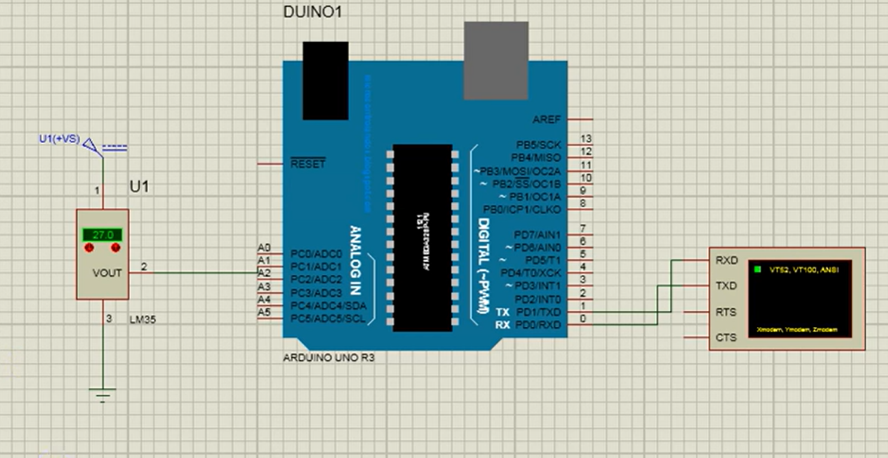
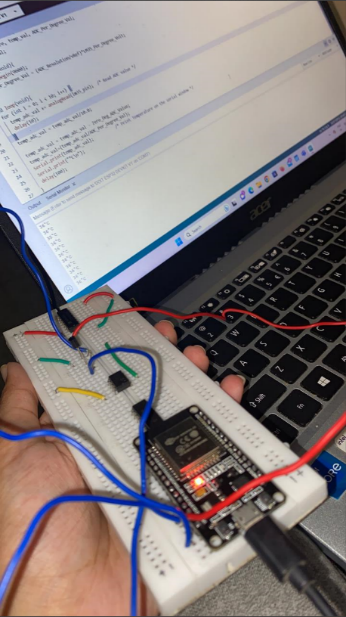
The initial circuit diagram was designed to control both the fan and light using voice commands. To achieve this, we incorporated an LM35 temperature sensor to monitor the room temperature. The main idea behind this integration was to automatically adjust the fan speed based on the room temperature. When the user issues voice commands, the system responds by turning on or off the light and regulating the fan's speed accordingly. This smart control system aimed to provide comfort and convenience by dynamically adapting the fan's operation based on the detected temperature, all of which can be controlled through voice commands.





To begin the project, we conducted a preliminary test of the LM35 temperature sensor to ensure its functionality. We successfully interfaced the sensor with the Arduino and displayed the room's temperature on the serial monitor of the Arduino IDE. This initial test confirmed that the LM35 sensor was operating correctly and providing accurate temperature readings.

Furthermore, to validate the sensor's behavior in a simulated environment, we simulated the connections in Proteus, a virtual electronics design software. The simulation yielded the expected output, reaffirming the accuracy and reliability of the LM35 sensor's performance.

With these successful tests, we laid the groundwork for implementing the temperature-based fan speed regulation and the integration of voice commands for controlling both the fan and the light in our smart home automation project.

Subsequently, we proceeded to create the circuit diagram for controlling both the light and fan in our smart home automation project. The circuit diagram was carefully crafted to incorporate the necessary components, including the LM35 temperature sensor, to enable precise temperature-based fan speed regulation. Additionally, we ensured that the circuit diagram facilitated the integration of voice commands, allowing users to control both the light and fan conveniently and intelligently.

A drawing of a circuit

Description automatically generatedA drawing of a circuit

Description automatically generated

In the final circuit design, we have utilized a total of 6 relays to achieve comprehensive control of the fan and light systems. Four of these relays are dedicated to regulating the fan speed based on the temperature readings obtained from the LM35 sensor. The remaining two relays serve the purpose of simple on and off control for the light system, operated through voice commands.

To interface the LM35 temperature sensor with the ESP32 microcontroller, we directly connected the sensor's power, ground, and digital pin to the corresponding pins on the ESP32.

To control the relays effectively, we employed BC547 transistors as switches. The relays require higher current to activate their coils, which the ESP32's GPIO pins cannot directly supply. The BC547 transistors act as interfaces between the ESP32 and the relays, allowing the microcontroller to control the relay states efficiently.

The emitter of the BC547 transistor is connected to the ground, while the base is connected to a GPIO pin of the ESP32. The collector of the transistor is connected to the relay module, which controls the flow of current to the fan or light system.

In the circuit, we have a common 220V supply connected to the relay common pin, as most household appliances operate on a 220V supply.

After finalizing the circuit diagram, we conducted thorough testing and verification using Proteus, a virtual electronics design software, to ensure the connections and functionalities were accurate and reliable. This validation process helped us confirm the circuit's performance before implementing it in the physical setup, providing confidence in the successful operation of the smart home automation system.

A computer diagram of a circuit board

Description automatically generated

Once we confirmed the circuit's functionality in Proteus, we proceeded to implement the hardware connections in the physical setup. Throughout the entire process, we exercised caution and diligently used a multimeter to check for any potential short circuits or wiring issues in the circuit.

Upon completing the entire circuit assembly, our first step was to test the functionality of each individual relay. We achieved this by uploading a simple test code to each relay and ensuring that it responded correctly, turning on and off as expected.

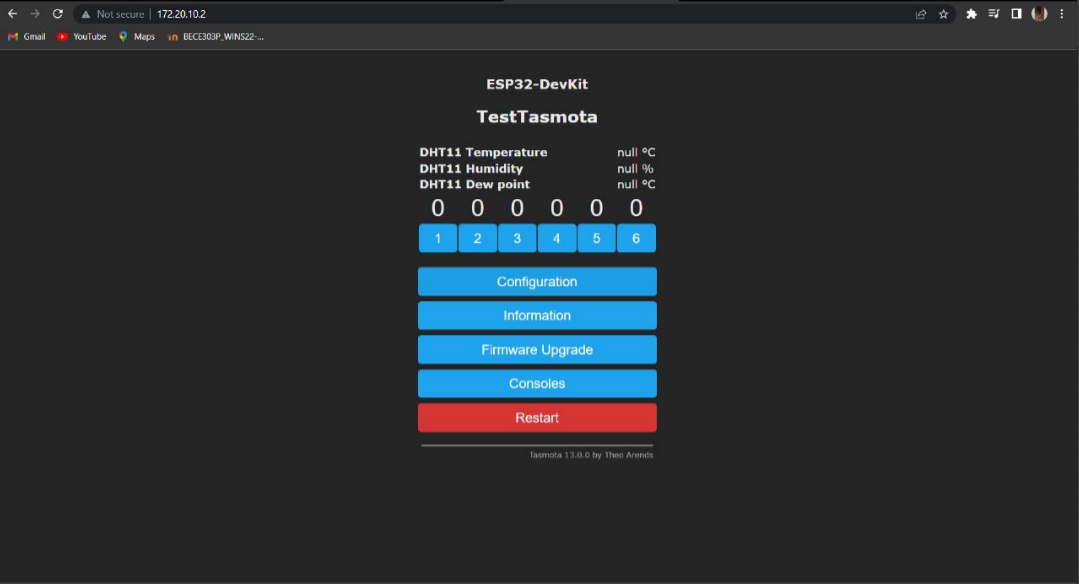
After successfully verifying the functionality of each relay, we proceeded with the integration of Alexa and Google Home into the system. This integration allowed us to control the smart home automation system using voice commands through these popular voice assistant platforms.

The project involves four main processes in its deployment. Firstly, the ESP32 communicates with the Tasmota firmware, establishing a connection that enables device control and data exchange. Next, the Tasmota firmware communicates with Home Assistant, integrating the devices into the home automation system. Subsequently, Home Assistant establishes communication with both Google Assistant and Alexa, enabling voice control and interaction with the smart devices. Lastly, users can communicate with Google Assistant and Alexa to issue voice commands and control the smart home devices seamlessly.

Tasmota is an open-source firmware designed to work with various ESP8266 and ESP32-based devices, providing them with enhanced functionalities and seamless integration with smart home systems and protocols. When Tasmota is used with an ESP32, it transforms the microcontroller into a versatile smart home automation device with capabilities to control and interact with a wide range of devices and services.

Tasmota firmware is written in C++ and is designed to be compatible with various ESP8266 and ESP32 development boards. The first step is to flash the Tasmota firmware onto the ESP32. This process involves uploading the Tasmota firmware binary using tools like Arduino IDE or PlatformIO. After flashing the firmware, the ESP32 creates a Wi-Fi access point named "tasmota-xxxx" (where "xxxx" is a unique identifier). By connecting to this access point, you can access the Tasmota web interface to configure various settings.

Through the web interface, you can set up Wi-Fi credentials, MQTT (Message Queuing Telemetry Transport) settings, device-specific parameters, and other options based on your specific use case and requirements.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

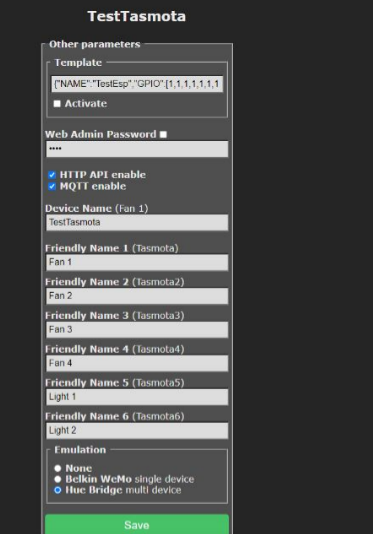
Description automatically generated Tasmota allows users to configure GPIO pins as input or output pins, providing flexibility in connecting and controlling various devices and peripherals. GPIO pins can be used to interface with sensors, switches, buttons, and LEDs, making them essential for creating smart home devices and automation projects. Tasmota supports the control of relay modules, allowing the user to switch high-power devices like lights, fans, or appliances on and off remotely or programmatically. By configuring relay pins, users can create smart switches for various home devices and appliances.

Once the pins are configured, we can then set up the MQTT protocol.

Setting up the MQTT protocol in Tasmota is a crucial step to enable communication and control of your Tasmota-powered devices through MQTT (Message Queuing Telemetry Transport). MQTT is a lightweight messaging protocol commonly used in IoT applications for efficient data exchange between devices and platforms.

The use of a Hue Bridge configuration in MQTT allows for integration between the Hue Bridge and MQTT-based smart home systems. The Hue Bridge is a hub that connects and controls Hue smart lighting devices, while MQTT is a lightweight messaging protocol used for communication between smart devices and platforms. By configuring the Hue Bridge to communicate via MQTT, you can expand its compatibility with various smart home automation systems and create a more versatile and interconnected smart home ecosystem.

With the Hue Bridge now using MQTT, it becomes compatible with various MQTT-supported smart home systems or platforms like Home Assistant, OpenHAB, Node-RED, and others.

By configuring the Hue Bridge to use MQTT, you can create a more flexible and interconnected smart home environment, combining the capabilities of the Hue ecosystem with other MQTT-enabled devices and services. This enhances the overall automation, control, and customization possibilities in your smart home.

The use of Home Assistant with ESP32 and Tasmota enables seamless integration and control of Tasmota-flashed ESP32 devices within the Home Assistant platform. Home Assistant is a popular open-source home automation platform that allows users to manage and automate a wide range of smart devices and services from a central dashboard. When combined with ESP32 devices running Tasmota firmware, it offers a powerful and flexible smart home automation solution.

A screenshot of a computer

Description automatically generated

To enable seamless communication and collaboration between Tasmota and Home Assistant, we set up MQTT integration in Home Assistant using the configuration details provided by Tasmota. This integration allows Tasmota devices to interact with Home Assistant through the MQTT broker, facilitating centralized control and coordination of smart devices within the home automation system. By establishing this connection, Tasmota and Home Assistant work together harmoniously, enabling automation, voice control, data exchange, and other advanced features that enhance the overall smart home experience.

A screenshot of a computer

Description automatically generated

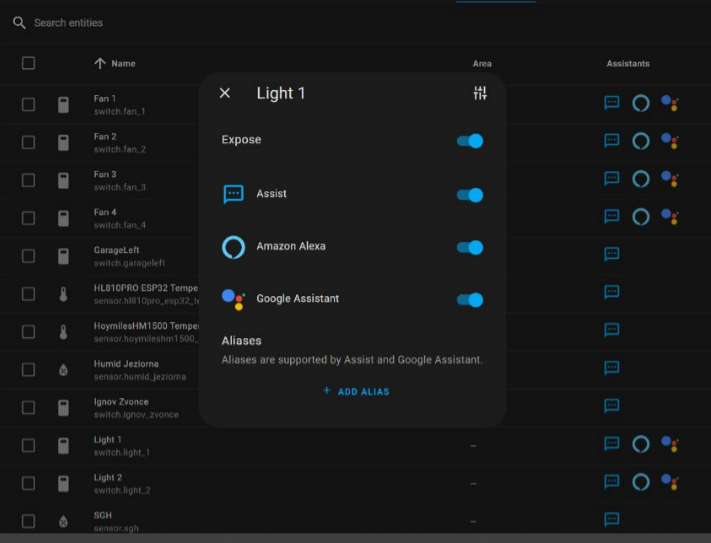
A screenshot of a computer

Description automatically generated After successfully discovering our Tasmota devices on Home Assistant, we proceed to enable voice assistant integrations. We log in to Home Assistant Cloud with our Google account and utilize the integration options to enable Alexa and Google Home. Through this process, we establish connections between Home Assistant and both voice assistants. Next, we identify the devices within Home Assistant and expose them to Alexa and Google Home. By doing so, we grant these voice assistants access to control and interact with our Tasmota-powered devices, enabling convenient voice control and management of our smart home ecosystem.

A screenshot of a computer

Description automatically generated

As depicted here, we have successfully discovered our devices within Home Assistant and seamlessly enabled voice assistants for integration.



**A screenshot of a phone

Description automatically generated** **A screenshot of a chat

Description automatically generated**

As evident from the demonstration, the devices are now visible on the Alexa app installed on the phone, allowing us to control them effortlessly using voice commands through our mobile device.