

# Embedded System Workshops

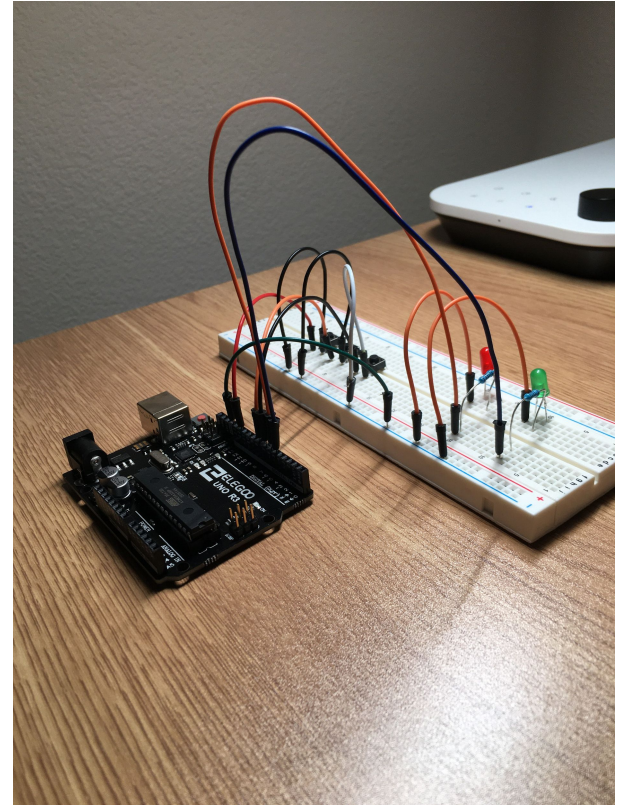
---

02. Digital Inputs: Buttons  
*CCA Girls Who Code*



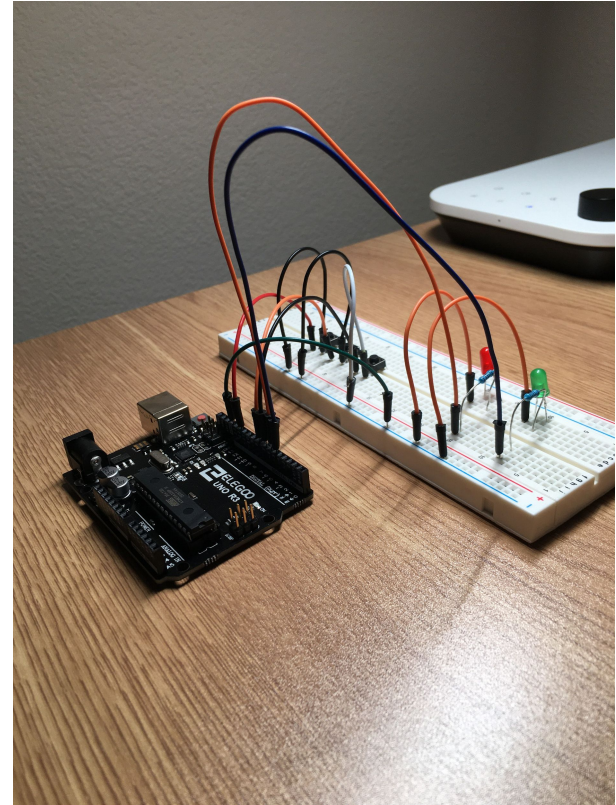
# Project Overview

- Purpose
  - ◆ Introduce digital inputs such as buttons
  - ◆ Use such inputs in a circuit
- Projects
  - ◆ Simple circuit with button
  - ◆ Button passcode project
- Grab your kit, and let's get started!



# What are we making?

- Button passcode project
  - ◆ Use a series of buttons to input a “password”
  - ◆ If the right password is input, a green LED turns on for a few seconds
  - ◆ If the wrong password is input, a red LED turns on for a few seconds
  - ◆ Can try again if needed



# Parts List

Below is the list of parts we'll be using during this lesson

- Arduino UNO R3 Controller Board
- USB Cable
- Breadboard
- 2 LEDs
- 2 220 $\Omega$  Resistors
- Male-male jumper wires
- 3 Buttons

# What are Switches?

A switch is a device that you can use to open or close a circuit at will. A closed circuit allows current to flow through it. An open circuit has a gap in the circuit, preventing current from flowing through it.

## Why is this relevant?

A button is a type of switch that closes the circuit when pressed down and opens it when released.

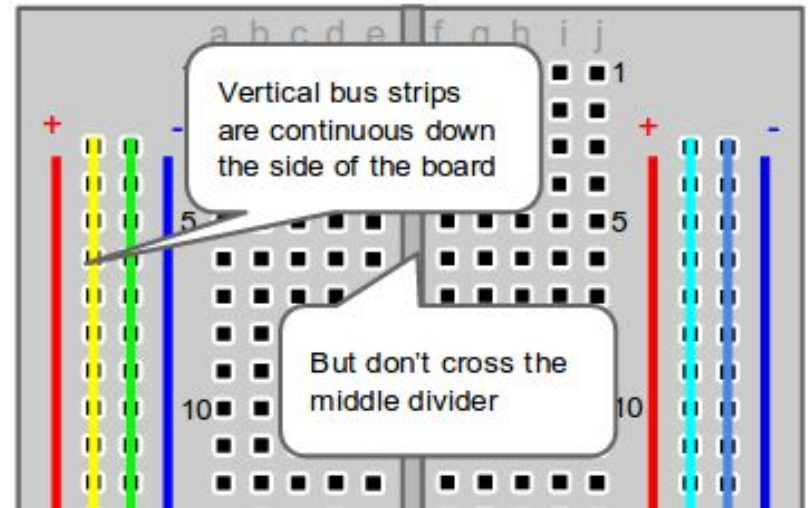
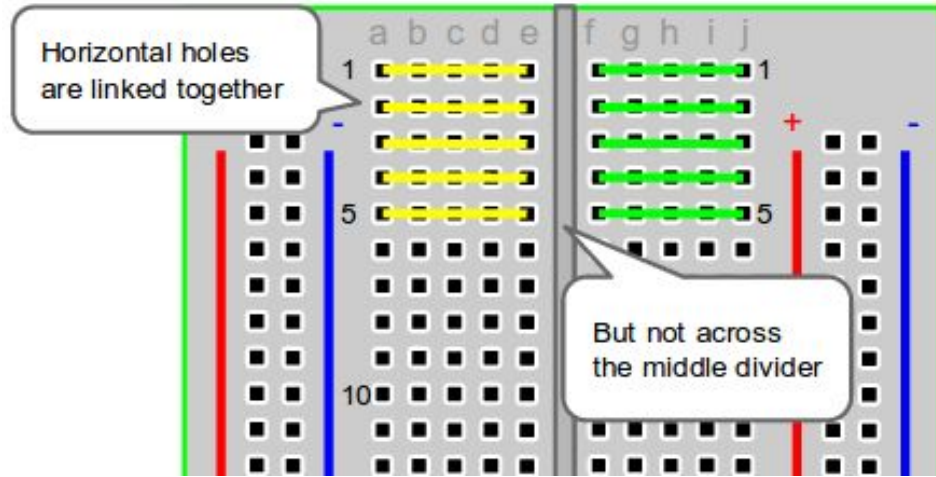


A button

# Review

---

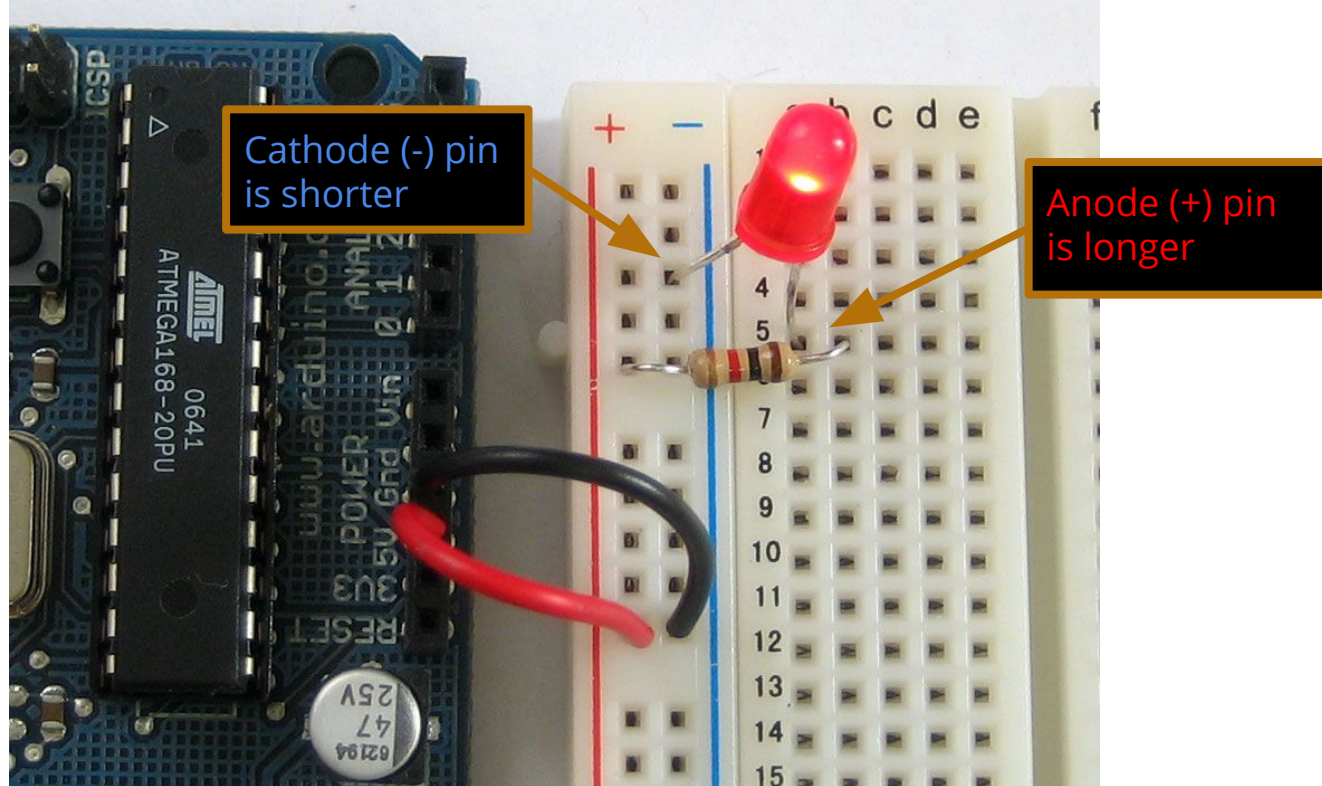
# Breadboards Explained



Tip: It is good practice to have your power input connected to the red/positive rail and your ground pin connected to the blue/negative rail.



# LEDs

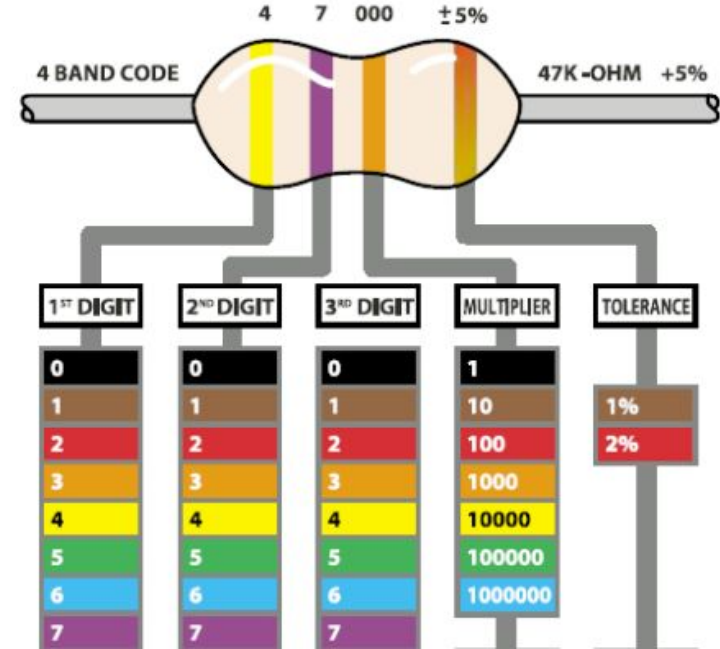


NOTE: Make sure the power input is connected to the Anode, and the ground pin is connected to the Cathode. Make sure you also have a resistor between either the power input and Anode, or the Cathode and ground pin. Failing to do either of these things can damage the LED or the Arduino.



# Resistors

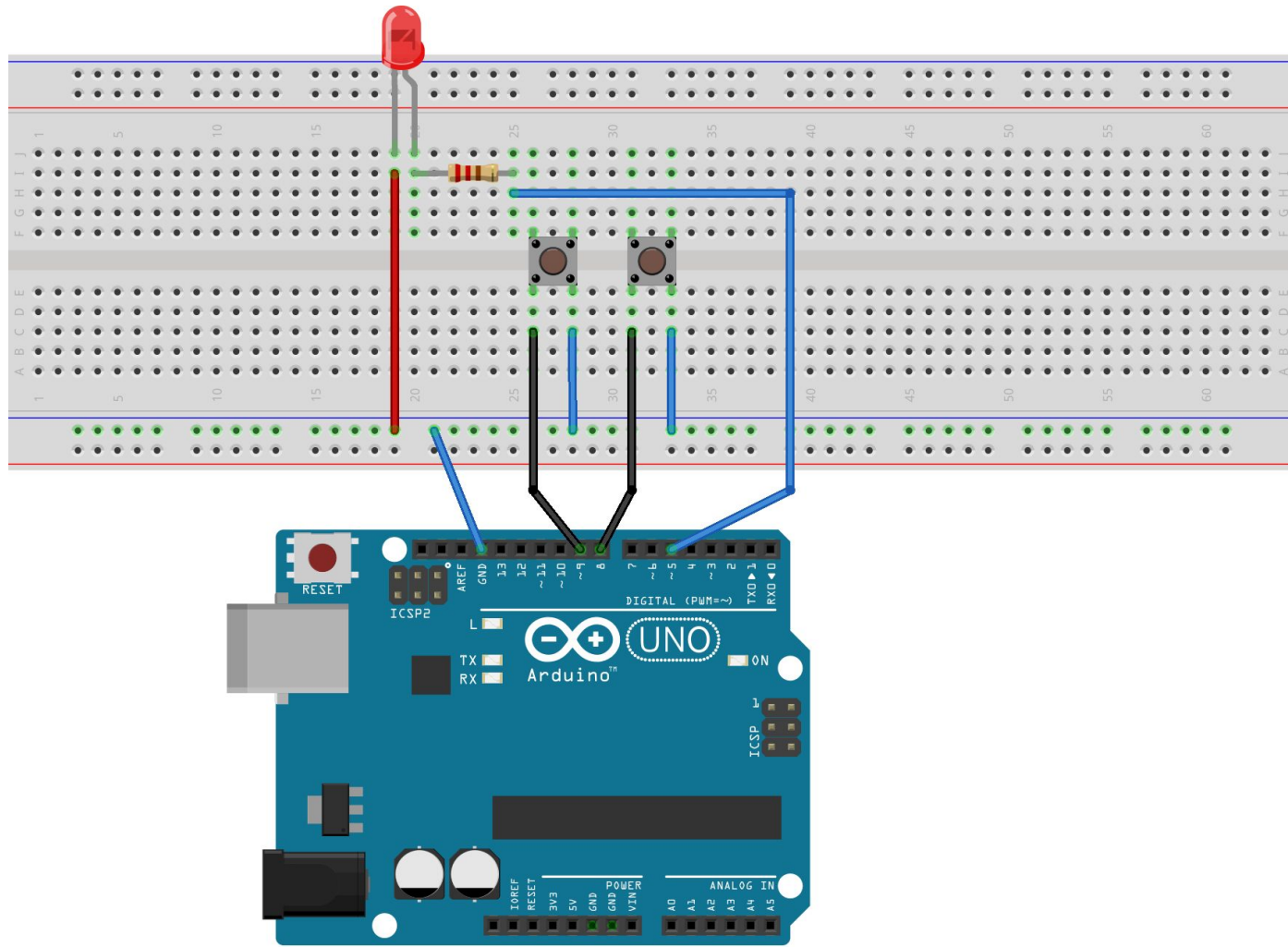
- Resistors slow the electric current, and control where and how fast the current flows
- Resistance value is measured in ohms  $\Omega$ , which is represented by colored stripes on the body of the resistor
- Each stripe has a different value depending on the color and location as shown in the reference chart
- A potentiometer is a variable resistor



# Project

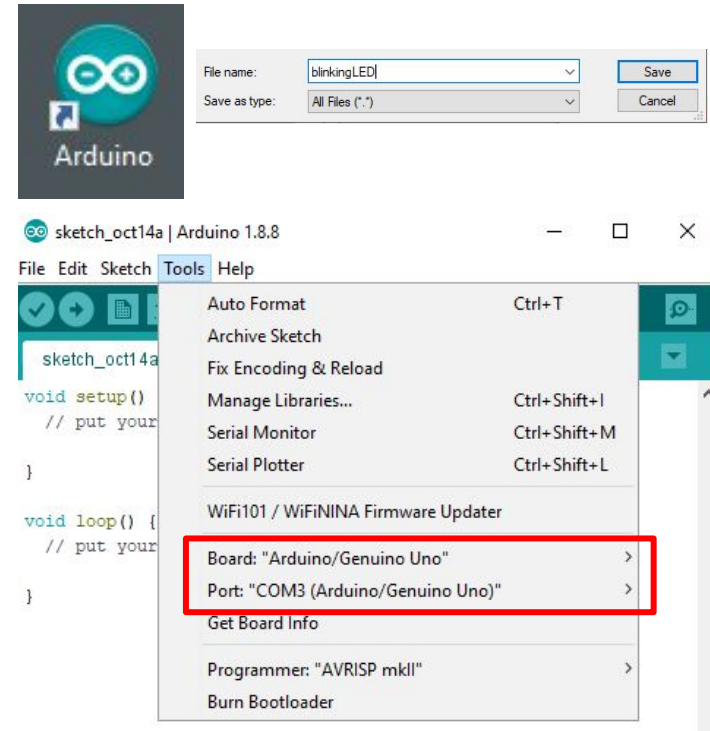
---

# Digital Input Schematic



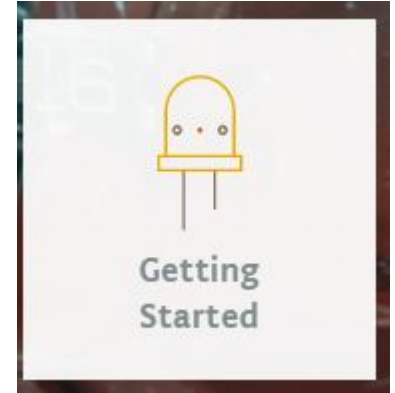
# Setting up Arduino

- Find and open Arduino on your desktop
- Click “File” in the top left corner and click save
- Save this tab as “digitalInputs”
- Connect the USB cord in your kit to the Arduino and the computer (USB port is on the left side of the monitor)
- Open the “Tools” Window and make sure the board has been recognized and the port is “COM#” with the name of the board after it




# Setting up Arduino (Online Web Editor)

- Search up `create.arduino.cc`
- Click Getting Started
- Scroll all the way down and click “Set up the Arduino Plugin”
- Click Next and follow the steps to download the plugin



# Setting up Arduino



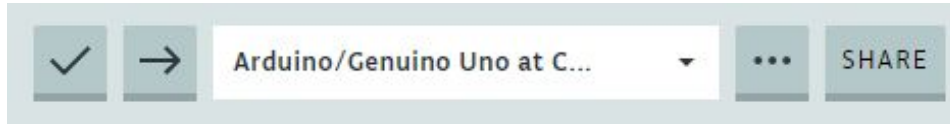
- If it doesn't automatically bring you to the login screen, click the 9 dots in the upper left hand corner and click Arduino Web Editor.
  - Click the Sign in with Google button
  - Sign in with your Google account
-  You must use a personal email!



The screenshot shows the login and account creation interface. On the left, under the heading 'LOGIN', there are two input fields: 'Username or e-mail' with the placeholder text 'USERNAME OR E-MAIL', and 'Password' with the placeholder text 'PASSWORD'. Below these fields is a link that says 'Forgot your username/password?'. At the bottom of the login section is a teal button labeled 'LOGIN'. On the right, under the heading 'CREATE A NEW ACCOUNT', there is a circular icon representing a user profile. At the bottom of the entire form, there is a button labeled 'Sign in with Google' which is highlighted with a red rectangular border.

# Getting Started

- Click “NEW SKETCH” in the top left corner
- Click on the sketch name and rename it “digitalInput”
- Connect the USB cord in your kit to the Arduino and the computer
- The type of Arduino should have been recognized. If not, please type in chat.





# Digital Inputs

**ledPin**, **buttonAPin**, and **buttonBPin** are variables of the data type **int**, meaning they hold integer values.

Each corresponds to a **pin number** on the Arduino.

To test your code, click the checkmark then the arrow!



digital\_inputs

```
int ledPin = 5; // single LED pin
int buttonAPin = 9; // the left button
int buttonBPin = 8; // the right button

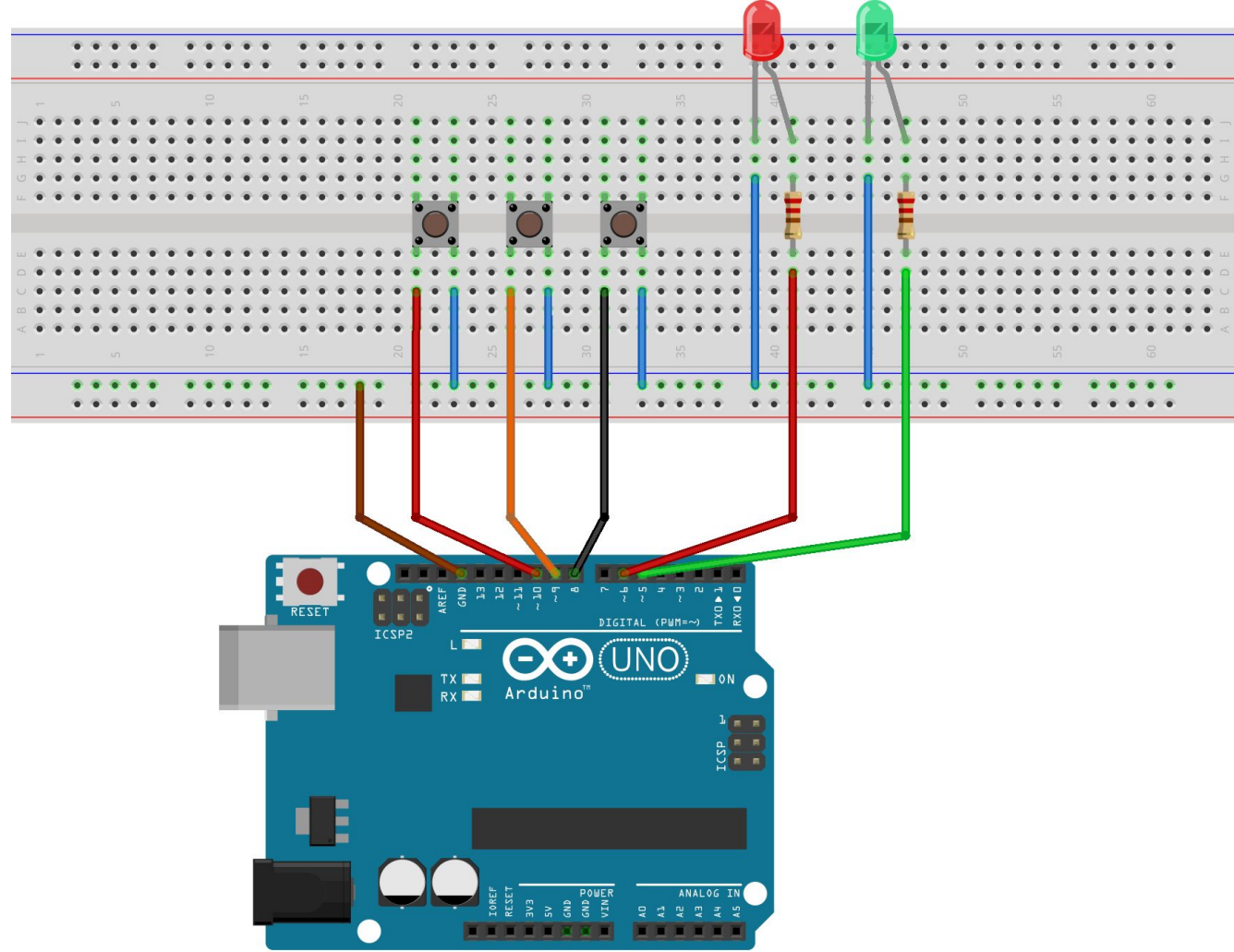
void setup() {
  // set buttons to be used as an input
  pinMode(ledPin, OUTPUT);
  pinMode(buttonAPin, INPUT_PULLUP);
  pinMode(buttonBPin, INPUT_PULLUP);
}

void loop() {
  if (digitalRead(buttonAPin) == LOW) {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBPin) == LOW) {
    digitalWrite(ledPin, LOW);
  }
}
```

The setup() function sets the led as an **output** device and the buttons as **input** devices.

The loop() function states that if **buttonA** is pushed, turn the LED **ON**. If **buttonB** is pushed, turn the LED **OFF**.

# Passcode Schematic



# Grab the Starter Code from GitHub

If you haven't made the repository yet, check these [slides](#)

- Go to your repository (username/embedded-systems-course)
- Click “Compare”
- Switch the repos so yours is the base repository and FTC9837 is the head
- Create pull request (x2)
- Merge pull request (and confirm)
- You should have the lesson2 folder in your repository

This branch is 1 commit behind FTC9837:main.

 Pull request  Compare

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



base repository: SamP923/embedded-systems-...

base: main



head repository: FTC9837/embedded-systems-c...

compare: main

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

Create pull request

# Passcode Code - setup()

Open the  
button\_passcode.ino  
file that you pulled  
from GitHub on your  
local system.

The variable  
declarations and  
setup() function  
similarly to the  
digital\_inputs file

```
button_passcode

// set pins for LEDs and buttons
int green_led_pin = 5;
int red_led_pin = 6;
int buttons[3] = {8,9,10};

// set hardcoded password
int passcode_arr[] = {0,1,2,0,0,0};
// store size of password array
const int SIZE = sizeof(passcode_arr) / sizeof(int);

// store user inputs
int states[SIZE];
int index_state = 0;

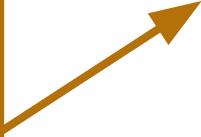
void setup() {
    // set leds as output and buttons as input
    pinMode(green_led_pin, OUTPUT);
    pinMode(red_led_pin, OUTPUT);
    for(int i = 0; i < 3; i++){
        pinMode(buttons[i], INPUT_PULLUP);
    }
    Serial.begin(9600);
    Serial.println("Good to go!");
}
```

The array  
**passcode\_arr[]**  
sets the values of  
the correct  
passcode and  
**size** stores the  
length of it.

# Passcode Code - loop() Part 1

Each of the if-statements checks if a **specific** button was pressed.


If so, add that to the array **states** that holds the user input, and print the number to the **Serial** Monitor so we can verify the input.



```
void loop() {  
  // check if inputs are pushed -- if so, add to the states array  
  if (digitalRead(buttons[0])==LOW) {  
    states[index_state] = 0;  
    index_state++;  
    Serial.println("0");  
    delay(250);  
  }  
  if (digitalRead(buttons[1])==LOW) {  
    states[index_state] = 1;  
    index_state+=1;  
    Serial.println("1");  
    delay(250);  
  }  
  if (digitalRead(buttons[2])==LOW) {  
    states[index_state] = 2;  
    index_state+=1;  
    Serial.println("2");  
    delay(250);  
  }  
}
```

# Passcode Code - loop() Part 1

To check if the passcode has been inputted, we first check if the correct number of values have been entered.



```
// if the user gets to the last index state (correct length of password),  
//check if the password matches.  
if (index_state == SIZE ){  
    if (checkpassword()){  
        digitalWrite(green_led_pin, HIGH);  
        delay(1500);  
        digitalWrite(green_led_pin, LOW);  
    }  
    else{  
        digitalWrite(red_led_pin, HIGH);  
        delay(1500);  
        digitalWrite(red_led_pin, LOW);  
    }  
    // reset the state array  
    index_state = 0;  
    reset_arr();  
    Serial.println("RESET");  
}  
}
```

Then, we use **checkpassword()** to see if the user entered the correct password.

If so, then we turn on **green\_led\_pin** for 1.5 seconds.

If not, we turn on **red\_led\_pin** for 1.5 seconds.

# Passcode Code - resetarr() & checkpassword()

```
//Reset the array and fill it with garbage values
void reset_arr(){
    for(int i = 0; i < SIZE; i++){
        states[i] = 999;
    }
}

//Check the password. If ANY of the values in the states array
//do not match the ones in the passcode array, return "false".
//Otherwise, return "true".
bool checkpassword(){
    for ( int i = 0; i < SIZE; i++){
        if (states[i] != passcode_arr[i] ){
            return false;
        }
    }
    return true;
}
```



# Thank you!

---

CCA Girls Who Code

[ccagirlswhocode@gmail.com](mailto:ccagirlswhocode@gmail.com)

[www.ccagirlswhocode.weebly.com](http://www.ccagirlswhocode.weebly.com)

# Production Team

Program Director: Stefan Prestrelski

Teaching Assistants: Sarah Luo, Samantha  
Prestrelski