Led Band IR Remote Control

Aluculesei Marius-Andrei

Project Objective

The main objective of the project is to be able to remotely control a led band using a web application and a esp32 outfitted with a infrared led. The web application should be able to replicate the behavior of the IR remote control that came with the led band.

Project Description

The web application will send a message via a MQTT broker and the ESP32 board will receive it and send the payload to the infrared led thus transmitting the message to the led band infrared receptor. The web application will be accessible via a browser.

An infrared receiver will be used to capture all the codes that the led band remote control uses to power the leds.

Hardware Description

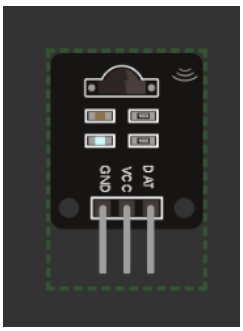
1. NodeMCU-32S-38

For this project the NodeMCU-32S-38 will be used. The board is a Wi-Fi and Bluetooth module based on the ESP32 chip.

2. IR Receiver & IR Transmitter

The IR Receiver was used to capture all the signals that the default led band IR remote control sent. This allowed us to be able to replicate those signals using our own IR Transmitter.

The IR Transmitter is a simple IR LED which we used alongside a 200Ω resistor. The transmitter will send 4 signals with the code provided via MQTT at an interval of 300 ms.

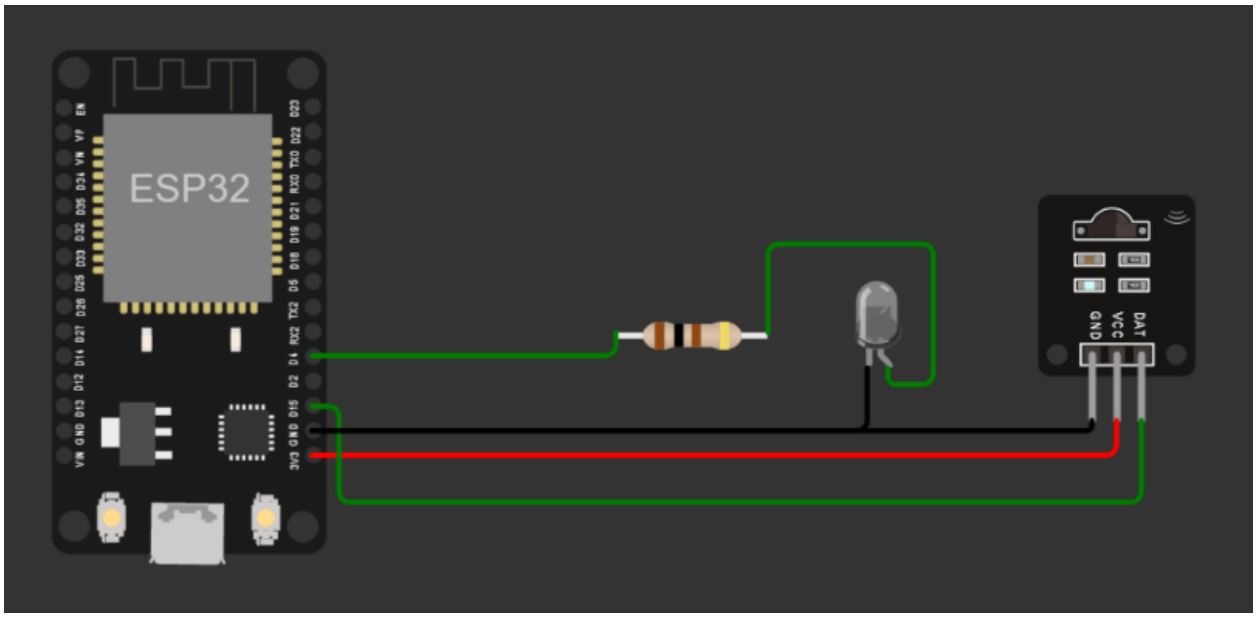


3. EverBrite Led Band

The LED band used is a simple EverBrite es500rgb interior led band. It contains a remote control and 5 meter RGB led band. The led band has attached a control unit that is able to receive IR signals as well as read sound signals.

4. Circuit diagram

Below is the circuit that was implemented. The IR receiver is hooked up to 3v3 VCC and pin 15 to be able to read IR signals. The IR transmitter is hooked in series with a 200Ω resistor on pin 4.



Software Description

1. Arduino IDE

To be able to make our application function correctly we needed to use the following libraries:

- IRremote (IR Receiver/Sender library)
- WiFi
- PubSubClient (MQTT library

To application communicates with the exterior via Wi-Fi + the MQTT connection.

a. IR Receiver

For the IR receiver we used the built-in decoding from the IRremote library so that we can determine what type of encoding the Led remote control uses. Following this we stored each code so that we will be able to use them in our application.
The encoding type we found is of type NEC.

```
if (irrecv.decode(&results)) {
  Serial.println(results.value, HEX);
  Serial.print(" - ");
  switch (results.decode_type){
      case NEC: Serial.println("NEC"); break ;
      case SONY: Serial.println("SONY"); break ;
      case RC5: Serial.println("RC5"); break ;
      case RC6: Serial.println("RC6"); break ;
      case DISH: Serial.println("DISH"); break ;
      case SHARP: Serial.println("SHARP"); break ;
      case JVC: Serial.println("JVC"); break ;
      case SAMSUNG: Serial.println("SAMSUNG"); break ;
      case LG: Serial.println("LG"); break ;
      case WHYNTER: Serial.println("WHYNTER"); break ;
      case PANASONIC: Serial.println("PNASONIC"); break ;
      case DENON: Serial.println("DENON"); break ;
    default:
      case UNKNOWN: Serial.println("UNKNOWN"); break ;
    }
  irrecv.resume(); // Receive the next value
}
```

b. IR Sender + MQTT

For the IR sender, given the fact that we discovered the IR remote uses the NEC protocol, the IRremote library provides a built-in function to allow to send IR signals using the NEC protocol.

For the MQTT part, we used the implementation provided in the laboratory to be able to subscribe to a topic from a public MQTT broker. The application waits for messages on a specific topic, and when the message arrives it reads it and converts it to an unsinged int to be able to send the message via the IRremote sendNEC function.

The sending happens 5 times with a delay of 300 ms to be sure the Led Band controller receives the message. If an invalid code is sent, the led controller simply ignores it.

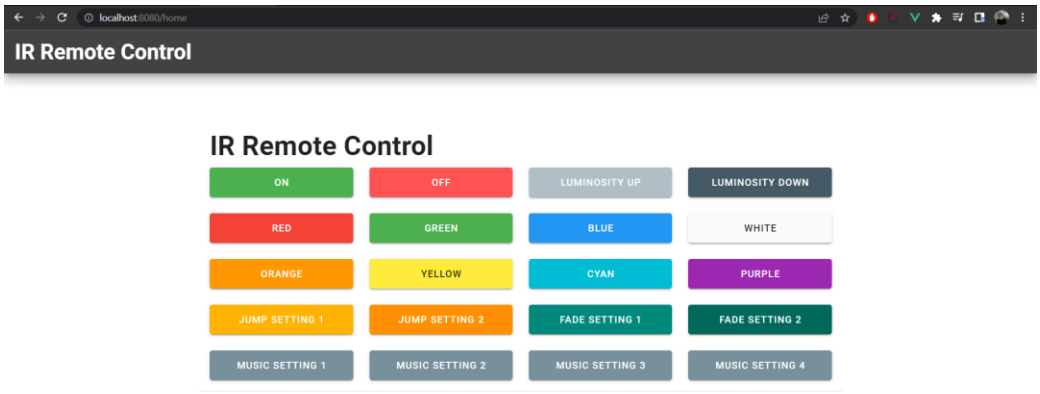
```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  String message = String(((char*)payload));
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }

  Serial.println();
  uint8_t len = 32;
  uint8_t reapeats = 5;
  for( int i = 0; i < reapeats; i ++) {
    IrSender.sendNEC(message.toInt(), len);
    delay(300);
  }

}
```

2. Web application

For the web part of this project we built a simple Vue application with all the buttons that are present on the real remote control. Each button sends a mqtt publish to the broker with the code that corresponds to the selected button. Below is a screenshot of the application



.

For the code part of the web application, it is written in typescript and there are two parts to it.

```
<v-container>
        <div class="row-lg-12 justify-center align-center">
            <v-card flat class="elevation-0 pt-6 mt-10 mb-10" height="100%">
                <v-card-title>
                    <h1>
                        IR Remote Control
                    </h1>
                </v-card-title>
                <v-card-text>
                    <v-row>
                        <v-col cols="3" v-for="(code, idx) in Codes" :key="idx">
                            <v-btn :class="idx === 'WHITE' || idx === 'YELLOW' ? '' :
'white--text'" block large :color="CodeColors[idx]" @click="sendCommand(code)">
                                {{CodeLabels[idx]}}
                            </v-btn>
                        </v-col>
                    </v-row>
                </v-card-text>
                <v-divider></v-divider>
                <v-card-actions>
                </v-card-actions>
            </v-card>
        </div>
    </v-container>
```

The user interface implements the buttons and web page that the user can use in it's browser to be able to communicate with the web server. The codes are passed internally using a state management library that creates a POST request to the webserver.

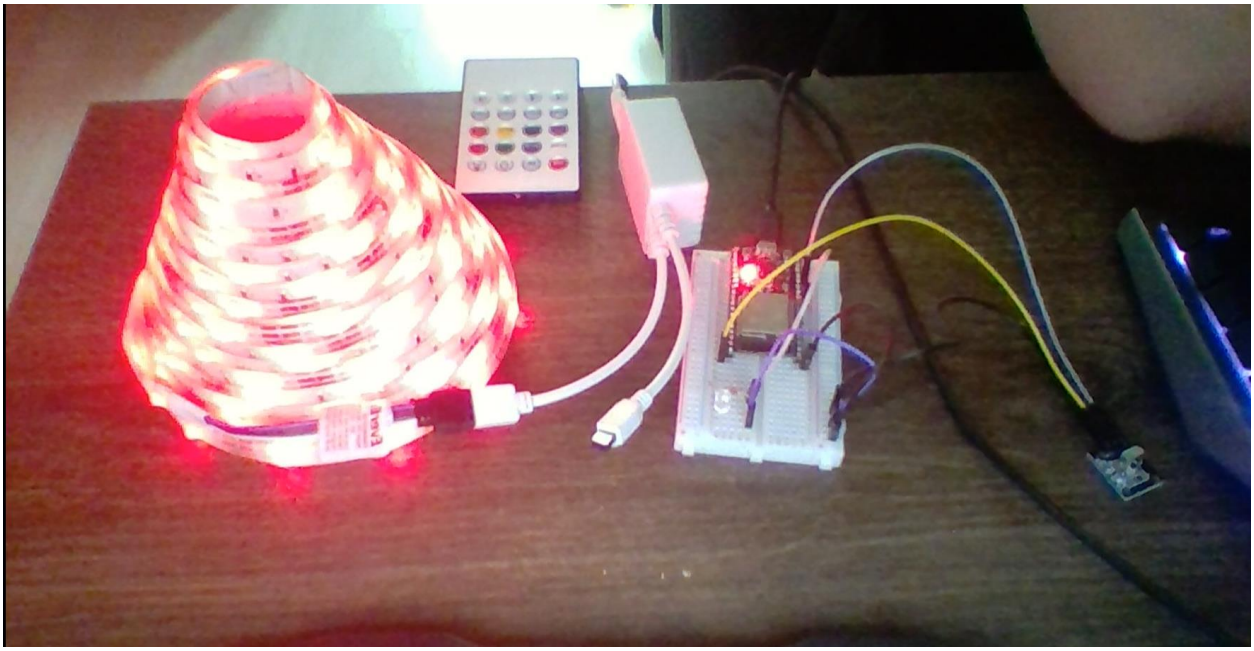
```
async sendCode (store, code: string): Promise<boolean> {
                try {
                    console.log(code);
                    const options: AxiosRequestConfig = {
                        url: `${this.state.API_URL}/send`,
                        method: "POST",
                        data: {
                            code
                        }
                    };
                    const response = await axios.request<boolean>(options);
                    if (response.data) {
                        return true;
                    } else {
                        return false;
                    }
                } catch (error) {
                    const e = error as Error;
                    console.error(e.message);
                    return false;
                }
            },
```

The server part receives input from the user interface and forwards it to the mqtt broker to facilitate the communication with the esp32 board.

```
router.post("/send",
    async (
        req: Request<{
            code: string
        }>,
        res: Response<IServerRES<boolean>>
    ) => {
        try {
            const { code } = req.body;
            const response = await mqtt.send(code);
            res.status(200).send({
                payload: response,
                err: ServerError.NO_ERROR
            });
        } catch (error) {
            const e = error as Error;
            console.error(e);
            res.sendStatus(500);
        }
    });
```

Conclusion

In conclusion the project works as intended, the web application successfully sends messages via mqtt to the ESP32 board and changes the lights of the led band according to the user's input.



The project implementation as well as a demo video can be found on [GitHub]

Bibliography

EverBrite Led Band = https://everbrightlights.com/products/led-linears/es500rgb

MQTT ESP32 = https://ocw.cs.pub.ro/courses/iothings/laboratoare/lab7

MQTT ESP32 = https://ocw.cs.pub.ro/courses/iothings/laboratoare/lab7

IR Receiver = https://www.electroniclinic.com/esp32-home-automation-using-ir-remote-and-keyes-ir-sensor/

IR Remote = http://reefwingrobotics.blogspot.com/2018/08/espressif-esp32-tutorial-ir-remote.html