WiFi Controlled ESP32 based IR blaster

**Guided By - KRS Workshop, KIIT**

By Shivam Jha

Kalinga Institute Of Industrial Technology

Jhashivam2020@gmail.com

Introduction

Today, with the help of IR blasters in our smart phones, we can easily do everything that a standard remote can do. However, not all the phones come with an IR blaster. With this project I have tried to solve that by building a WiFi controlled IR blaster.

This project uses an IR diode and an ESP-32 Micro-controller (any other cheap MCU with WiFi capabilities can also be used very easily). The IR diode is connected to the 4th pin (D4) of the MCU and MCU is connected to a WiFi network and serves a REST-API, for communication purposes. An app is also developed using Flutter which acts as the front-end (rest-api consumer) for the project.

The Rest-API takes the IR address and command as a POST request at the /remote endpoint and uses ArduinoJson library to parse the data. It then uses the IRremote library to send IR signals through the IR diode.

Literature Review:

Arduino forum as well as Google was used extensively to search about [1] [how to send IR signals through a IR diode](https://github.com/Arduino-IRremote/Arduino-IRremote" \l "readme), [2] [how to connect the ESP-32 to the WiFi](https://www.arduino.cc/en/Reference/WiFi), [3] [how to run a web server on the ESP-32](https://www.survivingwithandroid.com/esp32-rest-api-esp32-api-server/), [4] bugs that I encountered during the development cycle.

I also used [Dribble](https://dribbble.com/shots/5960398-Smart-controller-remote-app) to find a good UI interface for the Remote UI for the front-end part.

Basics Concepts/ Technologies Used:

1. **API** - An API is a set of definitions and protocols for building and integrating application software. It’s sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response).
2. **REST API -** A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.
3. Flutter - Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.
4. IR DIODE: An infrared light-emitting diode (IR LED) is a solid-state light-emitting (SSL) device that produces light in the infrared band or range of the electromagnetic radiation spectrum.
5. ESP-32: ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth
6. **Microcontrollers:** A microcontroller (MCU for microcontroller unit) is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip.

PROJECT COMPONENTS:

1. **ESP-32-WROOM Microcontroller** -

CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS.

Memory: 320 KiB RAM, 448 KiB ROM.

Wireless connectivity: Wi-Fi: 802.11 b/g/n.

1. **Breadboard:**

For temporary solderless connections, breadboard is very useful

1. **Jumper Wires:** For actual connections on the breadboard
2. **IR Receiver:** For extracting the HEX codes from the TV remote
3. **A general purpose PNP Transistor:** for amplifying the IR signal produced by the IR diode
4. **VS-Code And Arduino CLI**: For Programming and uploading the code to the ESP-32 chip

# PROPOSED MODEL:

**The proposed model is as follows:**

1. A Transistor’s common pin is connected to the D4 pin of the ESP-32
2. The collector terminal is connected to the IR LED negative terminal
3. The positive terminal of the LED is connected to the 3V3 voltage output pin of the ESP-32
4. The last pin of the transistor is grounded
5. The transistor is used to amplify the current in the IR LED to increase it’s range.
6. An IR receiver is connected to the 5th pin (D5) of the ESP-32
7. The IR receiver is used to extract the remote IR hex codes and the codes are noted in an excel sheet
8. Code is written to first connect the ESP-32 to a WiFi network.
9. A rest API is hosted on the esp32, so that mobile clients can communicate to the esp-32
10. A front-end app is prepared for consuming the API using google’s Flutter framework.
11. The app then sends a POST request to the ESP-32’s rest API at the /remote endpoint with the following body:

{

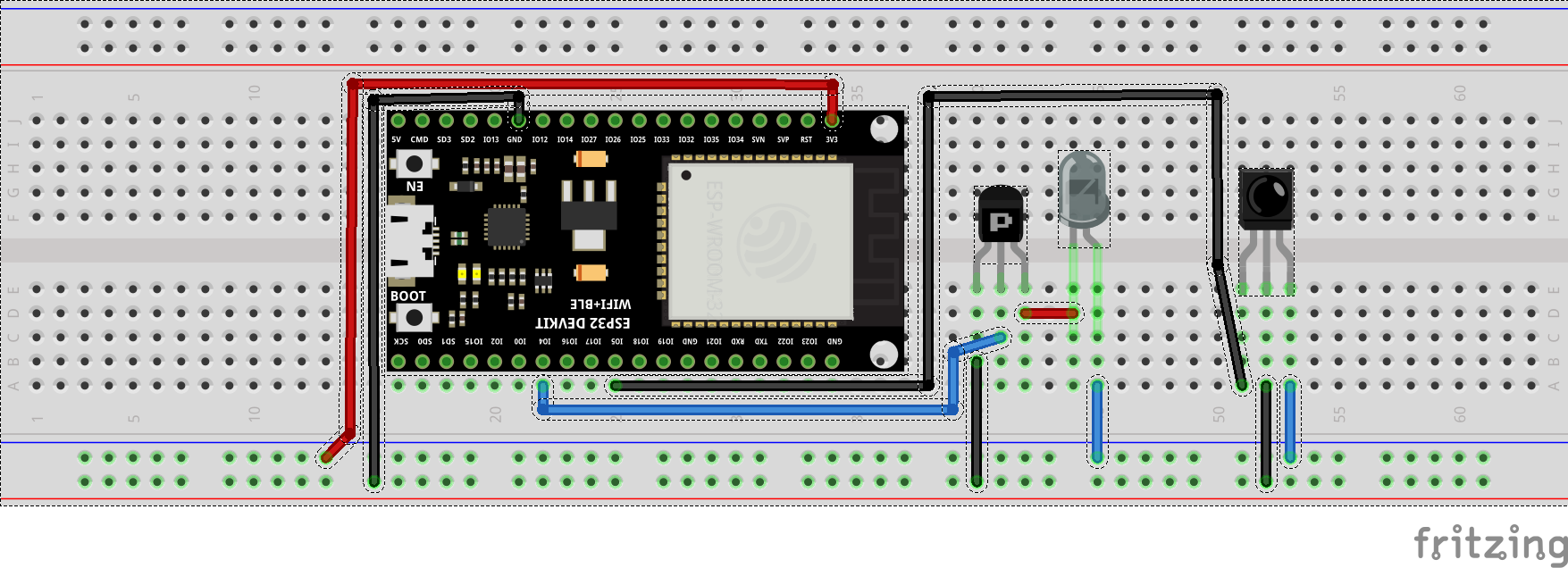
“address”: “0xFD01”,

“command”: “0xCMD”

}

Replacing the CMD with one of the values noted in step 6.

The address will also change according to the values noted in step 6.

 Fig. 1 - Proposed Circuit Connection on a breadboard

# Results:

We were able to successfully implement the proposed project. We are able to control a TV set-top-box without remote or IR blaster in our phones, over a WiFi network using an app.

Cost Analysis:

The cost of an ESP-32 is around INR. 485, however the ESP-32 is definitely an overkill for this rather simple project, which does not require the beefier specifications of the ESP-32 chip. Rather, we can use a much cheaper esp8266 and get the same results. I used the ESP-32 in this project just because that’s what I had already available in my inventory. Hence, In this analysis we will consider ESP-8266 in place of the ESP-32. Also we have used a breadboard in this project which is quite costly. In a real product, we will probably use a custom made PCB which on JLC PCB, costs $2 (INR 150/-, at the time of writing) for 10 units. That means $0.2 for 1 unit (INR 15). We will also add 5/- INR more to this amount for shipping charges.

1. The ESP-8266 costs around INR. 340/-
2. The IR diode costs around INR 20/-
3. PCB for INR 20/-
4. General purpose PNP transistor costs 1/- per piece
5. App development costs let us assume: 50/- per unit (we only need to the bear the app development costs once and after that we may need to bear just some maintenance charges)

**That gives us the total amount of INR 431/- per unit. But during bulk production, this cost will probably be much lesser.**

# Societal Impact and Future Scopes:

We can also extend our project to include an IR Receiver for the users to be able to extract IR hex codes form their own arbitrary remote devices and the the app can be extended so that it can store multiple profiles, each containing hex codes for a different remote. And the user can just switch between multiple remotes in the app.

# Conclusion:

We made a simple WiFi controlled IR remote that can be operated from our smart phones. We calculated the cost that one have to bear to build such a project for retail purposes which came out to be around 430/- including app development costs.

# References:

1. [How to send IR signals with IR diode](https://github.com/Arduino-IRremote/Arduino-IRremote)
2. [Connecting the ESP-32 to the WiFi](https://www.arduino.cc/en/Reference/WiFi)
3. [How to run a web server on the ESP-32](https://www.survivingwithandroid.com/esp32-rest-api-esp32-api-server/)
4. [Bugs](https://stackoverflow.com/questions/60940835/esp32-guru-meditation-error-core-0-paniced-loadprohibited-when-using-ble)

# Appendix A: Source code

The entire code for the project is open source and can be found at my [GitHub](https://github.com/lzzy12/wifi_remote)