Hacktoberfest 2018

Table of Contents

[Sign Up For Hacktoberfest 3](#_Toc527317633)

[GitHub 3](#_Toc527317634)

[Register for Hacktoberfest 3](#_Toc527317635)

[Lesson 1 – Hello World / Blink 4](#_Toc527317636)

[Lesson 2 – Fade 6](#_Toc527317637)

[Lesson 3 – IR Remote On / Off 7](#_Toc527317638)

[Adding the IR library 7](#_Toc527317639)

[IR Module Fritzing 8](#_Toc527317640)

[IR Receiver Code 9](#_Toc527317641)

[Lesson 4 – IR Remote Up / Down 12](#_Toc527317642)

# Sign Up For Hacktoberfest

We need to sign up for hactoberfest to participate. Before we can do that we need to create an account at GitHub.

## GitHub

GitHub is a git repository host, where people can store revisions of their source code. You can create a new project and share your source code with other developers, or you can contribute with other developers on their projects.

Sign up for a new account at;

<https://github.com/join?source=header-home>

## Register for Hacktoberfest

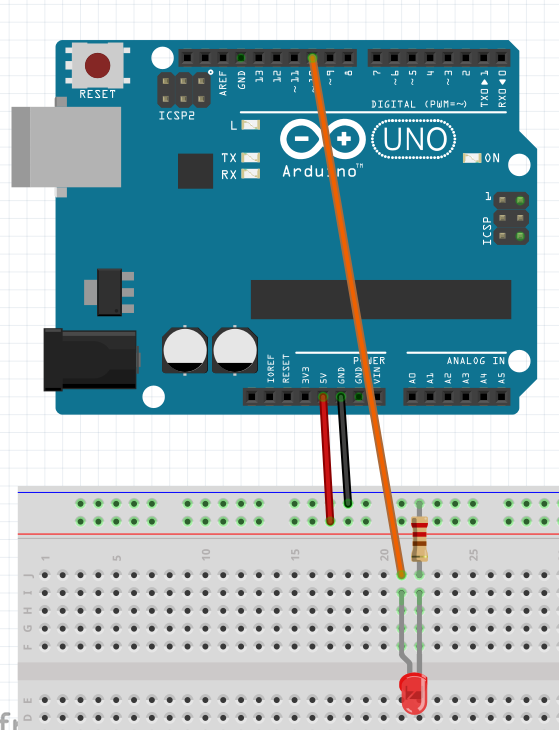
Hacktoberfest is an annual event attempting to get people involved in developing and contributing to open source projects.

Register to participate at;

<https://hacktoberfest.digitalocean.com/sign_up/register>

# Lesson 1 – Hello World / Blink

First lesson is to control an LED using a micro controller.



// Our LED is connected to pin 10

const int BLINK\_LED = 10;

/\*\*

\* The setup method gets called only once at the beginning of the application

\*/

void setup() {

// Before we can writ the LED we have to open this PIN for output

pinMode(BLINK\_LED, OUTPUT);

}

/\*\*

\* The loop method gets called over and over again as long as the board has power

\*/

void loop() {

// Turn the LED on

digitalWrite(BLINK\_LED, HIGH);

delay(1000); // Pause for 1000 MS or (1 second)

// Turn the LED off

digitalWrite(BLINK\_LED, LOW);

delay(1000); // Pause for 1000 MS or (1 second)

}

# Lesson 2 – Fade

Add the functionality to fade an LED. We will use this later with the IR remote to turn the brightness up and down.

Add the following constants and variable declarations at the top of the sketch.

const int FADE\_AMOUNT = 5; // How quickly to fade the LED by

int brightness = 0; // Variable to hold the current brightness of the LED start it at 0 or completely off

int direction = 1; // Variable to use for increasing the brightness or decreasing it (0 = down, 1 = up)

Add the following two functions.

/\*\*

\* Increases the brightness of the LED

\*/

void increaseBrightness() {

// Add the fade amount to the brightness to increase it

brightness += FADE\_AMOUNT;

// The LED can only go up to 255 (the maximum value for an 8 bit register)

if (brightness + FADE\_AMOUNT > 255) {

brightness = 255;

}

// Pulse the LED at the given brightness

analogWrite(BLINK\_LED, brightness);

}

/\*\*

\* Increases the brightness of the LED

\*/

void decreaseBrightness() {

// Subtract the fade amount from the brightness to decrease it

brightness -= FADE\_AMOUNT;

// The LED can only go down to 0

if (brightness - FADE\_AMOUNT < 0) {

brightness = 0;

}

// Pulse the LED at the given brightness

analogWrite(BLINK\_LED, brightness);

}

The contents of our main loop should look like the following.

if (direction == 1 && brightness >= 255) {

direction = 0;

}

else if (direction == 0 && brightness <= 0) {

direction = 1;

}

if (direction == 1) {

increaseBrightness();

}

else {

decreaseBrightness();

}

// Wait for 30 ms so we can see it slowly fade up or down

delay(30);

# Lesson 3 – IR Remote On / Off

In this lesson we will add IR functionality to our project so we can control the LED using a remote control.

For this we will take advantage of an open source project another developer created and posted on github.

<https://github.com/z3t0/Arduino-IRremote>

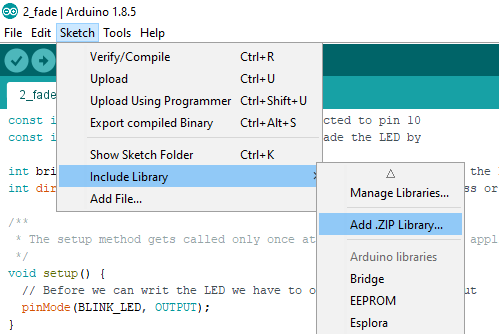
This is a library that was built to take care of some of the more complicated aspects of receiving an IR signal, so we can focus on the application logic without having to worry about the hard details of working with the IR serial protocols.

## Adding the IR library

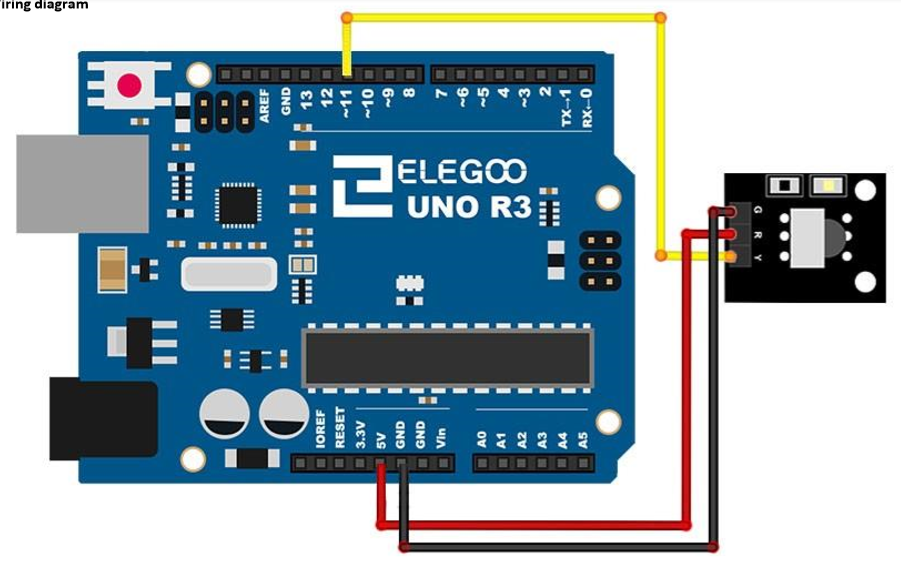
We will add the library from the zip file in our resources.

Click Sketch->Include Library->Add .ZIP Library

And select the IRremote.zip file.



## IR Module Fritzing



See the PDF in the resources for more conversation on the IR module.

## IR Receiver Code

Include the IRremote library into your project.

#include "IRremote.h"

Add a new constant to represent our GPIO pin that we will use to read the IR data from.

const int IR\_RECEIVER = 11; // Our IR receiver will be connected to GPIO Pin 11

We will now create a couple of new objects from our IR library to help us with the code.

These classes “wrap up” all of the complicated IR code into easy objects that we can use without our code.

IRrecv irReciever(IR\_RECEIVER); // create instance of 'irrecv'

decode\_results irResults; // create instance of 'decode\_results'

We will now need to create a function to toggle our LED from off to on.

/\*\*

\* Turns the LED On or off

\* If it's on already it will turn it off..

\* If it's completely off it will turn it on at full brightness

\*/

void toggleLED() {

if (brightness > 0) {

brightness = 0;

}

else {

brightness = 255;

}

analogWrite(BLINK\_LED, brightness);

}

We will also create a function to handle the IR codes we receive from the remote. We will map all the buttons, we will only wire up the power switch right now.

/\*\*

\* This function translates a code received from the remote to actual

\* application logic that we want to perform within our application.

\*/

void translateIR() {

switch(irResults.value) {

case 0xFFA25D: // POWER

toggleLED();

break;

case 0xFFE21D: // FUNC/STOP

case 0xFF629D: // VOL+

case 0xFF22DD: // FAST BACK

case 0xFF02FD: // PAUSE

case 0xFFC23D: // FAST FORWARD

case 0xFFE01F: // DOWN

case 0xFFA857: // VOL-

case 0xFF906F: // UP

case 0xFF9867: // EQ

case 0xFFB04F: // ST/REPT

case 0xFF6897: // 0

case 0xFF30CF: // 1

case 0xFF18E7: // 2

case 0xFF7A85: // 3

case 0xFF10EF: // 4

case 0xFF38C7: // 5

case 0xFF5AA5: // 6

case 0xFF42BD: // 7

case 0xFF4AB5: // 8

case 0xFF52AD: // 9

case 0xFFFFFFFF: // REPEAT

default:

break; // Do nothing

}

delay(100); // Pause for a little bit to "debounce" the remote button press

}

Now in our main loop we will check to see if we have received any IR codes and then act on it accordingly.

/\*\*

\* The loop method gets called over and over again as long as the board has power

\*/

void loop() {

if (irReciever.decode(&irResults)) {

translateIR();

irReciever.resume();

}

}

# Lesson 4 – IR Remote Up / Down

Now that we can interact with the remote, can you figure out how to use the volume up and volume down buttons to dim the LED?

Hint.. Modify the switch statement in the translateIR function to call your increase / decrease brightness functions.