Wireless Network

Report of LAB4: A Multi-Protocol Infrared Remote Library for the Arduino

Emil Vega-Gualán

3 de abril de 2019

I. Brief description

This work is oriented to implement an electronic system using Arduino Uno to: control LED's using Unused IR Remote keys, design and remote control of a RGB LED, Arduino Remote Emulator, and develop a simple burglar alarm. IR (infrared) is one of the most common modes of communication between electronic devices as TV [1]. The naked eye can't see the light because the wavelength is longer than visible light. However, this light can be seen through a camera. According to Lucas in his publication in Live Science [2], he says that, "IR frequencies range from about 3 gigahertz (GHz) up to about 400 terahertz (THz), and wavelengths are estimated to range between 1,000 micrometers (μm) and 760 nanometers (2.9921 inches), although these values are not definitive, according to NASA". In this way, the IR led transmits infrared waves in the range of 875 nm to 950 nm. These LEDs are made up of gallium arsenide or aluminum gallium arsenide. The IR sensors consists of an IR LED and a photo-diode pair. It emits IR radiation, which, on receiving at the photo-diode dictates the output of the sensor [1].

This system will use a remote control used in the most audio and video equipment. This control emits a infra-red light. This infra-red light can be radiated by everything that radiates heat, even our body [1]. We will use this remote control to control LEDs and its brightness through the IR sensor. Also, we are going to use an RGB LED and, in the same way, we will control its brightness. An RGB LED is a combination of 3 LEDs in just one package: red, green blue [3]. Finally, we are going to develop a burglar alarm. This alarm it is going to emit a sound when the transmission of an IR signal is interrupted by some object.

To develop this work, it is necessary to know certain concepts like: modulation of the signal, electrical intensity, and the principle of direct incidence. In this way, Modulation of the signal is the answer to make our signal stand out above the noise. With modulation we make the IR light source blink in a particular frequency. The IR receiver will be tuned to that frequency, so it can ignore everything else [1]. On the other hand, intensity of electric field at a point is measured by the force being experienced by a unit charge placed at that point[4]. Furthermore, The principle of direct incidence is when the IR LED is held directly in front of the IR receiver [5].

The goals of this work are:

- Understand the basic principles of radio communications
- Figure out how to set up an IR sensor.
- Understand how IR remote controls work
- Learn to use the IRRemote library.
- Learn how combine IR sensors with other devices (Buzzer)

II. Implementation and Development

For the implementation of this system we need the following materials:

- 1 x Arduino Uno
- 1 x Vs1838b or Vs1838
- 1 x Breadboard
- 2 x LEDs
- 1 x RGB LED
- 1 x IR LED transmitter
- 1 x IR LED receiver
- 5 x 200 Ω resistors
- 1 x 100 Ω resistor
- 1 x 10 k Ω resistor
- 1 x Buzzer
- Jumper wires

The schematic of the circuit is shown in the figure 1. In this figure are all the devices connected through a breadboard. The schematic was done in fritzing.

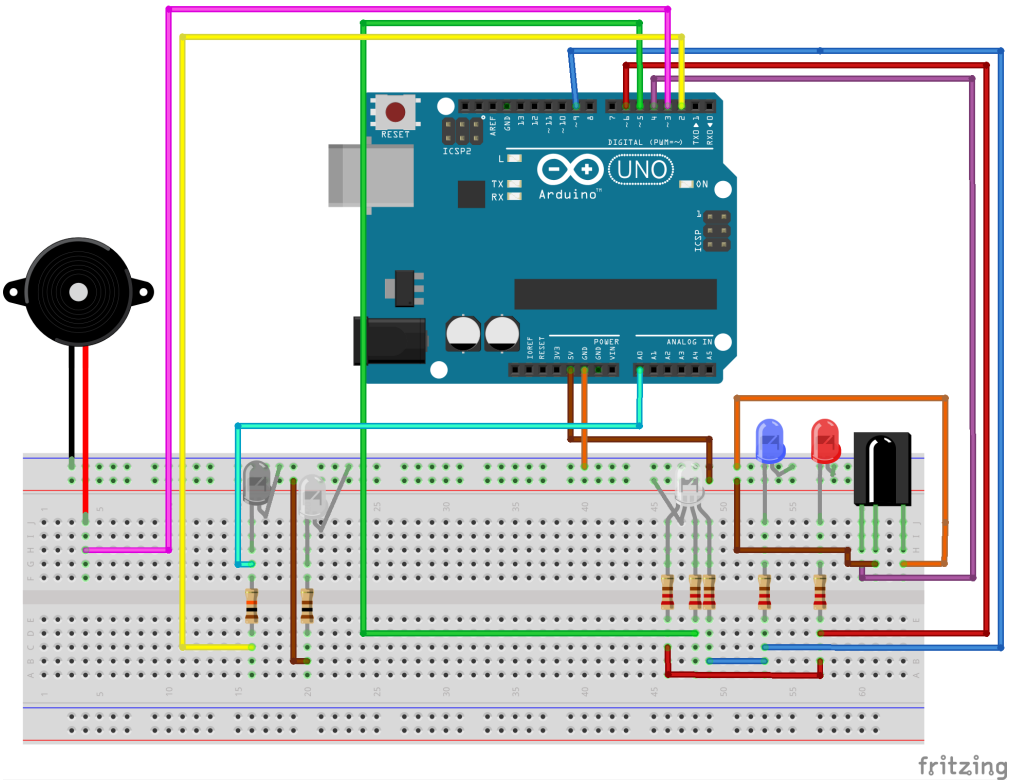

Fig. 1: Schematic of the circuit

Several scripts was developed for the Exercise 2, 3, 4, and 5 which are:

```
// Include IR Remote Library
#include <IRremote.h>
// Define sensor pin
const int RECV_PIN = 4;

// Define LED pin constants
const int redPin = 6;
const int yellowPin = 5;

// Define integer to remember toggle state
int togglestate = 0;
// Define IR Receiver and Results Objects

int brightness = 0; // how bright the LED is
int steps = 5;

IRrecv irrecv(RECV_PIN);
decode_results results ;

void setup(){
  irrecv.enableIRIn(); // Enable the IR Receiver
  pinMode(yellowPin, OUTPUT); // Set LED pins as Outputs
  pinMode(redPin, OUTPUT);
}

void loop(){
  if ( irrecv.decode(&results)){
    switch(results.value){
      case 0x6BC6597B:
        togglestate=0;
        while(togglestate <3){
          analogWrite(redPin, brightness) ;
          brightness = brightness + steps;
          if (brightness == 0 || brightness == 255) {
            steps = -steps;
            togglestate++;
          }
          delay(30);
        }
        analogWrite(redPin,LOW);
        break;
      case 0x735B797E:
        togglestate=0;
        while(togglestate <3){
          analogWrite(yellowPin, brightness) ;
          brightness = brightness + steps;
          if (brightness == 0 || brightness == 255) {
            steps = -steps;
            togglestate++;
          }
          delay(30);
        }
        analogWrite(yellowPin,LOW);
        break;
    }
    irrecv.resume();
  }
}
```

Listing 1: Exercise 2 - Control LED's using Unused IR Remote keys

```
// Include IR Remote Library
#include <IRremote.h>
// Define sensor pin
const int RECV_PIN = 4;
// Define LED pin constants

const int redPin = 6;
const int greenPin = 5;
const int bluePin = 9;

// Define integer to remember toggle state
int togglestate = 0;
// Define IR Receiver and Results Objects

int brightness = 0; // how bright the LED is
int steps = 5;

IRrecv irrecv(RECV_PIN);
decode_results results ;

void setup(){
  irrecv.enableIRIn(); // Enable the IR Receiver
  pinMode(greenPin, OUTPUT); // Set LED pins as Outputs
  pinMode(redPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  analogWrite(greenPin,255);
  analogWrite(bluePin,255);
  analogWrite(redPin, 255) ;
}

void loop(){
  if (irrecv.decode(&results)){
    switch(results.value){
      case 0x1EC81DBF:
        togglestate=0;
        while(togglestate <3){
          analogWrite(greenPin,255);
          analogWrite(bluePin,255);
          analogWrite(redPin, brightness) ;
          brightness = brightness + steps;
          if (brightness == 0 || brightness == 255) {
            steps = -steps;
            togglestate++;
          }
          delay(30);
        }
        analogWrite(redPin,255);
        break;
      case 0x450753D6:
        togglestate=0;
        while(togglestate <3){
          analogWrite(bluePin,255);
          analogWrite(redPin,255);
          analogWrite(greenPin, brightness) ;
          brightness = brightness + steps;
          if (brightness == 0 || brightness == 255) {
            steps = -steps;
            togglestate++;
          }
          delay(30);
        }
```

```
        analogWrite(greenPin,255);
        break;
      case 0xBA0F4EDF:
        togglestate=0;
        while(togglestate <3){
          analogWrite(redPin,255);
          analogWrite(greenPin,255);
          analogWrite(bluePin, brightness) ;
          brightness = brightness + steps;
          if (brightness == 0 || brightness == 255) {
            steps = -steps;
            togglestate++;
          }
          delay(30);
        }
        analogWrite(bluePin,255);
        break;
    }
    irrecv.resume();
  }
}
```

Listing 2: Exercise 3 - Design and remote control of a RGB LED

```
// Include IR Remote Library
#include <IRremote.h>
// Define sensor pin
const int RECV_PIN = 4;
// Define LED pin constants
const int redPin = 6;
// Define integer to remember toggle state
int togglestate = 0;
// Define IR Receiver and Results Objects
IRrecv irrecv(RECV_PIN);
decode_results results ;

void setup(){
  irrecv.enableIRIn(); // Enable the IR Receiver
  //pinMode(yellowPin, OUTPUT); // Set LED pins as Outputs
  pinMode(redPin, OUTPUT);
}

void loop(){
  if ( irrecv.decode(&results)){
    switch( results .value){
      case 0x6A68351E: //Update value one according to your readings
        // Toggle LED On or Off
        if ( togglestate ==0){
          digitalWrite (redPin, HIGH);
          togglestate =1;
        }
        else {
          digitalWrite (redPin, LOW);
          togglestate =0;
        }
        break;
    }
    irrecv .resume();
  }
}
```

Listing 3: Exercise 4 - Arduino Remote Emulator

```
#include <IRremote.h>

#define Buzzer  3

// Define sensor pin
const int RECV_PIN = 4;
// Define IR Receiver and Results Objects

int pd=2;
int led=6;
int ledIR=0;

int limit=850;

void setup(){
  // Serial Monitor @ 9600 baud
  Serial.begin(9600);
  pinMode(Buzzer, OUTPUT);
  pinMode(led,OUTPUT);
  pinMode(pd,OUTPUT);
  digitalWrite(Buzzer,LOW);
  digitalWrite(pd,HIGH);
}

void loop(){
  Serial.println(analogRead(ledIR));
  int val = analogRead(ledIR);
  if (val<=limit){
    digitalWrite(Buzzer,HIGH);
    delay(20);
  }else if(val>limit){
    analogWrite(led,255);
    digitalWrite(Buzzer, LOW);
    delay(5000);
  }
}
```

Listing 4: Exercise 5 - Developing a simple burglar alarm

III. Results

The system works properly. The figure 2, shows the exercise 2 where the buttons 1 and 2 of the remote control were enables to control the LEDS red and blue, increasing and decreasing their brightness.

In the figures 3, 4, and 5, show the exercise 3, how the RGB LED works for the buttons 3, 4, and 5 of the remote control. In the same way, the lights increase and decrease their brightness. For the exercise 4, we enable the power button to turn on and off the led red.

Finally, the figure 6 shows the burglar alarm, where are using IR sensor LEDS for emitting and transmitting the infra-red light. Also, it is using a buzzer to emit the sound of the alarm.

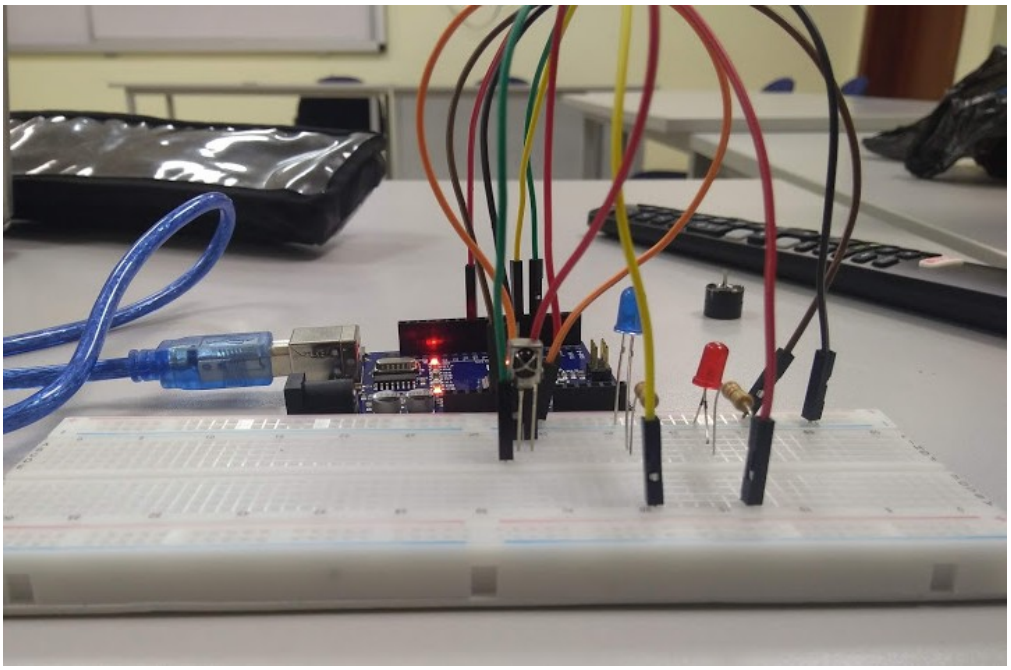

Fig. 2: Exercise 2 - Control LED's using Unused IR Remote keys

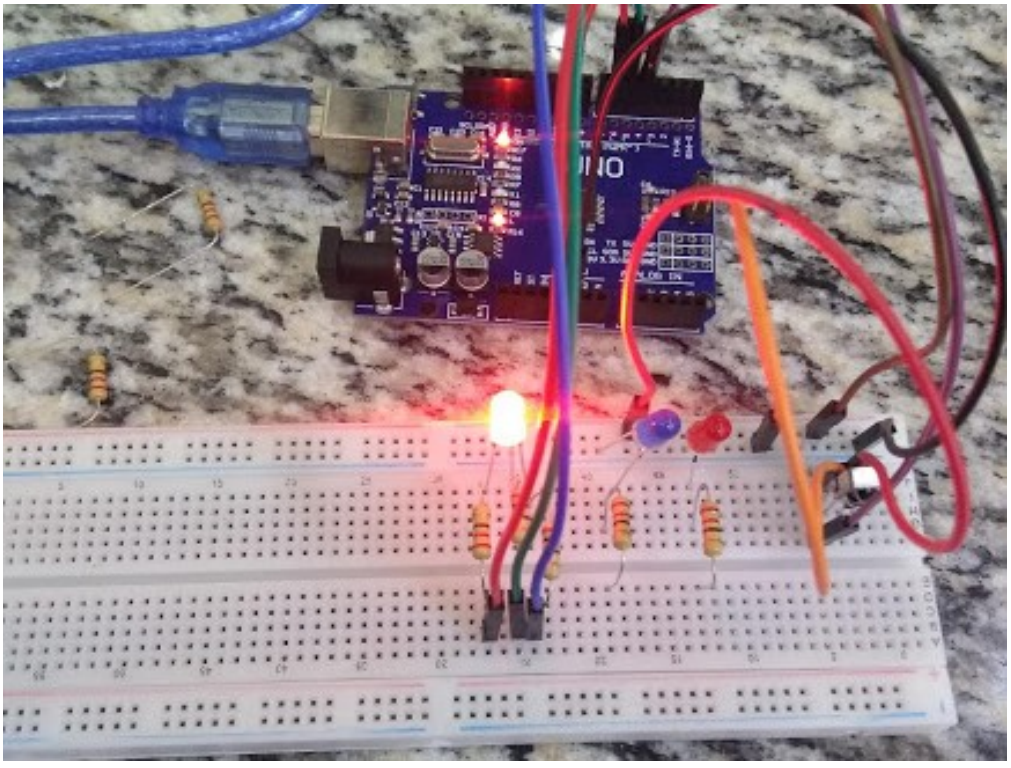

Fig. 3: Exercise 3 - Remote control of a RGB LED - Light red

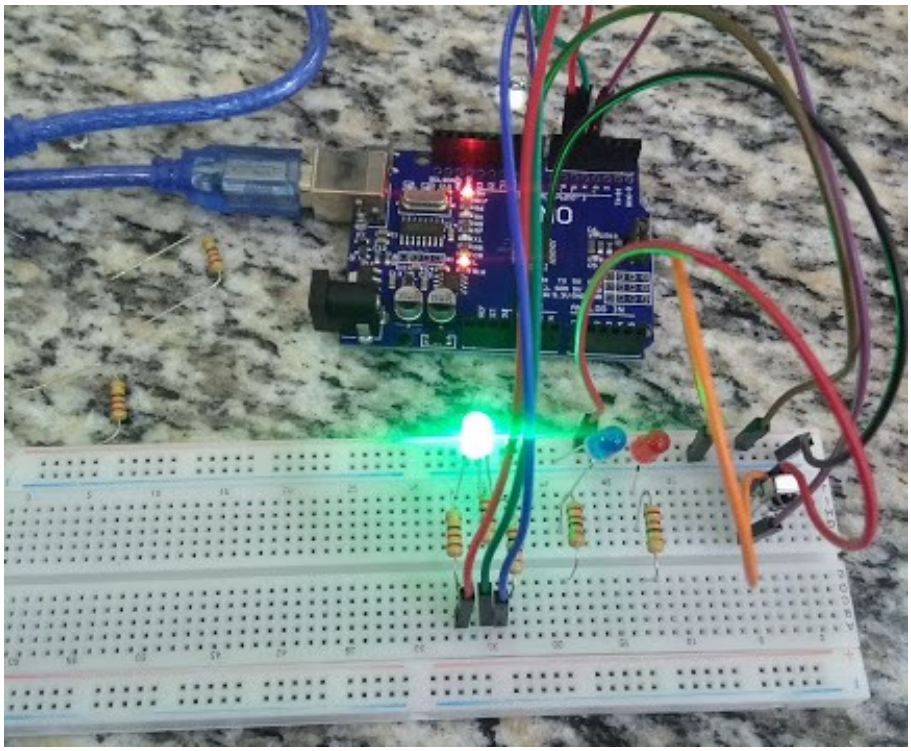

Fig. 4: Exercise 3 - Remote control of a RGB LED - Light green

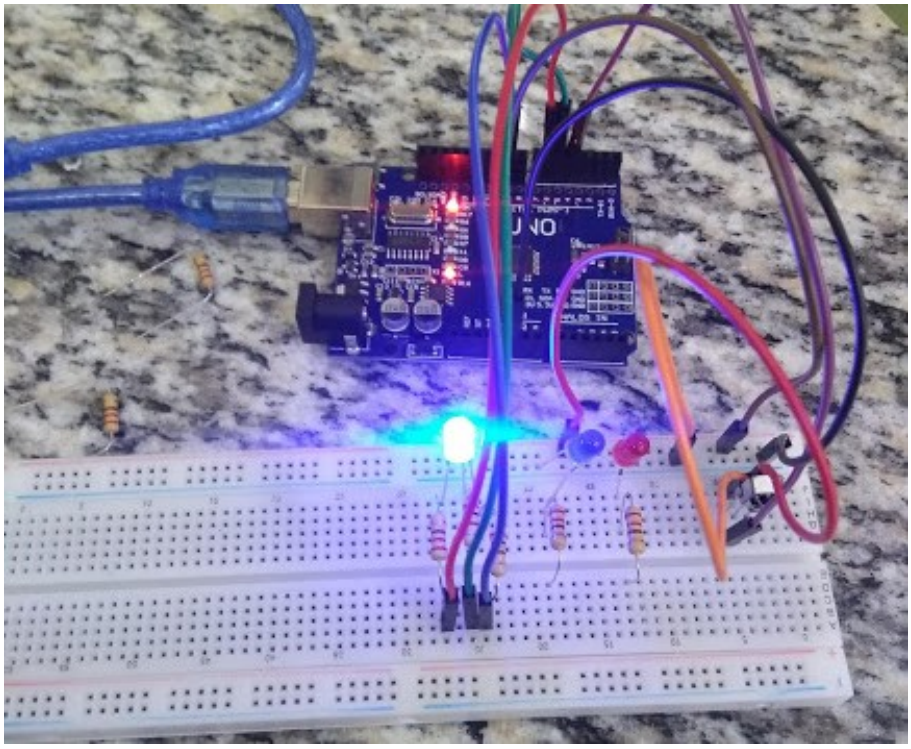

Fig. 5: Exercise 3 - Remote control of a RGB LED - Light blue

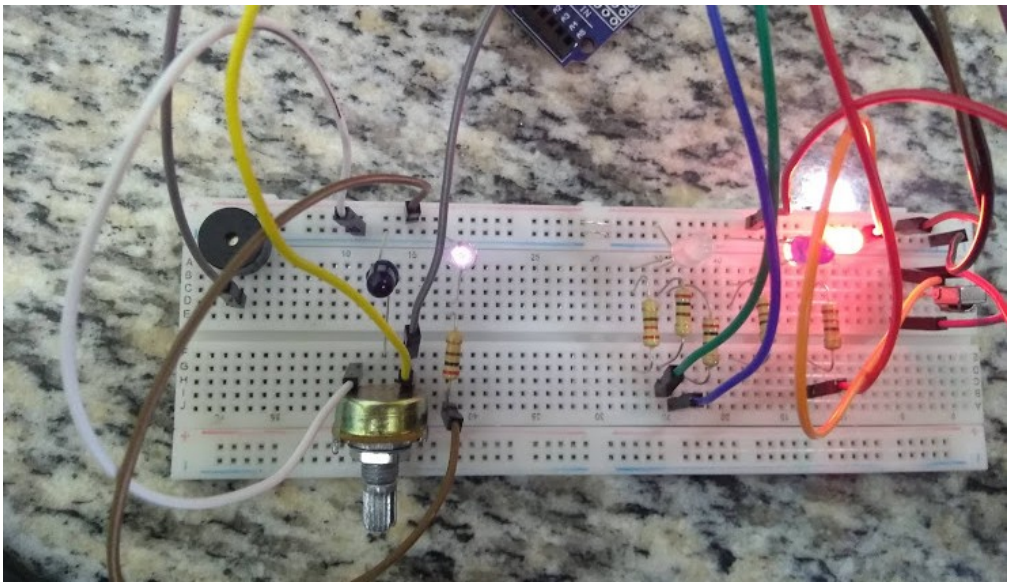

Fig. 6: Exercise 5 - Developing a simple burglar alarm

IV. Discussion

The most important part of the implementation is to be sure that the pins connected are in the correct place in both sides, Arduino Uno and Breadboard. Some problems found were that in the code the assignment of the pins of the leds were wrong. Also, not all analog pins work in the same way. For example, we had connected for an analog signal in the pin 11 and the LED emitted a fast light almost invisible. This probably happened because in this pin the analog signal is different. Furthermore, when we connected the RGB LED we thought that this is cathode common but it was anode common, hence there is not a pin for ground. In this way, the pins of RGB LED were concreted to the pins 5v, 5, 6, 9 of the arduino. In addition, we not have a 10kΩ resistor, instead we worked with a potentiometer which we had to figure out how it works.

References

[1] Iza C. (February, 2019). *LAB4: A Multi-Protocol Infrared Remote Library for the Arduino*.

[2] Lucas J. (February, 2019). *What Is Infrared?*. Retrieved from https://www.livescience.com/50260-infrared-radiation.html

[3] Random Nerd Tutorials. (February, 2019). *How do RGB LEDs work?*. Retrieved from https://randomnerdtutorials.com/electronics-basics-how-do-rgb-leds-work/

[4] Kshetrapal P. (April, 2018). *Electric Field Intensity*. Retrieved from https://www.tutorialspoint.com/electric_charges_and_fields/electric_field_intensity.asp

[5] Electronics Hub. (February, 2015). *IR SENSOR*. Retrieved from https://www.electronicshub.org/ir-sensor/