İSTANBUL KÜLTÜR UNIVERSITY

Faculty of Engineering

Department of Electrical & Electronics Engineering

Individual Project

Name of the Project	**:** ESP-32 Based Smart IR Signal Repeater and Amplifier
Student Name & Surname	**: Barış Erişen**
Student ID	**: 2300007007**

1. ABSTRACT

In this project, in order to find a solution to the "remote control not detecting" problem frequently experienced in daily life, caused by weak batteries or physical obstacles, an **ESP32 microcontroller-based Infrared (IR) Signal Repeater** system has been designed and prototyped. The system captures signals from standard remote controls using an IR receiver module, digitally decodes the data packets, and re-transmits them to the target device (e.g., Television) via a high-power IR LED driven by a transistor-based amplifier circuit. As a result of the study, remote control signals (specifically using the Samsung protocol) were successfully replicated, amplified, and device control was achieved even when the direct Line-of-Sight (LoS) was blocked.

2. INTRODUCTION AND PROBLEM DEFINITION

Infrared (IR) communication is the most common method for wireless control of consumer electronics. However, IR signals rely on light waves, which means they cannot penetrate physical obstacles such as walls, cabinet doors, or computer monitors (Line-of-Sight limitation). Additionally, as battery voltage drops in standard remotes, the signal strength decreases, reducing the effective range.

The objectives of this project are:

1. To capture and analyze raw IR signals from existing remote controls.
2. To process and decode these signals using the ESP32 microcontroller.
3. To regenerate and amplify the signal electrically to extend the control range and bypass obstacles.

3. HARDWARE AND CIRCUIT DESIGN

3.1. Components

The **ESP32** was selected as the main controller due to its high processing speed and dual-core architecture. For the signal amplification stage, an NPN Bipolar Junction Transistor (BJT) was utilized.

- **Microcontroller:** ESP32
- **Input Sensor:** 38kHz IR Receiver Module (TSOP Series)
- **Output Actuator:** 940nm Infrared (IR) LED
- **Switching Element:** 2N2222 NPN Transistor
- **Protection:** 330Ω Resistor

3.2. Circuit Schematic and Working Principle

The system consists of two main stages: The **Receiver Stage** and the **Transmitter/Amplifier Stage**.

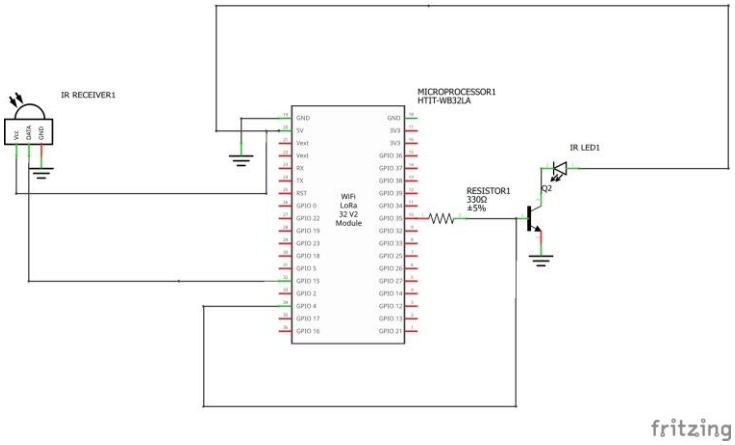


Figure 1: Circuit Schematic of the System

1. **Signal Acquisition:** The IR Receiver module is connected to `GPIO 15`. It demodulates the incoming 38kHz carrier signal and sends the pure digital data stream to the ESP32.
2. **Signal Amplification (Critical Stage):** Standard GPIO pins of a microcontroller can typically source only about 12-20mA, which is insufficient for long-range IR transmission. To solve this, a **2N2222 Transistor** is used as a switch.
    a. The control signal from `GPIO 4` is applied to the **Base** of the transistor through a **330Ω resistor**.
    b. **Why the Resistor?** This resistor limits the base current to a safe level (approx. 7-8mA) to protect the ESP32 GPIO pin from over-current damage while ensuring the transistor saturates.
    c. When the transistor is activated, it allows a higher current to flow from the **5V (VIN)** rail directly through the IR LED, maximizing the transmission power.

3.3. System Wiring List (Pinout Connections)

The following tables detail the physical connections between the ESP32 microcontroller and external components.

Table 1: IR Receiver Module Connections (Input Stage)

Component Pin	Description	Connects To (ESP32 / Power)	Cable Color (Recommended)

VCC	Power Input	**VIN (5V)**	Red
GND	Ground	**GND**	Black
DATA / OUT	Signal Output	**GPIO 15 (D15)**	Yellow/Green

Table 2: IR Transmitter & Amplifier Circuit Connections (Output Stage)

Component Pin	Description	Connects To	Cable Color
ESP32 GPIO 4	Control Signal	**Resistor (330Ω) - End A**	Orange
Resistor - End B	Current Limit	**Transistor BASE (Middle Pin)**	- (Direct Connection)
Transistor EMITTER	Ground Path	**GND**	Black
Transistor COLLECTOR	Switching Path	**IR LED Cathode (-) / Short Leg**	- (Direct Connection)
IR LED Anode (+)	Power Input	**VIN (5V)**	Red

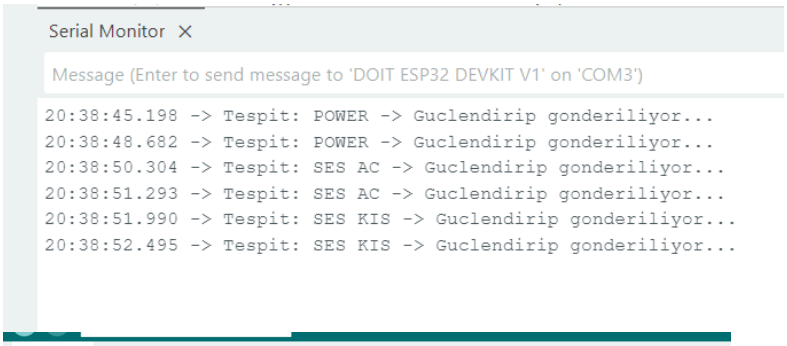
Note that - The IR LED is powered directly from the **VIN (5V)** pin instead of the 3.3V logic rail to ensure maximum infrared emission intensity and range, while the transistor handles the voltage switching.

4. SOFTWARE AND ALGORITHM

The software was developed using C++ in the Arduino IDE environment, utilizing the `IRremoteESP8266` library for protocol handling.

4.1. Algorithm Flow

1. **Initialization:** The serial communication (115200 baud) and IR receiver are started in the `setup()` function.
2. **Listening:** The `loop()` function continuously monitors for incoming IR signals.
3. **Decoding:** When a signal is detected, the protocol type (e.g., Samsung) and the Hexadecimal code (e.g., `0xE0E040BF`) are decoded.
4. **Verification & Transmission:** The received code is compared with pre-defined codes (Power, Volume Up, Volume Down). If a match is found, the `irsend` function triggers the transistor circuit to re-transmit the signal.



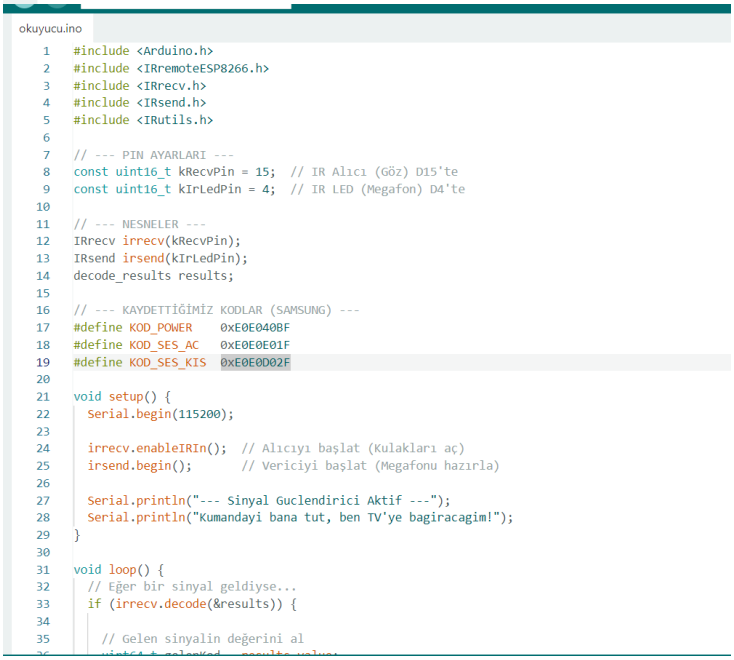


Figure 2,3 and figure 4: Software Implementation in Arduino IDE

5. PROTOTYPING AND TEST RESULTS

The system was assembled on a breadboard and subjected to real-time functional testing.

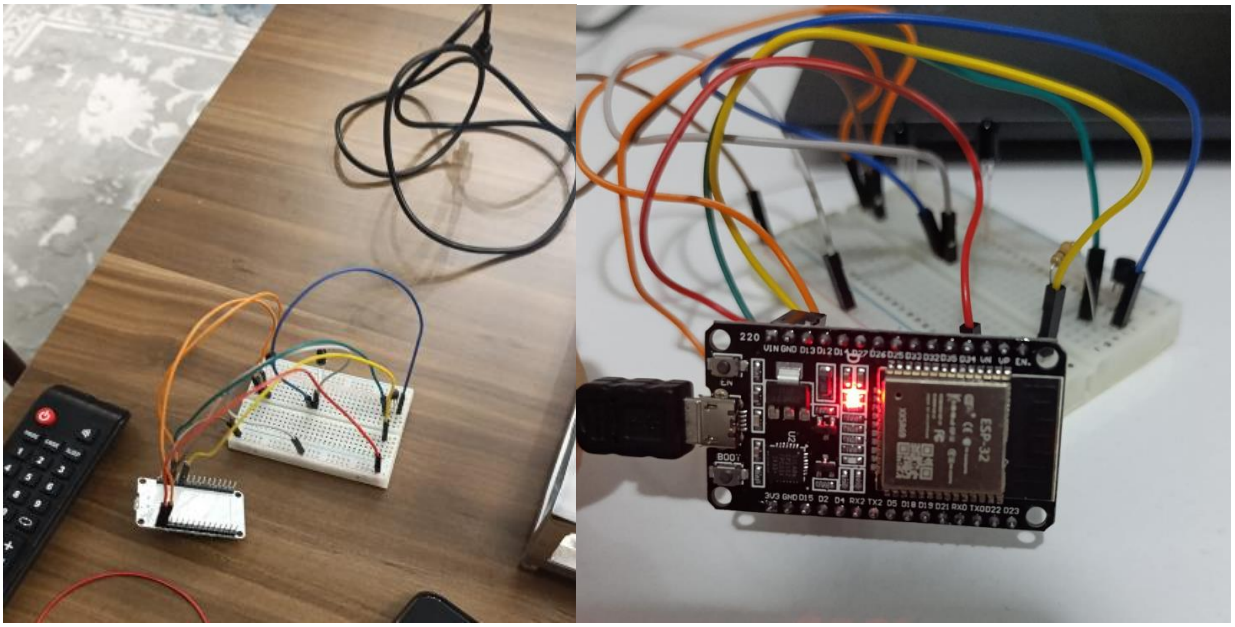


Figure 5: Hardware Prototype on Breadboard

5.1. Serial Port Analysis

During testing, the Serial Monitor was used to verify that the physical remote control signals were correctly detected and processed by the ESP32. As seen in the data logs below, the system successfully identified "Power", "Volume Up", and "Volume Down" commands and initiated the re-transmission process.

OPENING/CLOSING:

20:58:25.716 -> E0E040BF

20:58:25.716 -> Protokol: 7

VOLUME BOOST:

20:58:39.534 -> E0E0E01F

20:58:39.534 -> Protokol: 7

SOUND OFF:

20:58:59.531 -> E0E0D02F

20:58:59.531 -> Protokol: 7

okuyucu.ino

```ino
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

// Alıcı sensörü bağladığın pin (D15)
const uint16_t kRecvPin = 15;

IRrecv irrecv(kRecvPin);
decode_results results;

void setup() {
  Serial.begin(115200);
  irrecv.enableIRIn();  // Alıcıyı başlat
  Serial.println("Hazirim! Kumandayi bana dogrult ve tuslara bas...");
}

void loop() {
  if (irrecv.decode(&results)) {
    // Okunan kodu ekrana (Serial Monitor) yaz
    serialPrintUint64(results.value, HEX);
    Serial.println("");

    // Protokol tipini de görelim (NEC, Samsung vs.)
    Serial.print("Protokol: ");
    Serial.println(results.decode_type);

    irrecv.resume();  // Sonraki sinyali dinlemek için resetle
  }
  delay(100);
}
```

Figure 6: Serial Monitor codes showing that Signal Detection and Protocol Analysis

5.2. Performance Evaluation

- **Protocol Compatibility:** The system achieved 100% success rate with Samsung TV protocols.
- **Range and Obstacle Test:** The device successfully controlled the television even when placed behind a computer monitor (blocking the direct line of sight of the original remote), proving the effectiveness of the repeater logic.
- **Current Driver:** The use of the transistor driver circuit significantly improved the IR LED's emission intensity compared to direct GPIO connection.

5.3. Theoretical Connection to Signals and Systems

This project demonstrates practical applications of fundamental concepts covered in the **Signals and Systems** course, specifically regarding **Sampling, Discrete-Time Processing, and Modulation**.

- **Continuous-to-Discrete Conversion (Sampling):** The infrared signal transmitted by the remote is a continuous-time analog signal x(t). The IR receiver module demodulates this signal, and the ESP32 microcontroller processes it as a discrete-time sequence x[n]. The microcontroller measures the time duration of logical HIGH and LOW states using interrupts. For the system to accurately detect a pulse width of approximately 560uS (typical for NEC/Samsung protocols), the sampling rate (or interrupt resolution) of the processor must satisfy the **Nyquist-Shannon sampling theorem** requirements to avoid aliasing and timing jitter.
- **Modulation and Carrier Frequency:** The transmission stage of the project involves **Modulation**. The raw data bits are not sent as simple DC pulses; they are modulated onto a **38 kHz carrier frequency**. This is a form of **Pulse Width Modulation (PWM)** or On-Off Keying (OOK). The ESP32 generates this 38 kHz square wave to drive the IR LED, ensuring the signal passes through the band-pass filter of the target device's receiver.

- **System Linearity:** The repeater acts as a system H that takes an input signal x[n] processes it, and produces an output y[n]. Ideally, for the correct protocol, the system behaves as a linear time-invariant (LTI) system where $y[n] = A * x[n-tau] (where A is the amplification factor and tau is the processing delay).

6. CONCLUSION

In this project, a low-cost, programmable "Smart IR Repeater" was successfully designed and implemented. The circuit schematic adheres to engineering standards, ensuring electrical safety for the microcontroller through proper base-resistor calculation and transistor switching.

A key finding regarding protocol selectivity is as follows: The system was designed to operate specifically with the **Samsung protocol**. During tests, it was observed that while the system flawlessly repeated commands for the target device, it did not react to remote controls from different manufacturers (e.g., LG or Sony). This indicates that the device functions as a **"Digital Signal Processor"** rather than a passive optical reflector. It successfully demodulates the incoming 38kHz carrier wave, decodes the data packet, verifies the protocol type, and **only regenerates the signal if it matches the target protocol.** This selectivity is advantageous for preventing signal interference and eliminating false triggers in environments with multiple electronic devices.

Future Work:

- **Universal Compatibility:** To control devices regardless of their brand, the software algorithm can be upgraded to record "Raw Timing Data" (pulse lengths) instead of decoding specific protocols.
- **IoT Integration:** Wi-Fi capabilities of the ESP32 can be utilized to control the TV via a mobile application or voice assistants.