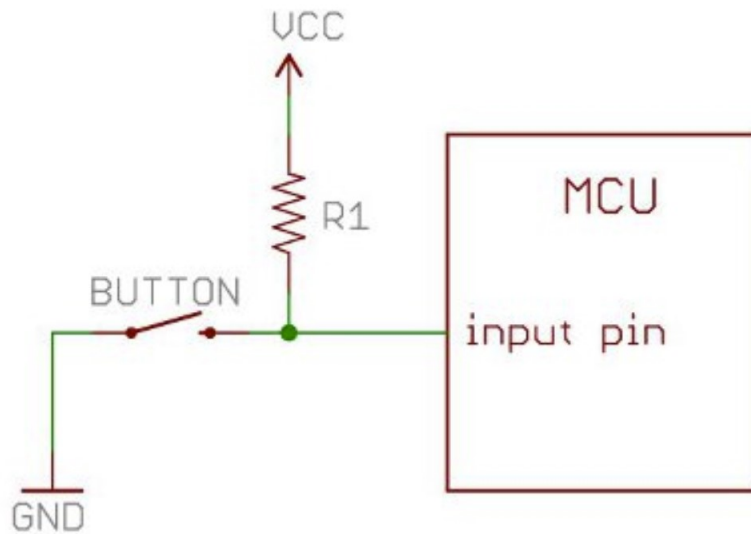
Luckily, the Ardunio Nano has built-in pull-up resistors for all it's digital pins, but we must activate them in firmware. To do this, we simply change the pinMode of the pin. Instead of using **pinMode(num, INPUT)**, we use **pinMode(num, INPUT_PULLUP)**. You can read more here.

4.2 Debouncing

Due to mechanical/electrical issues, when a button is pressed it may be detected as many button presses.
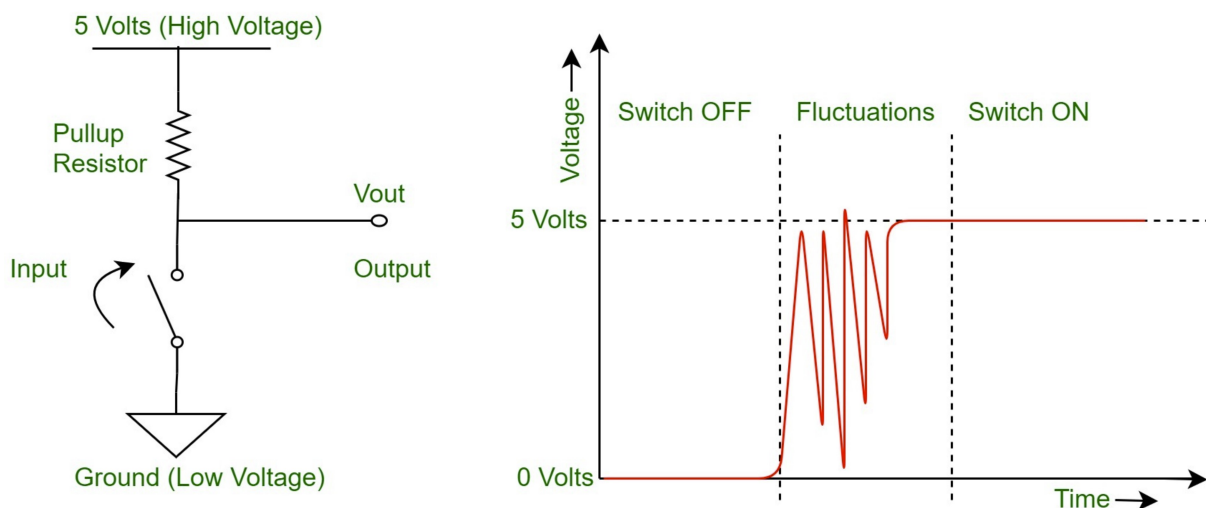
These fluctuations might be detected by the MCU as many button presses, causing issues in firmware. To counter this issue, there a few things we can do. For now, we can simply add a delay after we detect the

change of state. However you decide to detect a button press to increment the binary counter, you can add a **delay(50)** after so that the MCU doesn't even look for another change of state until the delay is over. 50 ms is more than enough time to counter this issue.