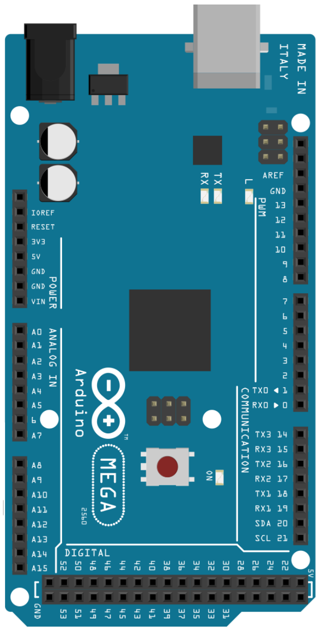
Communication



Electrical and Computer Engineering 5

Wireless Communication using Arduino

Developed by Raul Pegan, Scott Zhao and Professor Vikash Gilja

Professor John Eldon | Professor Vikash Gilja | Professor Drew Hall | Professor Truong Nguyen

What You Will Need

Materials:

- 1 Arduino Mega (or Uno)
- 1 USB Cable A-B
- 1 Bread board
- 10 Jumper Wires
- 5 Resistor (330 Ω)
- 1 IR LED
- 1 IR Receiver (TSOP382)
- 1 IR Remote
- 1 Pushbutton
- 1 RGB LED

Machinery:

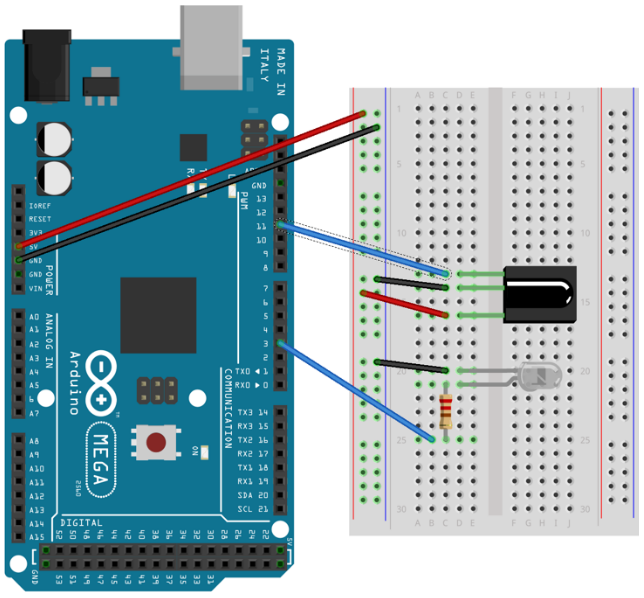
- Computer / Laptop

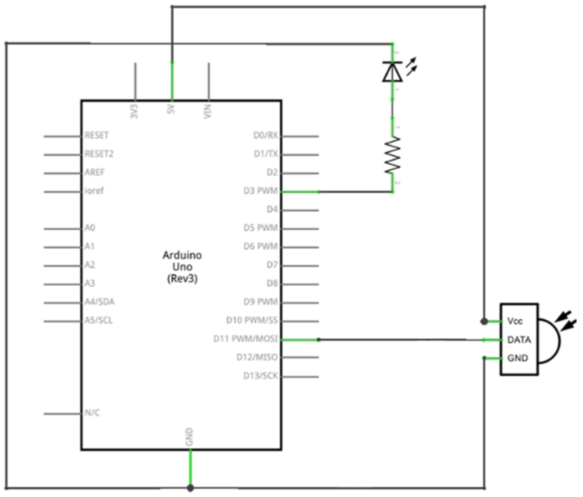
Software:

- Arduino Software (IDE)
- Timer Library (https://code.google.com/p/arduino-timerone/downloads/list) TimeOne-**r11**.zip
- Ken Shirriff's IR Remote Arduino Library (https://github.com/z3t0/Arduino-IRremote/releases) IRremote.zip

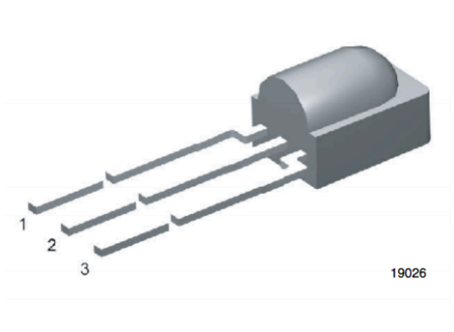
Challenge #1: IR Sensor and IR LED

Objective: As you might have experienced, the photo-resistor had difficulties recognizing when the LED was on (see Lab 0). This is due to noise caused by many other light sources. A way to alleviate this and improve the circuit is to incorporate Infrared (IR) light into the equation. IR is invisible to humans, but it is detectable with the right sensors. The goal of this challenge is to emulate the last circuit by replacing visible light with IR. Since the IR sensor only does a digital read, the values of the output will only be either 1 or 0.





Components:
1. 1 Arduino Mega
2. 1 Bread-board
3. 7 Male to Male wires (2 red for voltage/pin, 3 black (ground), 2 colored wire for I/O)
4. 2 Female to Male Wires (allow for extension of IR LED)
5. 1 resistors (330 ohms)
6. 1 IR LED
7. 1 TSOP382 IR receiver.



MECHANICAL DATA

Pinning for TSOP382.., TSOP384..:

1 = OUT, 2 = GND, 3 = V_S

Data Sheet: https://www.vishay.com/docs/82491/tsop382.pdf

You will also be working with new concepts: ***libraries*** and ***timer interrupts***. Sometimes we want to tackle a problem that has already been solved before. If we all had to build everything from scratch, it would not be the most efficient way to create our projects, so we resort to building from other people's code. These solutions are compiled into packages called "libraries". A library has a variety of files in it, all organized in a specific structure. They are designed to work with Arduino and to flow nicely with the IDE. Adding a library is very simple. All you have to do is go to **Sketch** → **Include Library** → **Add .ZIP Library** and then select the .ZIP file from your computer. Libraries are often found all over the internet, anyone can create them! Arduino libraries are written in C++ (an object oriented programming language), which looks slightly different from the Arduino C that you've been learning.

In the sketch (Arduino IDE interface), the Arduino repeats the loop() code over and over again, and the speed in which this code runs depends on how intense the work is. So the time it takes to run each iteration of loop() will vary. Sometimes we want to run a specific piece of code in a precise timed manner. We are able to accomplish this using timer interrupts. Timer interrupts ***interrupt*** whatever is running inside loop() in order to run the time sensitive code of your choosing. We are able to set the specific frequency in which we require our sensitive code to run, with great results.

In this example, we need to turn the IR LED on and off at a frequency of *38kHz*, or every 26 microseconds. This is necessary because our IR receiver has this characteristic, it will only read IR in that specific frequency. In order to accomplish this, we will use a Timer library, found at https://code.google.com/p/arduino-timerone/downloads/list

Please download the TimerOne-r11.zip library. In the Arduino IDE you can install the .ZIP file by selecting the top tab **Sketch** → **Include Library** → **Add .ZIP Library**…. From there, find the .ZIP file you downloaded. In the code below you will see

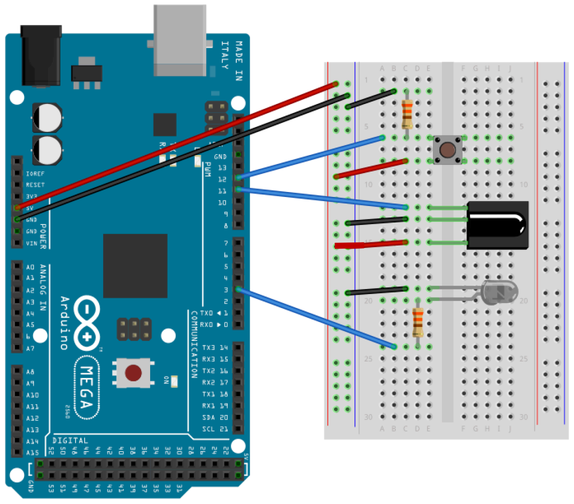
```
#include <TimerOne.h>
```

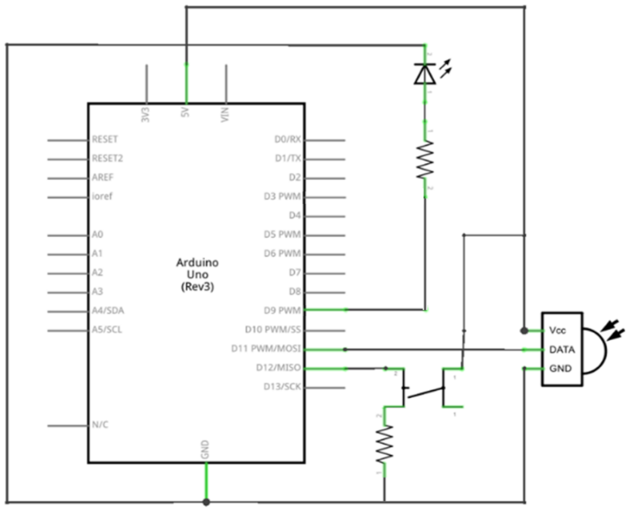
Instead of writing that in, you should include the library "TimerOne-r11" in your code by selecting the tab **Sketch** → **Include Library** → Then scroll down and select **TimerOne-r11**. Arduino IDE will generate that include line for you.

Coding/Programming: The code is very similar to the photoresistor program. Can you identify what changed?

```
#include <TimerOne.h>
// constants won't change. They're used here to
// set pin numbers:
const int ledPin =  _______;       // the number of the IR LED pin
const int sensorPin = ______;      // the number of the IR sensor pin

// variables will change:
int lightLevel;          // variable for storing the sensor data

void setup() {
  Serial.begin(9600);
  pinMode(__________, INPUT);   // Input to Arduino
  pinMode(__________, OUTPUT);   // Output from Arduino

  Timer1.initialize(26); // set a timer of length 26 microseconds (or 38kHz)
  Timer1.attachInterrupt(callback); // attach the routine that will happen at the interrupt
}

void loop(){
  lightLevel = digitalRead(sensorPin);
  Serial.print("The IR Sensor is reading:");
  if (lightLevel == 0) {
    Serial.println("IR Communication ON");
  }
  else {
    Serial.println("IR Communication OFF");
  }
  delay(500);    // delay to slow serial monitor output
}

void callback() // time sensitive function to toggle the IR LED
{
  digitalWrite(ledPin, digitalRead(ledPin) ^ 1); //Toggle the IR LED
}
```

Challenge #2: IR Remote





Objective: Now we get to the cool stuff. IR is the technology used in TV remotes, they transmit the exact same type of light from the last exercise! So we could potentially use our circuit to function just like a TV remote. And that is what we will do.

Components:
1. 1 Arduino Mega
2. 1 Bread-board
3. 10 wires (3 red for voltage/pin, 4 black (ground), 3 colored wire for I/O)
4. 2 resistors (330 ohms)
5. 1 IR LED
6. 1 TSOP382 IR receiver
7. 1 Push Button
8. 1 IR remote

Resources:
We will be using Ken Shirriff's IR library found on GitHub:

https://github.com/z3t0/Arduino-IRremote/releases/download/v2.4.0/IRremote.zip

This library contains all the encoding and decoding necessary in order to properly send and receive the information. Since it is so complex, and someone has already done it, it is always better to use a library.

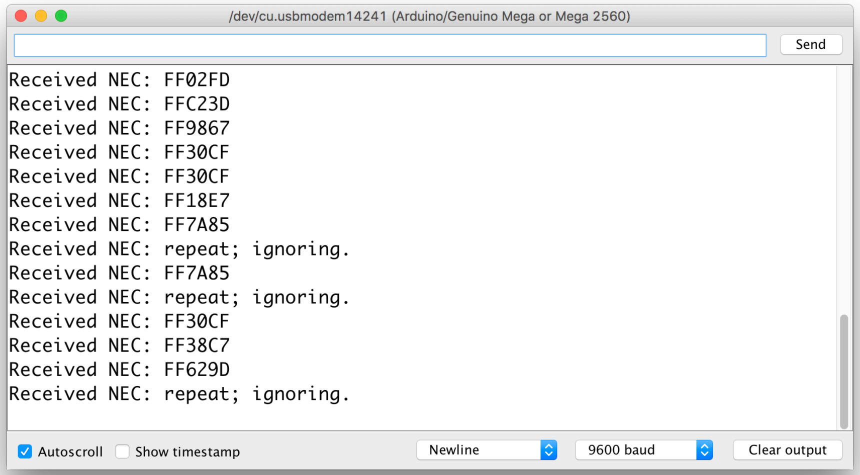
After downloading and adding the library by clicking **Sketch** → **Include Library** → **Add .ZIP Library** and then select *IRremote.zip* you just downloaded, **open the "IRrecord" program from File → Examples → IRremote → IRrecord**

Try to read through the code and understand the basic concepts of it all. The comments do a good job at explaining what the code does, which is act as a 'relay'. This means that the circuit will read the signal sent from the remote, and then send the same signal through the IR LED whenever you press the pushbutton. You can have a lot of fun with this. Program it to send the TV on/off button, and sneakily turn your roommate's TV off, even when they have the remote with them.

Tasks:

- Take out your remote controller. *Make sure you've put in the battery*. Upload the **IRrecord** program to Arduino without any modification, then open the Serial Monitor. Choose six buttons on your remote controller, push them, and record down the decoded HEX codes shown on Serial Monitor for each button. (*You will be using these HEX codes for next challenge*)

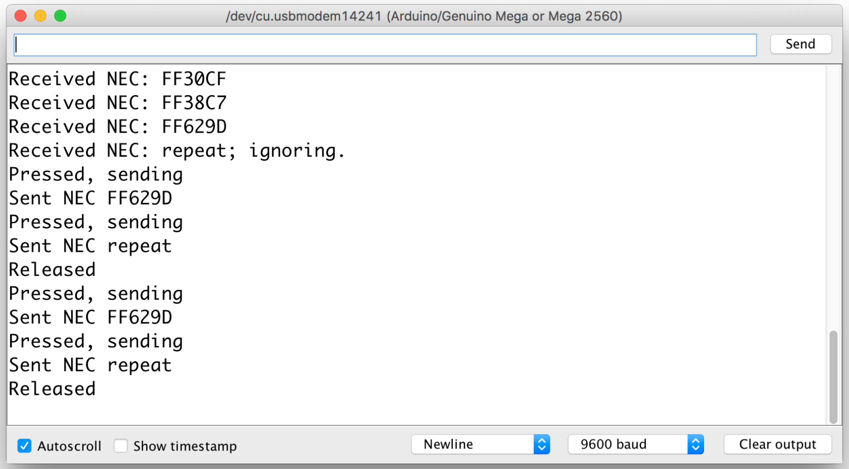
Example Serial Monitor:



```
Received NEC: FF02FD
Received NEC: FFC23D
Received NEC: FF9867
Received NEC: FF30CF
Received NEC: FF30CF
Received NEC: FF18E7
Received NEC: FF7A85
Received NEC: repeat; ignoring.
Received NEC: FF7A85
Received NEC: repeat; ignoring.
Received NEC: FF30CF
Received NEC: FF38C7
Received NEC: FF629D
Received NEC: repeat; ignoring.
```

- (Optional) Test the relay! Once the **IRrecord** program records a button press, try pointing your IR LED to someone else's IR sensor, press the button and see if the same signal gets decoded by their Arduino on Serial Monitor.

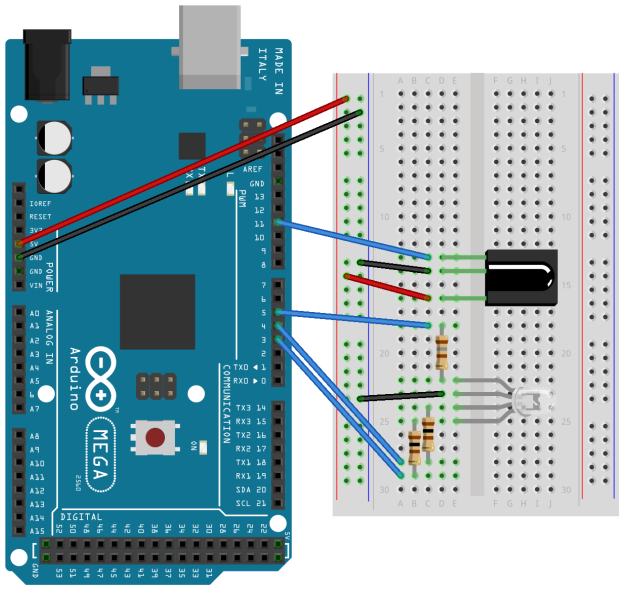
Example Serial Monitor on the sender side after pressing button:



```
Received NEC: FF30CF
Received NEC: FF38C7
Received NEC: FF629D
Received NEC: repeat; ignoring.
Pressed, sending
Sent NEC FF629D
Pressed, sending
Sent NEC repeat
Released
Pressed, sending
Sent NEC FF629D
Pressed, sending
Sent NEC repeat
Released
```

Challenge #3: IR Controlled RGB LED

Objective: Time for some RGB fun! In this lab, we will use the IR remote to control the color of an RGB LED. We are also going to use a different programming concept: ***functions.*** Functions are blocks of code used to perform a specific task.



Components:

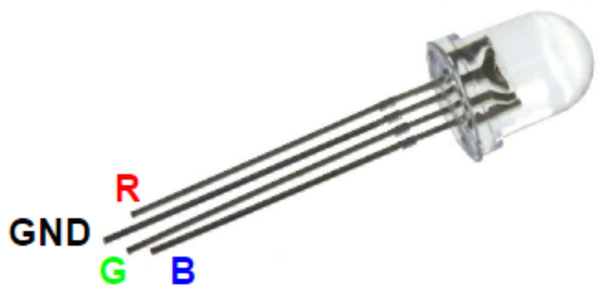
1. 1 Arduino Mega
2. 1 Bread-board
3. 9 wires (2 red for voltage/pin, 3 black (ground), 4 colored wire for I/O)
4. 3 resistors (1 180ohms for Red pin, 2 100ohms for Blue and Green pins), scroll down to see how to approximate these resistor values using the 330ohms resistors given
5. 1 RGB LED
6. 1 TSOP382 IR receiver

Note: For the nine wires, choose 2 red for voltage/pin, 3 black (ground), 4 any colored wire for I/O (input/output). The longest pin on the RGB LED is the ground pin, the other three depend on the manufacturer so check the datasheet!

You have already used many different functions at this point. When you use analogRead(), you are calling a function from deep inside the Arduino libraries. In the Timer example, callback() is a function. Functions are useful because they allow you to split up your code into smaller segments, instead of putting it all in loop(). This helps with simplicity of the code. When you want to repeat a specific task many times, it is much easier to call a function instead of rewriting the same code over and over again. We will use functions to simplify the process of changing the LED color.

Data Sheets:

The RGB LED is a common component. Check out the following data sheet to learn more about the specific component and find out which pins are for Red, Green, Blue, and Ground.



https://www.sparkfun.com/datasheets/Components/YSL-R596CR3G4B5C-C10.pdf

Let's look at one specific function, `setColor`:

```
void setColor(int red, int green, int blue){
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

This function's sole purpose is to write to the three RGB pins (redPin, greenPin, and bluePin) and assign the three parameters `(int red, int green, int blue)` as their values. `setColor` has return type void, meaning that it does not send any information back to the caller.

```
#define EXAMPLE_KEY 0xFD00FF // Example
#define ON_OFF_KEY 0x______
#define TOGGLE_COLOR_KEY 0x______
#define BRIGHTNESS_UP_KEY 0x______
#define BRIGHTNESS_DOWN_KEY 0x______
#define CIRCLE_KEY 0x______
#define CUSTOM_FUNCTION_KEY 0x______
```

At the beginning of the program, we defined the HEX codes for a series of buttons. Fill in the HEX codes you copied from Serial Monitor in last challenge to here. Note that you are informing Arduino that a value is in HEX by starting with a "***0x***". An example is provided on the first line, "EXAMPLE_KEY".

```
  // Color operations
   if (isOn == true) {
     if (results.value == TOGGLE_COLOR_KEY)      // Toggle active colors
       toggleActiveColor();
     else if (results.value == BRIGHTNESS_UP_KEY) // Brightness up
       brightnessUp();
     else if (results.value == BRIGHTNESS_DOWN_KEY) // Brightness down
       brightnessDown();
     else if (results.value == CIRCLE_KEY) // Circle
       circle();
     else if (results.value == CUSTOM_FUNCTION_KEY) { // Be creative with your own code
       // FILL IN YOUR OWN CODE
     }
  }
```

Part of our loop() contains a series of *if / else if*s, telling Arduino to call different functions to perform different tasks when specific key press is received by IR receiver. Notice we put what we defined at the beginning of the program for different key codes into the if statements.

Fill in your own code for what will happen if you press the "CUSTOM_FUNCTION_KEY" you defined at the beginning. You could call function `powerOn();` and `powerOff();` with delays in between to blink the RGB LED, or modify the values for red, green, and blue, then call `setColor(red, green, blue);` to update the color, etc. Be creative!

Wiring: Using the circuit at the beginning of the section, wire the ARDUINO as shown.

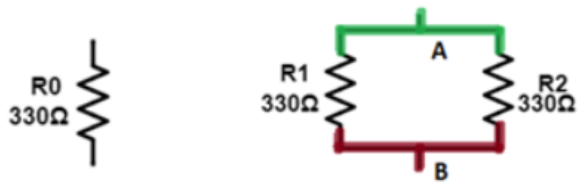
We can make an LED more vibrant by lowering the resistance. However, instead of 100 ohm and 180 ohm resistors, we only have 330 ohm resistors which would make the LED dimmer. How do we fix this? We can set multiple 330 ohm resistors in parallel. Because the current has more paths to follow, charges reach the LED quicker and we get an equivalent resistance (R_{eq}) from the resistors set in parallel ($R_1 \ldots R_n$) that is lower than the resistance of an individual 330 ohm resistor.

$$R_{eq} = \left(\frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}\right)^{-1}$$

Equation for Equivalent Resistance from Parallel Resistors

Components are in parallel when the voltage difference across them is the same. The voltage value before and after a component is different because a component changes the voltage as it supplies or uses energy.

We can see if components have the same voltage difference by examining if components share the same two nodes. A node is an area of the same voltage value so it exists between components and changes after a component.



$$R = 330\Omega \qquad Req = \left(\frac{1}{330} + \frac{1}{330}\right)^{-1} = 165\Omega$$

For example, resistors R1 and R2 are in parallel because they share nodes A and B. This means that the voltage difference across them, $V_A - V_B$, is the same.

We can reduce the resistance further by adding more resistors in parallel!

Tasks:

- Download the code **Lab1_Challenge3.ino** from *TritonED*.
- Fill in the HEX codes you recorded in last challenge to the button definitions at the beginning of the program.
- Fill in your own code for what will happen if you press "CUSTOM_FUNCTION_KEY".
  - Feel free to add more keys definition on top then add more custom function by adding similar else-ifs.
- Upload the code and open the Serial Monitor. Have fun with your remote controlled RGB LED!

Coding/Programming:

```
#include <IRremote.h>
#define ON_OFF_KEY 0x______
#define TOGGLE_COLOR_KEY 0x______
#define BRIGHTNESS_UP_KEY 0x______
#define BRIGHTNESS_DOWN_KEY 0x______
#define CIRCLE_KEY 0x______
#define CUSTOM_FUNCTION_KEY 0x______

const int RECV_PIN = 11;
const int redPin = 5;
const int greenPin = 4;
const int bluePin = 3;
int red = 0;  // Initial intensity
int green = 0;
int blue = 0;
int currentBrightness = 0;  // Initial active color intensity
String targetColor = "RED"; // Initial active color
bool isOn = false;
IRrecv irrecv(RECV_PIN);  // Initialize IR Library
decode_results results;   // Initialize IR Library
void setup() {
  Serial.begin(9600);   // Start Serial
  irrecv.enableIRIn();  // Start the receiver
  // Set the three LED Pins as outputs
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
void loop() {
  if (irrecv.decode(&results)) {
    // Serial.println(results.value, HEX);  // Prints HEX code IR receiver receives
    irrecv.resume(); // Receive the next value
    // Power switch
    if (results.value == ON_OFF_KEY) {
      if (isOn == true) {
        Serial.println("Turning Off"); powerOff();
      }
      else if (isOn == false) {
        Serial.println("Turning On"); powerOn(); showCurrentActiveColor();
      }
    }
    // Color operations
    if (isOn == true) {
      if (results.value == TOGGLE_COLOR_KEY)      // Toggle active colors
        toggleActiveColor();
      else if (results.value == BRIGHTNESS_UP_KEY) // Brightness up
        brightnessUp();
      else if (results.value == BRIGHTNESS_DOWN_KEY) // Brightness down
        brightnessDown();
      else if (results.value == CIRCLE_KEY) // Circle
        circle();
      else if (results.value == CUSTOM_FUNCTION_KEY) { // Be creative with your own code
        // FILL IN YOUR OWN CODE
      }
    }
  }
  delay(100);
}
// Print current color setting on serial monitor
void showCurrentActiveColor() {
  Serial.print("Now controlling color ");
  Serial.println(targetColor);
}
// Turn Off LED
void powerOff() {
  analogWrite(redPin, 0);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 0);
  isOn = false;
}
// Turn on LED
void powerOn() {
  updateColor();
  isOn = true;
}
// Turn up brightness for active color
void brightnessUp() {
  currentBrightness = (currentBrightness + 25) % 256;
  updateColor();
}
// Turn down brightness for active color
void brightnessDown() {
  currentBrightness = max(0, (currentBrightness - 25) % 256);
  updateColor();
}
// Toggle active color that you are changing brightness on
void toggleActiveColor() {
  if (targetColor == "RED") {
    targetColor = "GREEN";
    currentBrightness = green;
  }
  else if (targetColor == "GREEN") {
    targetColor = "BLUE";
    currentBrightness = blue;
  }
  else if (targetColor == "BLUE") {
    targetColor = "RED";
    currentBrightness = red;
  }
  showCurrentActiveColor();
}
// Update LED color to current setting
void updateColor() {
  if (targetColor == "RED")
    red = currentBrightness;
  else if (targetColor == "GREEN")
    green = currentBrightness;
  else if (targetColor == "BLUE")
    blue = currentBrightness;
  setColor(red, green, blue);
  showColorOnSerialMonitor();
}
// Display current color in Serial Monitor
void showColorOnSerialMonitor() {
  Serial.print("Red ");
  Serial.print(red / 255.0);
  Serial.print("\tGreen ");
  Serial.print(green / 255.0);
  Serial.print("\tBlue ");
  Serial.println(blue / 255.0);
}
// Output desired voltages to LED
void setColor(int red, int green, int blue) {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
// Circle through different colors
void circle() {
  red = 255;
  green = 127;
  blue = 0;
  Serial.println("Start Circle");
  while (true) {
    red = (red + 25 + 256) % 256;
    green = (green + 25 + 256) % 256;
    blue = (blue + 25 + 256) % 256;
    setColor(red, green, blue);
    showColorOnSerialMonitor();
    delay(150);
    if (irrecv.decode(&results)) {
      Serial.println("Stop Circle");
      irrecv.resume();
      return;
    }
  }
}
```

Resources

The Arduino website is a very good resource for grasping circuits and Arduino in general. Use the following link and click on the "Learning" tab, where you will find how to get started, tutorials, and other useful references!

https://www.arduino.cc/en/Main/Documentation#

SparkFun is a vendor that sells a variety of platforms, one of them Arduino. Luckily, they have great resources on the site to help you learn Arduino.

https://learn.sparkfun.com/tutorials/what-is-an-arduino

Adafruit is a DIY (Do-It-Yourself) vendor that sells and utilizes a variety of platforms. Luckily, they have a section dedicated to Arduino! Simply click on the "Learn" tab and follow the Arduino links. It will take you to a large amount of projects for you to build on your own!

https://www.adafruit.com

Instructables is a good project-based resource with a variety of platforms just like Adafruit. Check the website out for more interesting projects!

http://www.instructables.com

Last updated 1/14/2019