AUTODESK Instructables

DomoticX: Advanced IoT-Based Home Automation System

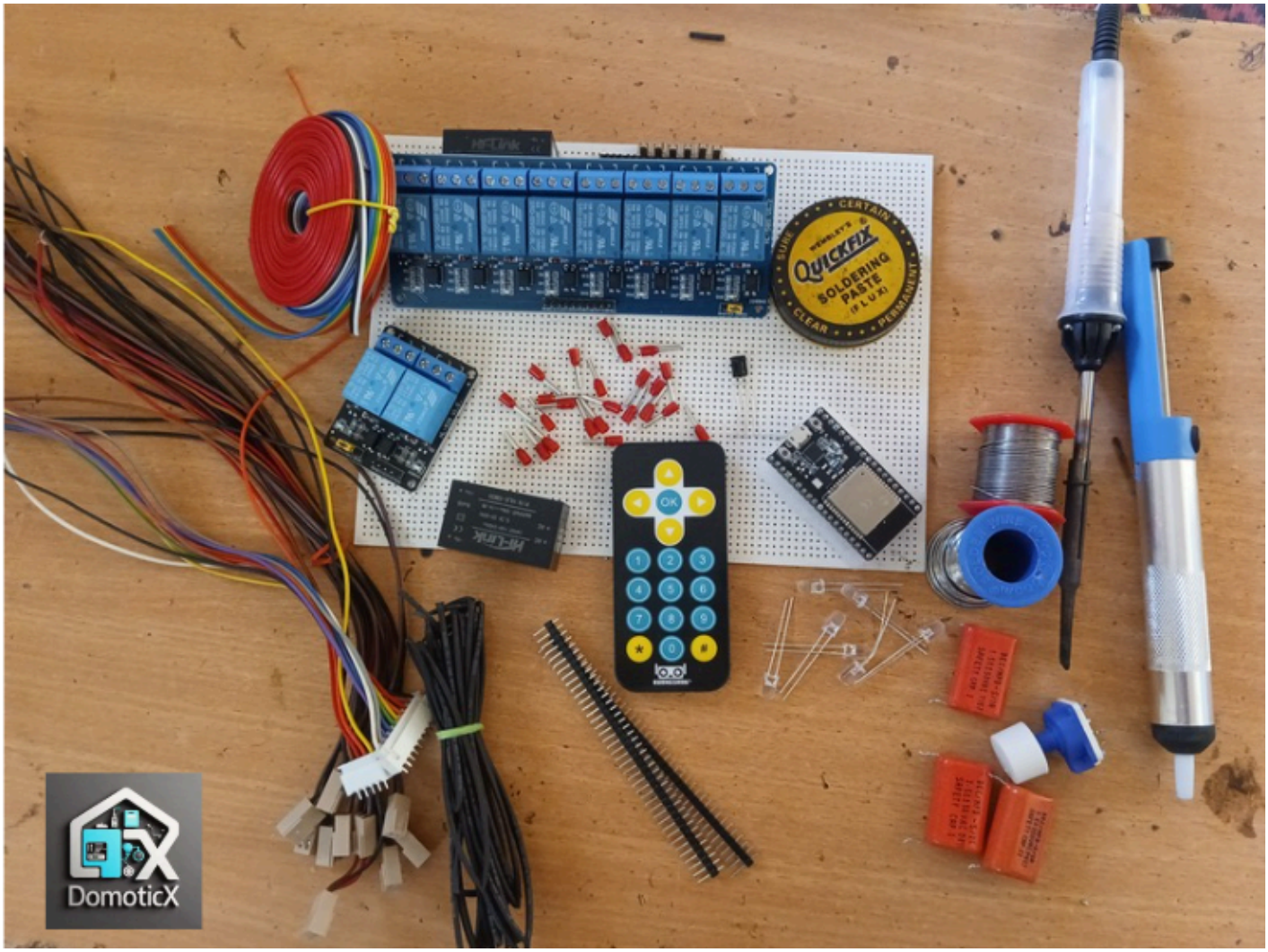
By TriMAP in CircuitsElectronics

Introduction: DomoticX: Advanced IoT-Based Home Automation System

DomoticX is an innovative home automation system that integrates advanced technologies to enable intelligent control and monitoring of household appliances. Leveraging the power of ESP32 microcontrollers, Rain-Maker, IR remotes, and relay modules, DomoticX provides a seamless experience for managing home devices in both manual and automated modes. With features such as light and bulb control, speed adjustments, and secure door lock/unlock functions, this system enhances convenience, energy efficiency, and security within the home. DomoticX uses wireless data transmission to ensure real-time operation, offering scalability for future integration with additional devices. This project represents the future of smart home ecosystems, promoting energy optimization, operational simplicity, and an enhanced user experience through IoT-driven solutions.

Download Full Code: DomoticX Code

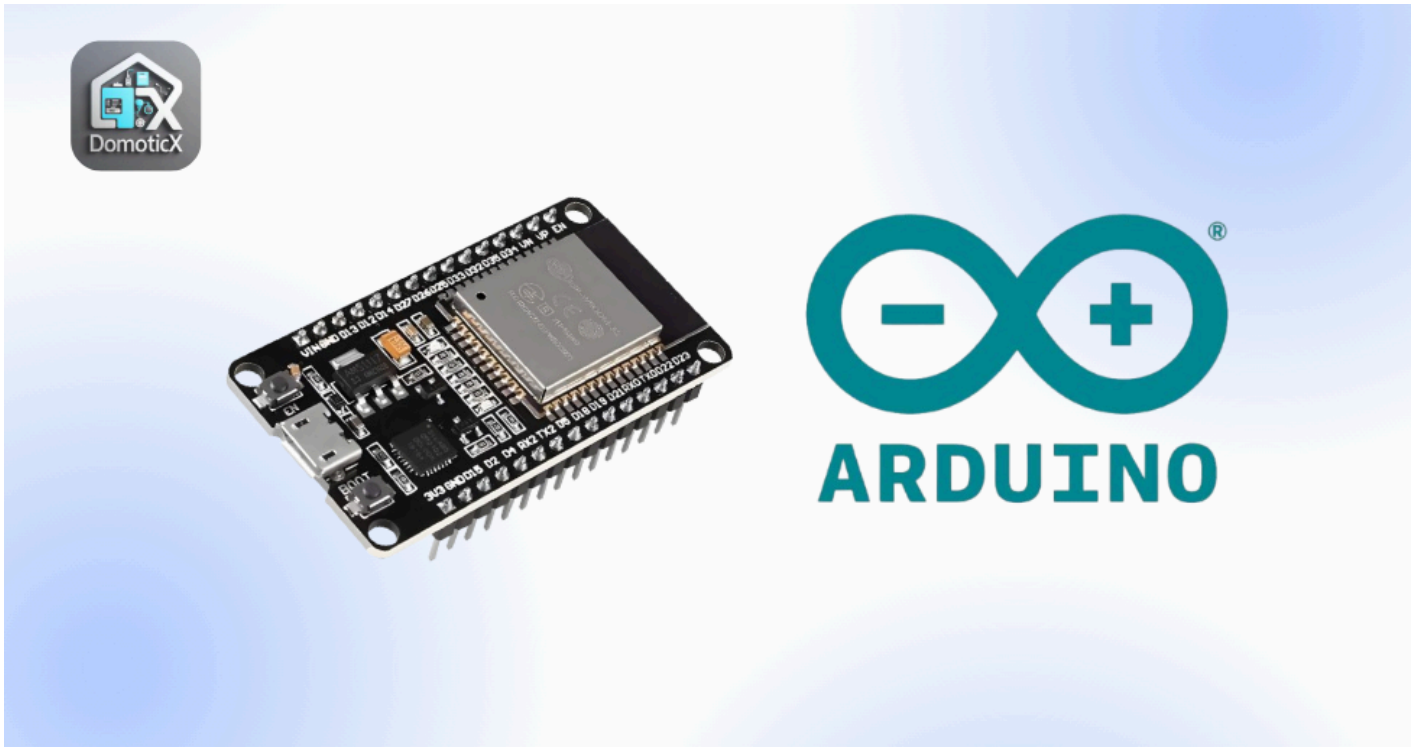
Supplies



1. **ESP32 Microcontroller** – Core processing unit for automation and IoT functionality.
2. **8-Channel Relay Module** – To control light/bulb and door operations.
3. **2-Channel Relay Module** – For speed adjustments of appliances.
4. **Hi-Link Converter (220V to 5V)** – To regulate power supply for the ESP32.
5. **TSOP IR Receiver** – For receiving signals from the IR remote.
6. **IR Remote** – For manual remote control functionality.
7. **Menu Switches and Manual Switches** – For direct, physical control of appliances.
8. **Rotary Switch 5 Speed** - For Fan Speed Control
9. **Wires and Connectors** – To connect all components.
10. **Power Supply (220V AC)** – Main power source for the system.
11. **LEDs and Bulbs** – For testing and implementation of lighting controls.
12. **Fan or Motor** – For testing speed control functionality.
13. **Screwdrivers, Soldering Kit, and Tools** – For assembling the system.

14. **PCB or Breadboard** – For organizing and connecting circuit components.

Step 1: Setting Up the ESP32 Microcontroller



1. Install Arduino IDE

Download and install the latest version of the Arduino IDE from the official Arduino website. After installation, open the IDE and navigate to **File > Preferences**. In the **Additional Board Manager URLs** field, add the following URL:

```
https://dl.espressif.com/dl/package_esp32_index.json
```

2. Add ESP32 Board to Arduino IDE

In the Arduino IDE, navigate to **Tools > Board > Boards Manager**. In the search bar, type "ESP32" and press Enter. Once the ESP32 package appears, click the **Install** button to add the ESP32 board support to your IDE. After the installation is complete, you will be able to select the ESP32 board from the **Tools > Board** menu.

3. Connect ESP32 to PC

Connect the ESP32 to your computer using a USB cable. In the Arduino IDE, go to **Tools > Port** and select the correct COM port associated with your ESP32 device. This will allow the IDE to communicate with the board for uploading code.

4. Install Required Libraries

Go to **Sketch > Include Library > Manage Libraries** and install the following libraries:

1. **Rain Maker Library** (for cloud connectivity).
2. I**Rremote Library** (for IR remote functionality).

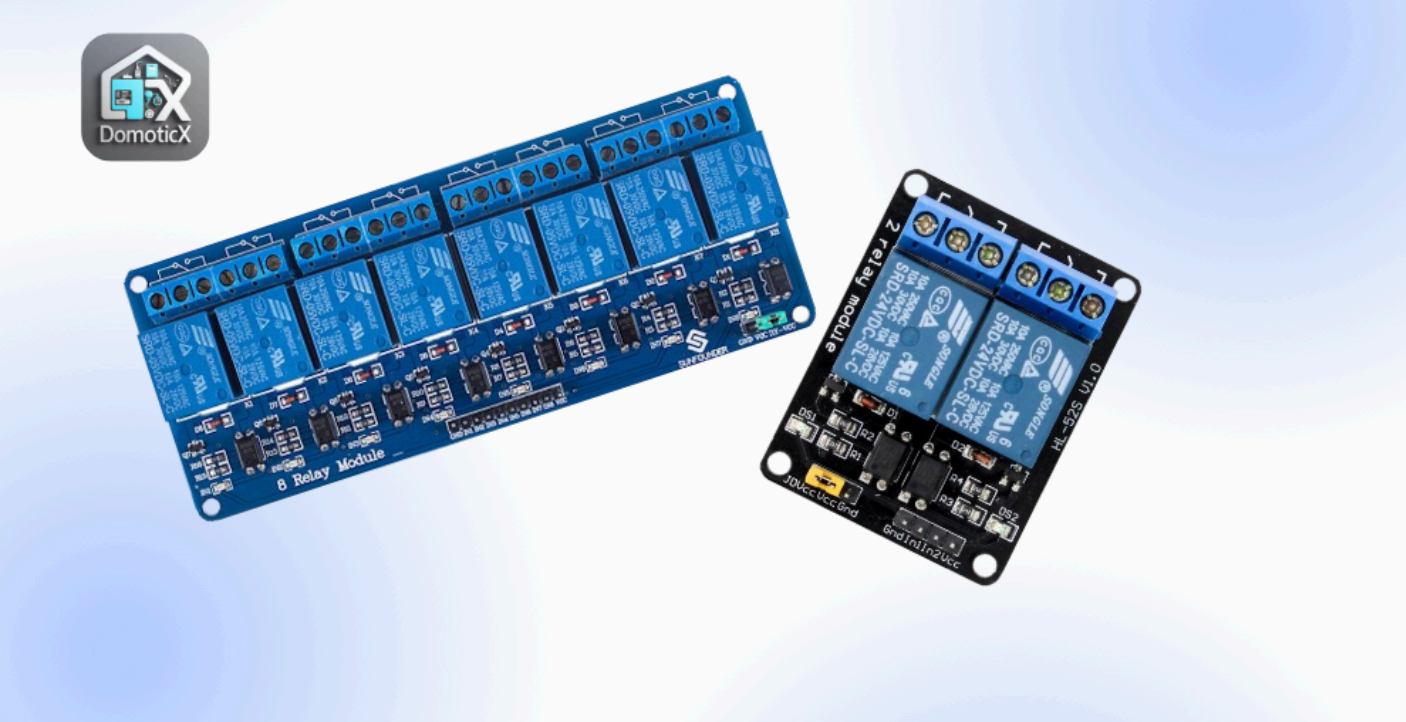
5. Test the ESP32

Upload a basic "Blink" sketch to verify the ESP32 is functioning correctly:

```
void setup() {
pinMode(2, OUTPUT);
}

void loop() {
digitalWrite(2, HIGH);
delay(1000);
digitalWrite(2, LOW);
delay(1000);
}
```

Click Upload and observe the onboard LED blinking.

Step 2: Setting Up the Relay Modules



1. Understand the Relay Module

Locate the pins on the relay module, which typically include **VCC**, **GND**, and **INx** (control pins for each relay channel). Connect **VCC** to the 5V pin and **GND** to the ground pin of your ESP32. The **INx** pins correspond to individual relays, with each controlling a separate device such as a light, bulb, or motor.

2. Connect the Relay Module to ESP32

For the **8-Channel Relay Module**, connect the **VCC** pin to the **3.3V** pin on the ESP32 and the **GND** pin to the ESP32's **GND**. Then, connect each control pin (**IN1**, **IN2**, etc.) to individual GPIO pins on the ESP32, such as **GPIO22, 21, 19, 18, 5, 17, 16, 4** ensuring each relay corresponds to a specific GPIO for light, bulb, or door control.

For the **2-Channel Relay Module**, use the same process by connecting **VCC** to **3.3V**, **GND** to **GND**, and the **INx** control pins to separate GPIOs, such as **GPIO2** and **GPIO15**, for speed control operations.

3. Wire the Appliances to the Relay Module

To connect appliances to the relay module, wire the **Common (COM)** terminal of each relay to one side of the appliance's circuit. Connect the **Normally Open (NO)** terminal to the power source for the appliance, ensuring the circuit completes when the relay is activated. For high-power appliances like lights running on 220V AC, use an external power source and exercise caution, ensuring all connections are secure and insulated to avoid electrical hazards.

4. Write and Upload Control Code

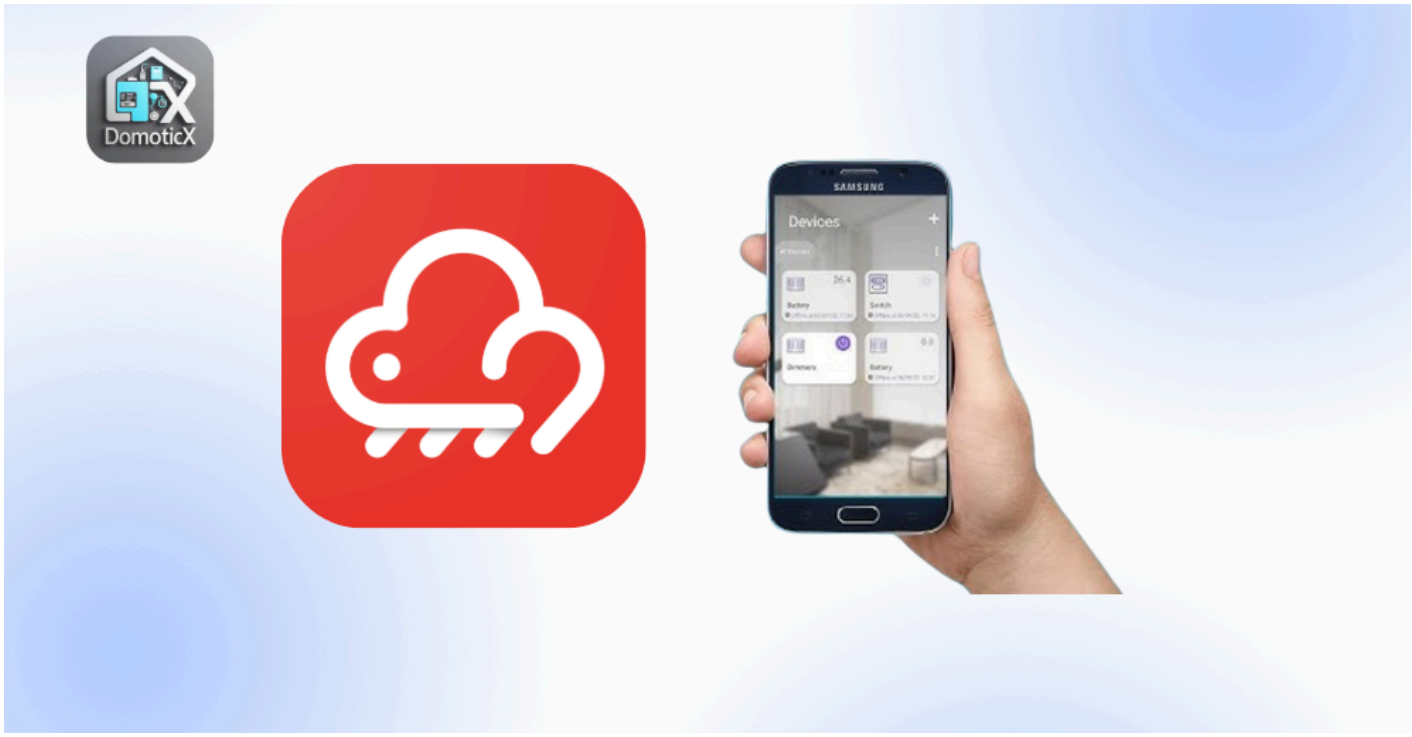
Example code to control relays:

```
#define RELAY1 23 // Assign GPIO for relay control
#define RELAY2 22

void setup() {
pinMode(RELAY1, OUTPUT);
pinMode(RELAY2, OUTPUT);
}

void loop() {
digitalWrite(RELAY1, HIGH); // Turn on relay 1
delay(1000);
digitalWrite(RELAY1, LOW); // Turn off relay 1
delay(1000);
}
```

5. Test the Relays

Power the system, then check if the relays toggle as programmed. Connect a test device like a light bulb to verify proper switching when toggling the relay controls.

Step 3: Integrating the Rain Maker Platform



1. Create a Rain Maker Account

To begin with Espressif Rain Maker, visit the official website by searching for "Espressif Rain Maker" in your browser. Navigate to the website, where you will find an option to sign up or create an account. Complete the registration process by entering the required details such as your email address, a secure password, and any additional information requested. Once registered, confirm your account through the verification link sent to your email. Afterward, download the Rain Maker app, which is available on both the Google Play Store and Apple App Store. Install the app on your smartphone, log in using the account credentials you created on the website, and start exploring its features to integrate with your ESP32-based home automation projects.

2. Set Up the ESP Rain Maker Library

Open the Arduino IDE and navigate to **Sketch > Include Library > Manage Libraries**. Search for "ESP Rain Maker" in the Library Manager. If it's not installed, click **Install** to add it. This sets up the library for your ESP32 projects.

3. Initialize the ESP32 with Rain Maker

Use the sample code from the ESP Rain Maker library to set up the ESP32. Below is a basic template:

```
#include <WiFi.h>
#include "ESP_RainMaker.h"

// Wi-Fi credentials
const char *ssid = "Your_SSID";
const char *password = "Your_PASSWORD";

// Device and parameters
RainMakerDevice device("DomoticX Device");
RainMakerParam relay1("Light Control", RAINMAKER_PARAM_TOGGLE, true);
```

```
void setup() {
Serial.begin(115200);

// Connect to Wi-Fi
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(1000);
Serial.println("Connecting to Wi-Fi...");
}
Serial.println("Wi-Fi connected!");

// Add device and parameter to Rain Maker
device.addParam(relay1);
RainMaker.addDevice(device);

// Start Rain Maker
RainMaker.start();
}

void loop() {
// Add logic to handle parameter state (e.g., relay control)
if (relay1.getVal() == true) {
// Turn ON relay
digitalWrite(23, HIGH);
} else {
// Turn OFF relay
digitalWrite(23, LOW);
}
delay(100);
}
```

4. Upload and Configure

Upload the code to your ESP32 through the Arduino IDE by selecting the correct board and port, then clicking **Upload**. Once the upload is complete, open the Serial Monitor, set the baud rate (usually 115200), and follow the displayed instructions to link your device to your Rain Maker account.

5. Control Devices via Rain Maker App

Open the Rain Maker app on your smartphone and log in with your account. Add your ESP32 device by following the app's prompts, then map parameters like Light Control to the appropriate relays. Test the setup by toggling the controls in the app and confirming that the relays respond as expected, controlling the connected appliances.

Step 4: Integrating IR Remote and TSOP IR Receiver

1. Understand the TSOP IR Receiver

The TSOP IR receiver has three pins: **VCC**, **GND**, and **OUT**. Connect **VCC** to the **3.3V** pin on the ESP32, **GND** to the **GND** pin on the ESP32, and **OUT** to a GPIO pin (e.g., **GPIO33**) on the ESP32 to receive signals from an IR remote.

2. Install the IRremote Library

Open the Arduino IDE and navigate to **Sketch > Include Library > Manage Libraries**. In the Library Manager window, search for **IRremote** and click **Install** to add the library. This will allow you to use the IR remote functionality with your ESP32.

3. Wire the TSOP IR Receiver

Connect the TSOP IR receiver as follows:

1. VCC → 3.3V on ESP32
2. GND → GND on ESP32
3. OUT → GPIO21 (or any available GPIO pin)

4. Decode the IR Remote Signals

Upload the following code to capture the IR remote's signals:

```
#include <IRremote.h>

const int RECV_PIN = 33; // Pin connected to TSOP IR receiver
```

```
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup() {
Serial.begin(115200);
irrecv.enableIRIn(); // Start the IR receiver
Serial.println("IR Receiver is ready.");
}

void loop() {
if (irrecv.decode(&results)) {
Serial.print("Received IR code: ");
Serial.println(results.value, HEX); // Print the code in HEX format
irrecv.resume(); // Receive the next signal
}
delay(100);
}
```

Open the serial monitor and press buttons on the IR remote to note the HEX codes for each button.

5. Control Relays Using the IR Remote

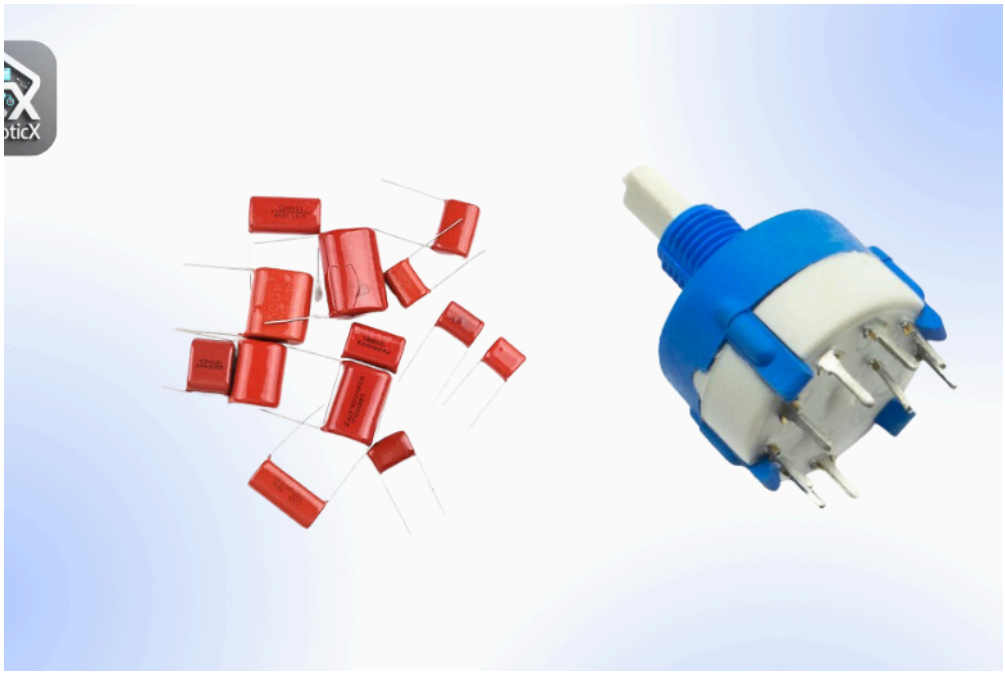
Use the captured HEX codes to control relays. Example:

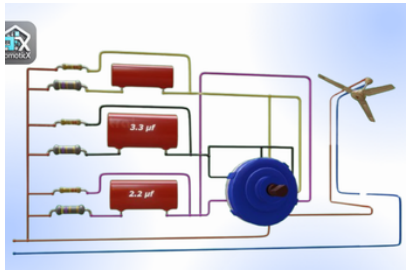
```
const int RELAY1 = 23; // GPIO pin for relay 1
const int RELAY2 = 22; // GPIO pin for relay 2

void loop() {
if (irrecv.decode(&results)) {
if (results.value == 0xFFA25D) { // Replace with your IR code
digitalWrite(RELAY1, HIGH); // Turn on relay 1
} else if (results.value == 0xFF629D) { // Replace with your IR code
digitalWrite(RELAY1, LOW); // Turn off relay 1
}
irrecv.resume();
}
delay(100);
}
```

6. Test the IR Remote Functionality

Press the assigned buttons on the IR remote and verify that the relays respond as programmed. If the relays toggle correctly when you press the corresponding buttons, this confirms that the system is set up for manual control using the IR remote. This integration adds flexibility and ease of operation to your DomoticX system, allowing you to control the connected devices via both the app and the remote.

Step 5: Integrating Fan Speed Control With Capacitor





1. Understanding the Rotary Regulator Setup

The rotary fan regulator in this project serves as a manual input device to control the speed of the fan. It has multiple positions (typically five), each corresponding to a specific speed level. Instead of directly controlling the fan speed as in traditional setups, the regulator is interfaced with an ESP32 microcontroller. The regulator sends signals to the ESP32, which processes these inputs and activates specific relays. These relays, in turn, engage a capacitor-based circuit to adjust the fan speed.

2. Preparing the Rotary Regulator for Connection

To connect the rotary regulator to the ESP32, you need to first modify the regulator's wiring. This involves:

1. **Desoldering the Components:** Carefully desolder the components and the existing connections on the rotary regulator to isolate its pins.
2. **Soldering Wires to the Regulator Pins:** Once the components are removed, solder wires to the regulator’s output pins. Each pin corresponds to a specific speed level of the regulator.
3. **Connecting to the ESP32:** After soldering, connect these wires to the GPIO pins of the ESP32. For example:
4. Pin 1 of the regulator connects to GPIO16.
5. Pin 2 connects to GPIO4, and so on

3. Wiring the Rotary Regulator

The rotary regulator now functions as an input device for the ESP32. When the regulator is rotated to a specific position, it completes the circuit for one pin at a time, sending a HIGH signal to the corresponding GPIO pin on the ESP32.

4. Signal Detection and Processing by ESP32

The ESP32 continuously monitors the GPIO pins connected to the regulator. For example:

1. If GPIO23 is HIGH, the regulator is set to Speed 1.
2. If GPIO22 is HIGH, the regulator is set to Speed 2, and so on.

The microcontroller identifies the speed level selected and triggers the corresponding relay.

5. Activating the Relay Module

Each relay in the module corresponds to a specific capacitor or combination of capacitors in the circuit. Capacitors adjust the voltage and current supplied to the fan motor, determining its speed. For instance:

1. Signal from GPIO16 activates Relay 1, engaging Capacitor A.
2. Signal from GPIO4 activates Relay 2, engaging Capacitor B.
3. Signal from GPIO2 activates Relay 3, engaging Capacitor C.

6. Adjusting Fan Speed

As the ESP32 activates the relay, it connects the selected capacitor configuration to the fan motor circuit. This adjusts the power supplied to the fan, changing its speed based on the regulator’s position.

Step 6: Adding Manual Switches for Control

1. Understand the Manual Switch Setup

Manual switches, such as push buttons or toggle switches, are used to directly control appliances, providing a physical override to the automated or remote controls. Each switch should be connected to a GPIO pin on the ESP32. When pressed or toggled, the switches will activate or deactivate the corresponding relays, controlling specific appliances like lights, doors, or fans, based on the relay connections.

2. Wiring the Manual Switches

Connect one side of each manual switch to **GND** (Ground). Then, connect the other side of the switch to an available **GPIO pin** on the ESP32, such as **GPIO17**, **GPIO16**, or any other available pin. Ensure the switches are configured as pull-down switches, meaning that when the switch is not pressed, the GPIO pin will read **LOW**, and when the switch is pressed, the pin will read **HIGH**. This configuration allows the switches to control the relays properly when toggled.

3. Update the Code to Read Switch Input

Write a code to monitor the state of the switches and control the corresponding relays. Here's an example:

```
#define RELAY1 23 // GPIO pin for relay 1
#define SWITCH1 17 // GPIO pin for switch 1

void setup() {
pinMode(RELAY1, OUTPUT);
pinMode(SWITCH1, INPUT_PULLDOWN); // Set the switch pin as input with pull-down resistor
digitalWrite(RELAY1, LOW); // Start with relay off
}

void loop() {
int switchState = digitalRead(SWITCH1); // Read the switch state
```

```
if (switchState == HIGH) {
digitalWrite(RELAY1, HIGH); // Turn on the relay if the switch is pressed
} else {
digitalWrite(RELAY1, LOW); // Turn off the relay if the switch is not pressed
}
delay(100); // Debounce delay
}
```

This code reads the state of **SWITCH1** and controls **RELAY1** based on whether the switch is pressed or not.

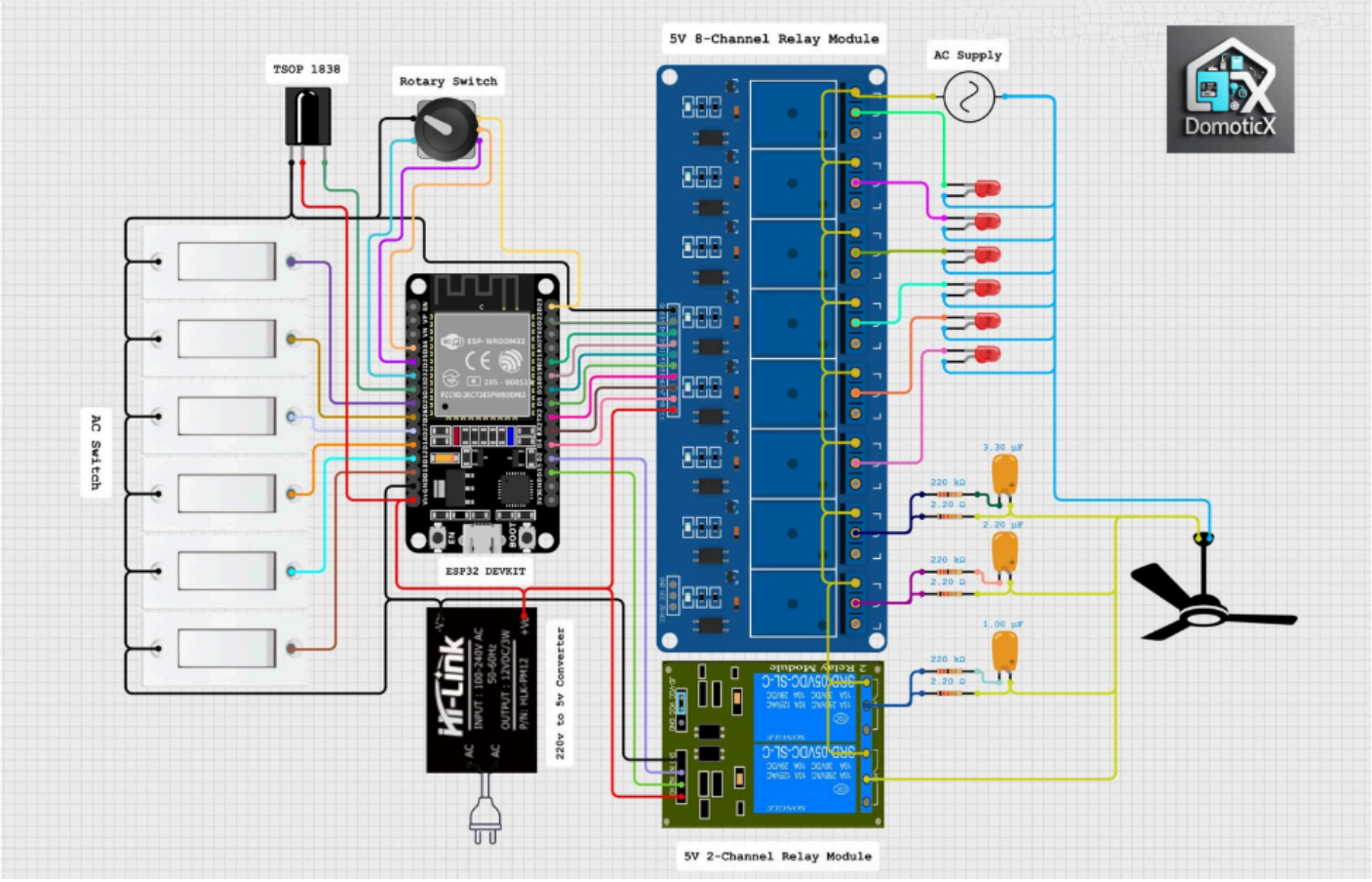
4. Test the Manual Switches

After uploading the code to the ESP32, press each manual switch to verify if the corresponding relay responds correctly by switching the connected appliance on or off. Ensure that each switch is wired to the correct GPIO pin and is controlling the appropriate relay. Double-check the connections to make sure the switches are functioning as expected, and the relays are controlling the intended appliances correctly.

5. Add More Manual Switches (Optional)

If you want to add more switches, follow the same wiring and code structure as described above. Just ensure each switch is connected to a different GPIO pin, and update the code accordingly to handle multiple switches and relays.

Step 7: Testing and Finalizing the DomoticX System



1. Check All Connections

Ensure that all components (ESP32, relay modules, IR remote, TSOP IR receiver, manual switches, servo motor/door lock, and Rain Maker integration) are correctly wired according to the previous steps. Double-check that each device is connected to the correct GPIO pins on the ESP32 and is powered by the appropriate sources. For instance, make sure the ESP32 and servo motor are powered by 5V, the relays are wired for 220V AC to control lights or motors, and the TSOP IR receiver and switches are correctly connected to the GPIO pins. Verify the connections to prevent any power issues or incorrect operation, ensuring all devices function as expected.

2. Upload Final Code to ESP32

To combine the code for relays, switches, IR remote, and Rain Maker, define GPIO pins for relays and switches, and use the IRremote library for decoding remote signals. Manual switches are detected via interrupts or polling to control relays. Integrate the Rain Maker library to enable cloud-based relay control. This unified code ensures the system responds to manual, remote, and cloud commands, keeping relay states synchronized across all control methods..

Download Full Code: DomoticX Code

3. Test Manual Switches

Test each manual switch to ensure it correctly controls the connected appliances, such as turning lights on/off, adjusting fan speeds, or locking/unlocking doors. When pressing each switch, observe whether the corresponding relay responds as expected and the appliance operates properly. Verify that the manual control flow functions smoothly without any malfunctions, and make sure the

switches are wired correctly to their respective GPIO pins on the ESP32. This will confirm that the manual controls are working as intended.

4. Test IR Remote

Use the IR remote to test if the system responds to specific button presses. Ensure that the relays trigger correctly, turning appliances on or off, and verify that the door lock (servo or electromagnetic) operates as expected when the corresponding button is pressed.

5. Test Rain Maker Cloud Control

Open the Rain Maker app and verify that you can control all appliances, including lights, fan speed, and the door lock, remotely. Ensure the app correctly toggles the states of each appliance and reflects real-time changes in the system. Additionally, test the linking and unlinking of devices within the app to confirm that the system can easily be expanded with more devices in the future. This will ensure smooth operation and scalability.

6. Test All Integration

Run tests where multiple control methods are used simultaneously to ensure proper operation. Test the system with both manual switches and the Rain Maker app controlling the same appliances to confirm that both methods work independently without interfering with each other. Next, test the IR remote and Rain Maker app together, ensuring that pressing a button on the remote doesn't disrupt the app's control, and vice versa. This will verify that activating one control method doesn't affect the functionality of the others, ensuring smooth operation across all control sources.

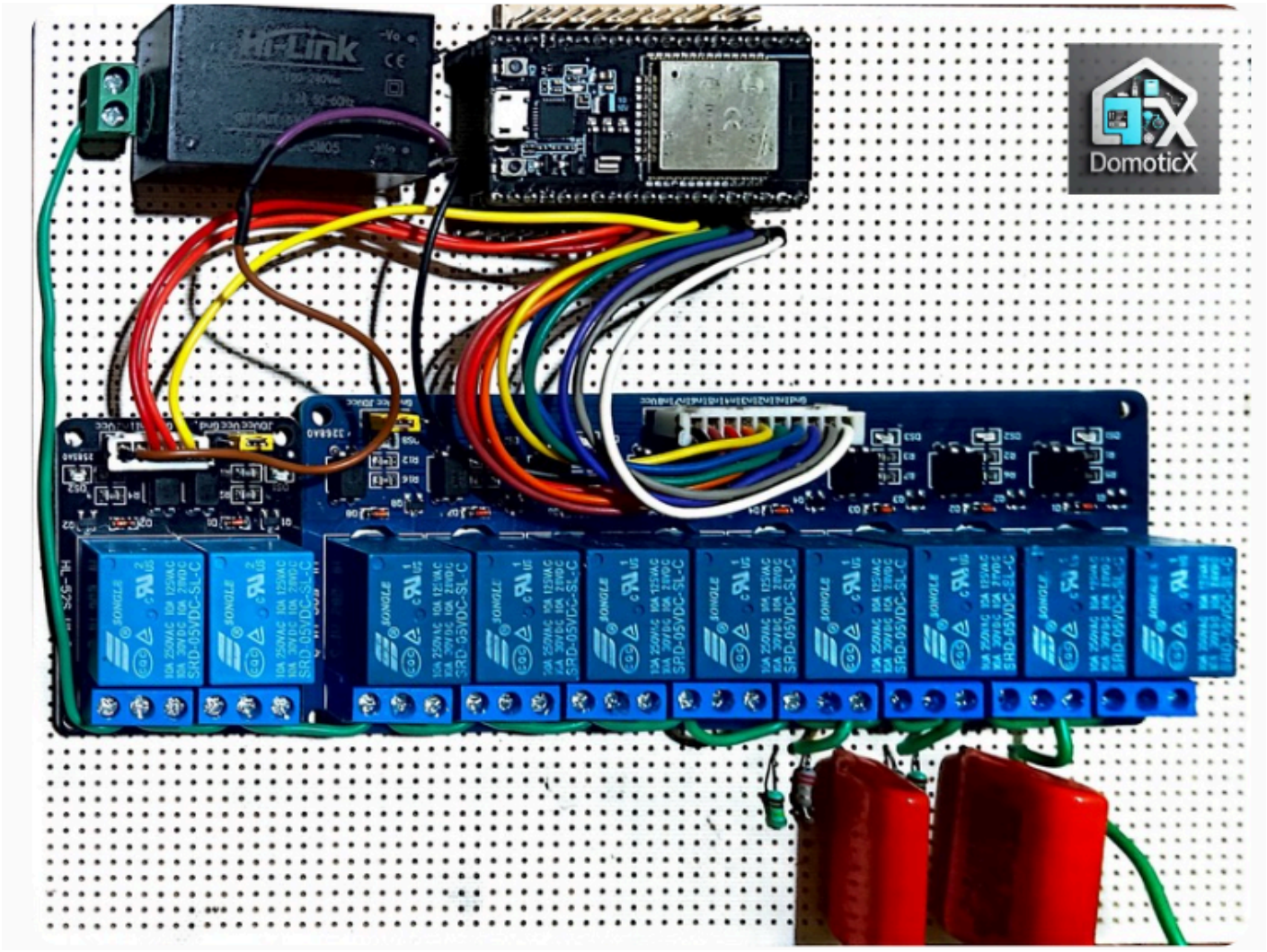
7. Troubleshoot Issues

If any part of the system doesn't function as expected, start by checking the wiring connections to ensure everything is correctly connected. Recheck the code for any errors or misconfigurations, and monitor the serial output for any error messages or warnings that might help diagnose the issue. Additionally, verify that all components, such as the IR receiver, relays, and servo motors, are operating within their specified voltage and current limits to prevent malfunctions or damage to the hardware. Properly troubleshooting these aspects will help ensure the system works as intended.

8. Secure and Enclose the System

Once all testing is successful, securely mount the ESP32 and other components in protective enclosures to safeguard them from dust, moisture, or accidental short circuits. Ensure that the components, such as relay pins and switches, are clearly labeled for easy troubleshooting in the future. This final step will make the DomoticX system fully functional, reliable, and ready for real-world use, offering multiple control options and the flexibility to expand the system with additional devices as needed.

Step 8: Summary



I have designed DomoticX to be an advanced yet easy-to-follow home automation solution. By following the setup steps, you can seamlessly control appliances like lights, fans, and doors using the ESP32 with Rain Maker, IR remotes, and manual switches. Make sure to configure the correct GPIO pins and double-check connections to avoid issues. Don't forget to enter the correct Wi-Fi credentials and authenticate your Rain Maker account for smooth integration. With proper setup, you'll successfully control your appliances both manually and wirelessly. I plan to share more exciting updates and enhancements for DomoticX soon, so stay tuned! Feel free to share your feedback or questions in the comments. (This project is developed and documented by Ashutosh Mahindrakar, Parshwa Patil, Pprathmesh Anande – Innovator in IoT-based home automation solutions).