National Textile University,

Faisalabad

Department of Computer Science

Name:	Aleeha Aiman Essha Imran Naima Naveed
Class:	Bs Cs 5th A
Registration No:	23-NTU-CS-FL-1243 23-NTU-CS-FL-1022 23-NTU-CS-FL-1032
Course Name:	Embedded IOT

Submitted To:	Sir Nasir Mehmood
Submission Date:	1 - January - 2026

Title : Automated Tea Maker

Documentation

Table of Contents

- Project Overview
- Introduction
- Problem Statement
- Project Objectives
- Key Features
- System Architecture
- Block Diagram
- Hardware Architecture
- Software Architecture
- State Machine Flow
- Hardware Components
- Circuit Connections
- Software Components
- Screenshots of Output
- Network Configuration
- Methodology
- Cloud Database Integration
- MQTT Integration
- Node RED Integration
- Issues and Solutions
- Results
- Conclusion
- Future Scope

Project Overview

Introduction

The Automated Tea Maker is an IoT-powered device built with ESP32 microcontroller that automates the entire tea brewing process. The system controls motors for liquid dispensing, servos for ingredient addition, and heating elements for boiling, all through a beautiful web interface with real-time status updates and monitoring.

Problem Statement

In today's fast-paced world, people often struggle to find time for simple tasks like making tea. Traditional tea preparation requires:

- Constant monitoring and attention
- Manual measurement of ingredients (water, milk, tea, sugar)
- Precise timing for boiling and brewing
- Risk of overcooking or undercooking
- Wastage of ingredients due to improper measurements
- Inability to multitask while making tea

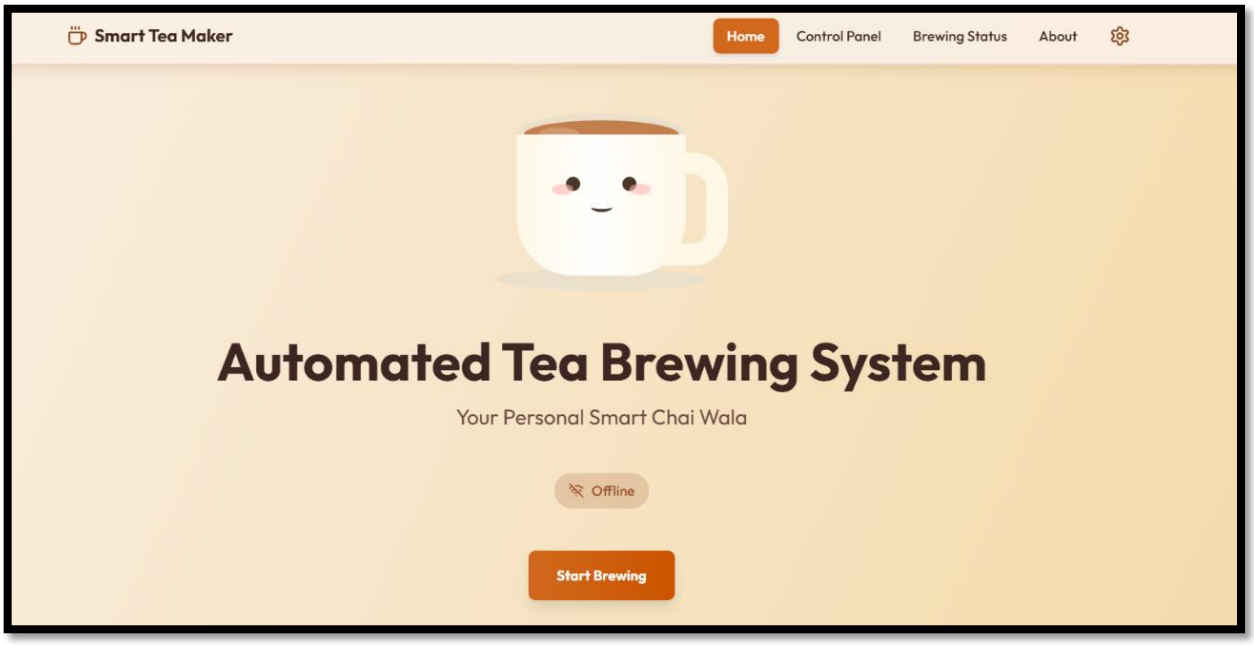
Project Objectives

The main objectives of this project are:

1. **Automation of Tea Making Process**: Develop a fully automated system that can prepare tea without manual intervention, from ingredient dispensing to heating and completion.
2. **Precise Ingredient Control**: Implement accurate measurement and dispensing mechanisms for water, milk, tea leaves, and sugar to ensure consistent taste and quality.
3. **IoT Integration**: Create a web-based interface accessible from any device on the local network, allowing users to control and monitor the tea-making process remotely.
4. **User Customization**: Provide flexibility for users to customize their tea preferences including strength (tea spoons), sweetness (sugar amount), and milk ratio according to their taste.
5. **Time Efficiency**: Reduce the time and effort required to make tea by automating the entire process, allowing users to multitask effectively.
6. **Recipe Management**: Enable users to save their favorite recipes and access quick presets for different tea types (strong brew, mild tea, classic, sugar-free).
7. **Cost-Effective Solution**: Design an affordable automated tea maker using readily available components like ESP32, relays, servos, and pumps.

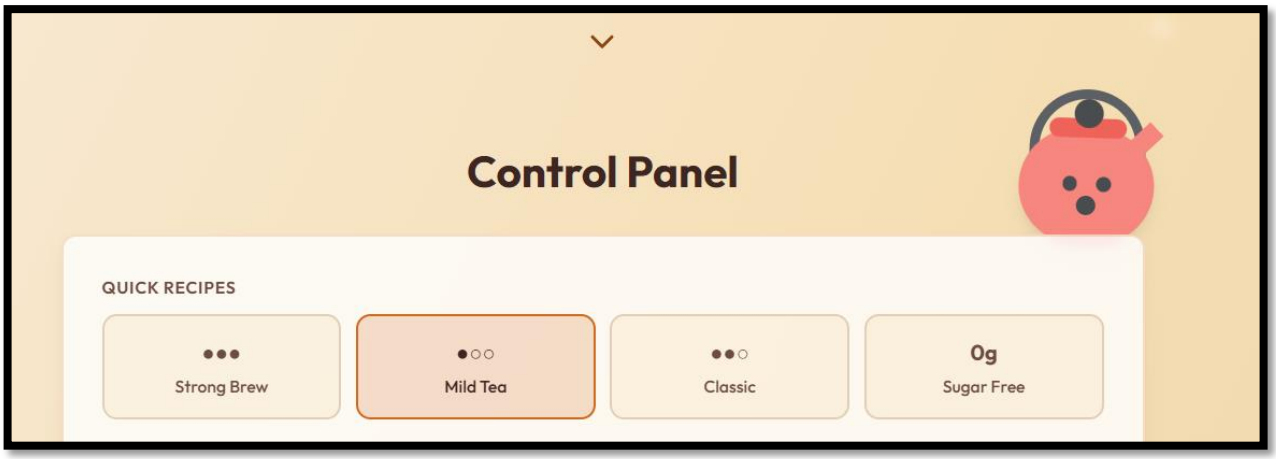
Key Features

- **Web-based Control Interface**: Intuitive UI accessible from any device on the network

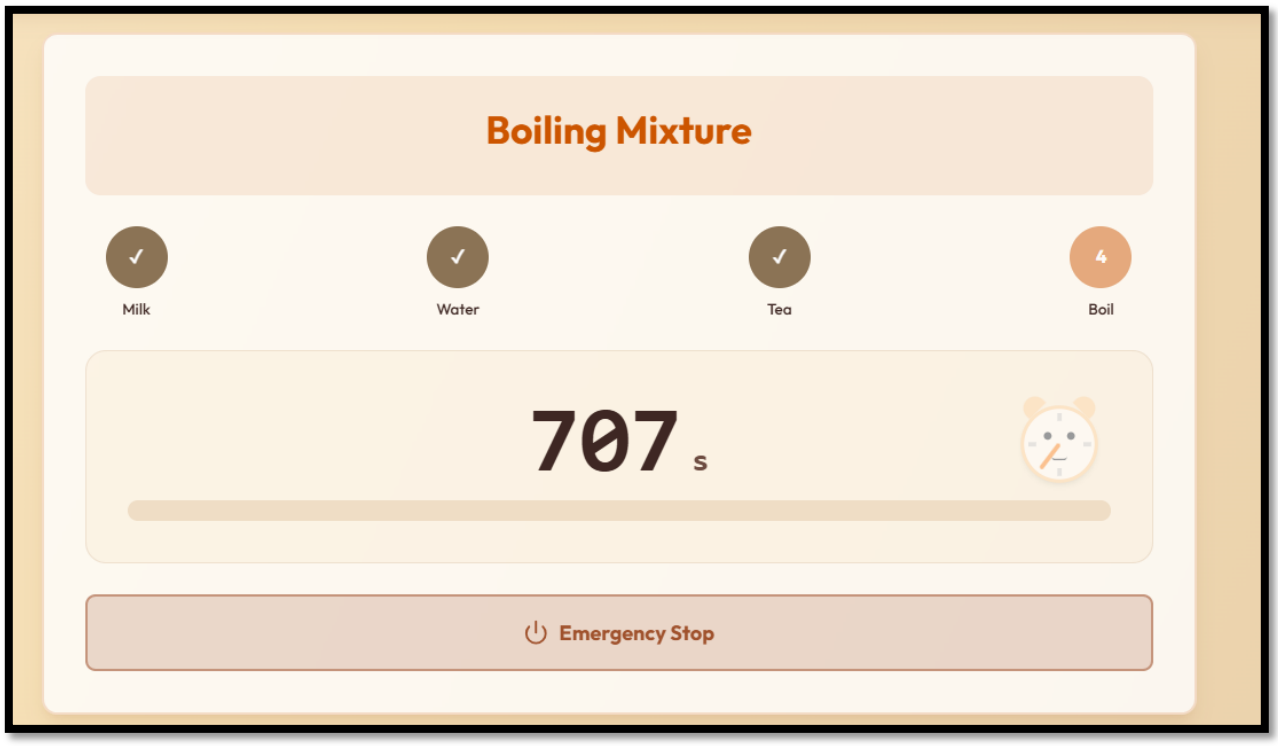


- **Sequential Automation**: Automated brewing sequence.
- **Real-time Monitoring**: Live brewing status and remaining time display.

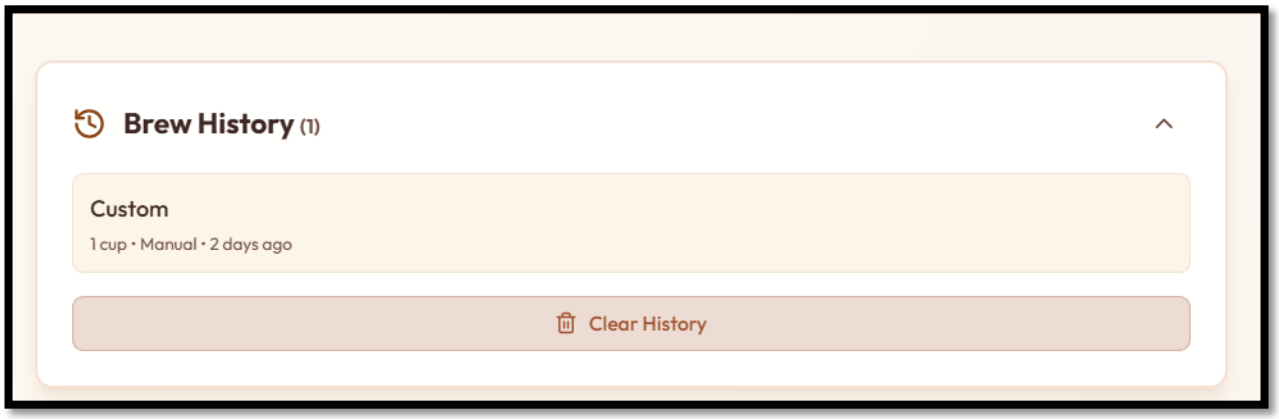
- **Quick Recipes**: Pre-configured recipes for different tea strengths.



- **Safety Features**: Automatic shutoff and manual emergency stop. So, if you are running late you want to stop the system it stops immediately.

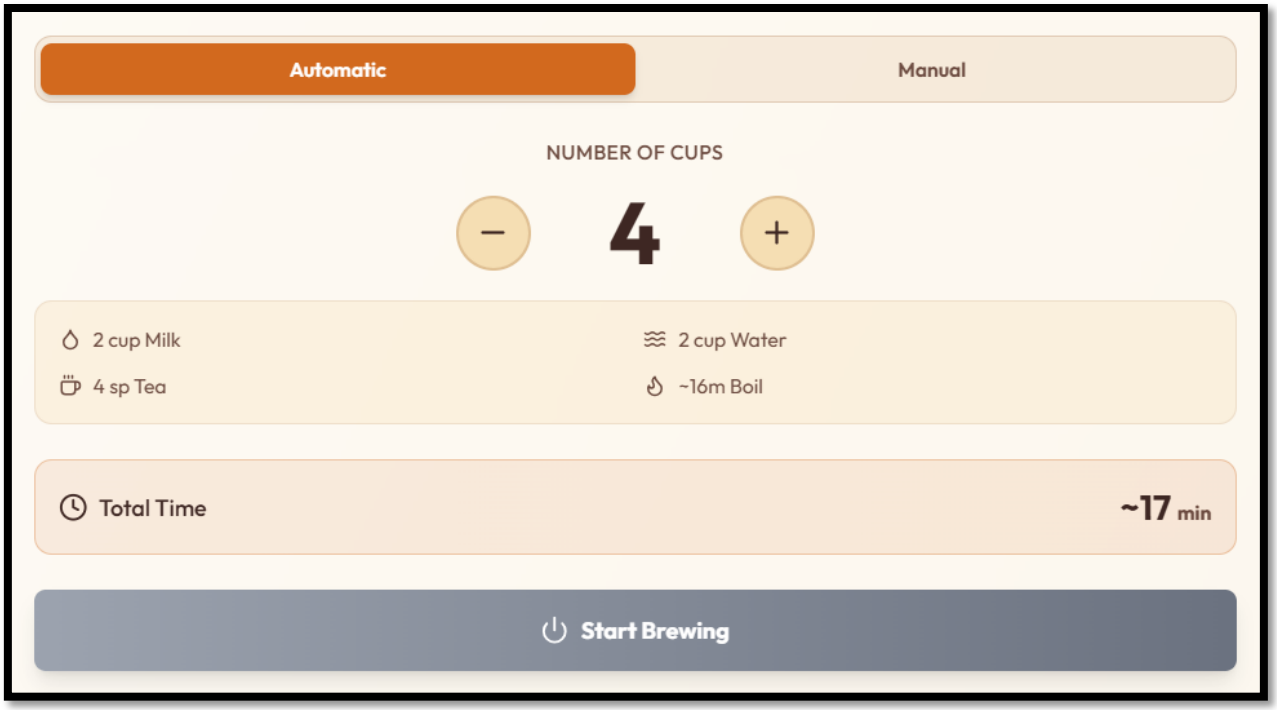


- **Brewing History**: Showing the recipes that we tried earlier.



- **Recipe Saving**: Store favorite recipes for quick access. If you love a particular recipe add it to favorites so you can use it next time and save time.

- **Automatic Selection :** It has already saved automatic system selection. You can also select the number of cups you want.

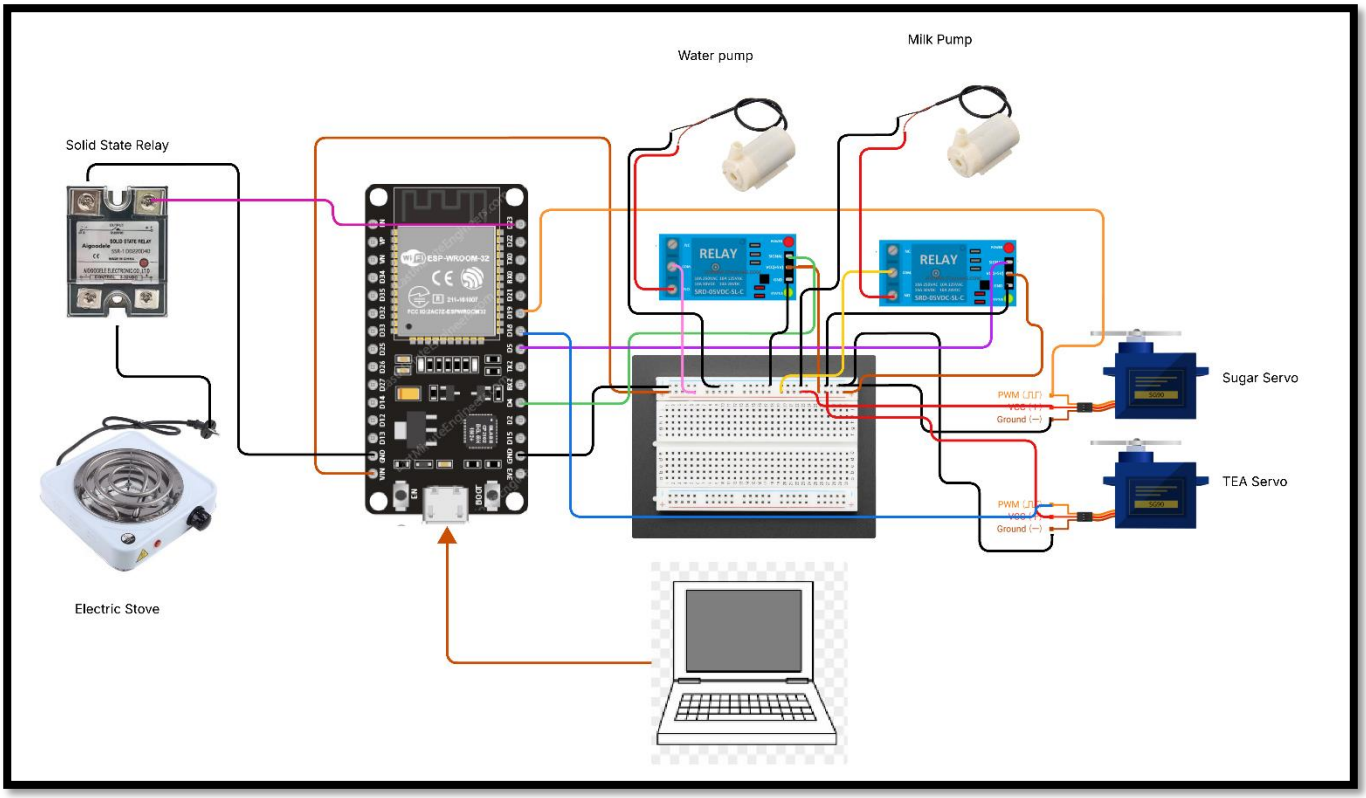


- **Manual Selection:** You can manually customize tea

  - ❖ With Sugar
  - ❖ Without Sugar
  - ❖ With Milk or not
  - ❖ With Water or not
  - ❖ The number of Teaspoons you want i.e. strong or mild tea.

System Architecture

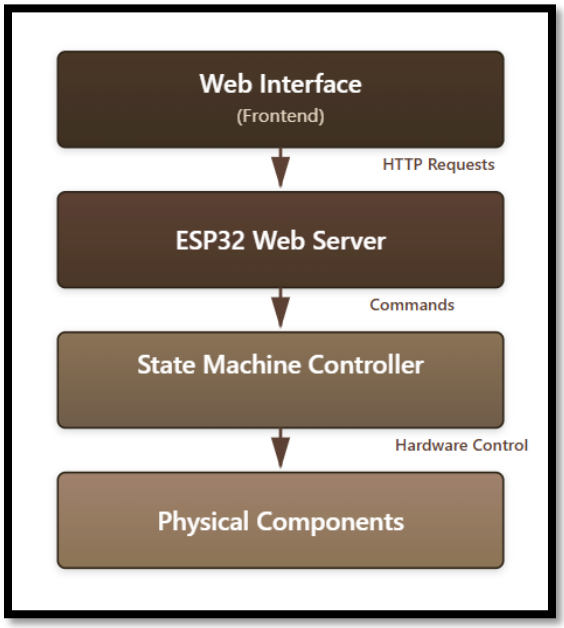
Block Diagram



Hardware Architecture

- ESP32 Microcontroller
- Motor Control
  - Relay 1 (GPIO 27) → Water Pump Motor
  - Relay 2 (GPIO 26) → Milk Pump Motor
- Heating Element Control
  - GPIO 23 → Stove
- Ingredient Dispensers (Servos)
  - Servo 1 (GPIO 18) → Tea Leaves Dispenser
  - Servo 2 (GPIO 19) → Sugar Dispenser

Software Architecture



State Machine Flow

Hardware Components

Component	Quantity	Purpose
ESP32 Development Board	1	Main controller
2-Channel Relay Module	1	Motor control
DC Water Pump	2	Water & milk dispensing
Servo Motors	2	Tea and Sugar dispensing
Stove	1	Tea Boiling
Power Supply	1	System power
Connecting Wires	-	Connections
Food Containers	4	Water, milk, tea and sugar storage
SSR	1	Automatically control stove

Batteries	1	To provide power to motors

Circuit Connections

Relay Module:

- VCC → 5V
- GND → GND (ESP32)
- IN1 → GPIO 27
- IN2 → GPIO 26
- COM → 12V+ (Power Supply)
- NO1 → Water Pump +
- NO2 → Milk Pump +

Servos:

- Red (VCC) → 5V
- Brown (GND) → GND
- Orange (Signal) → GPIO 18 and19

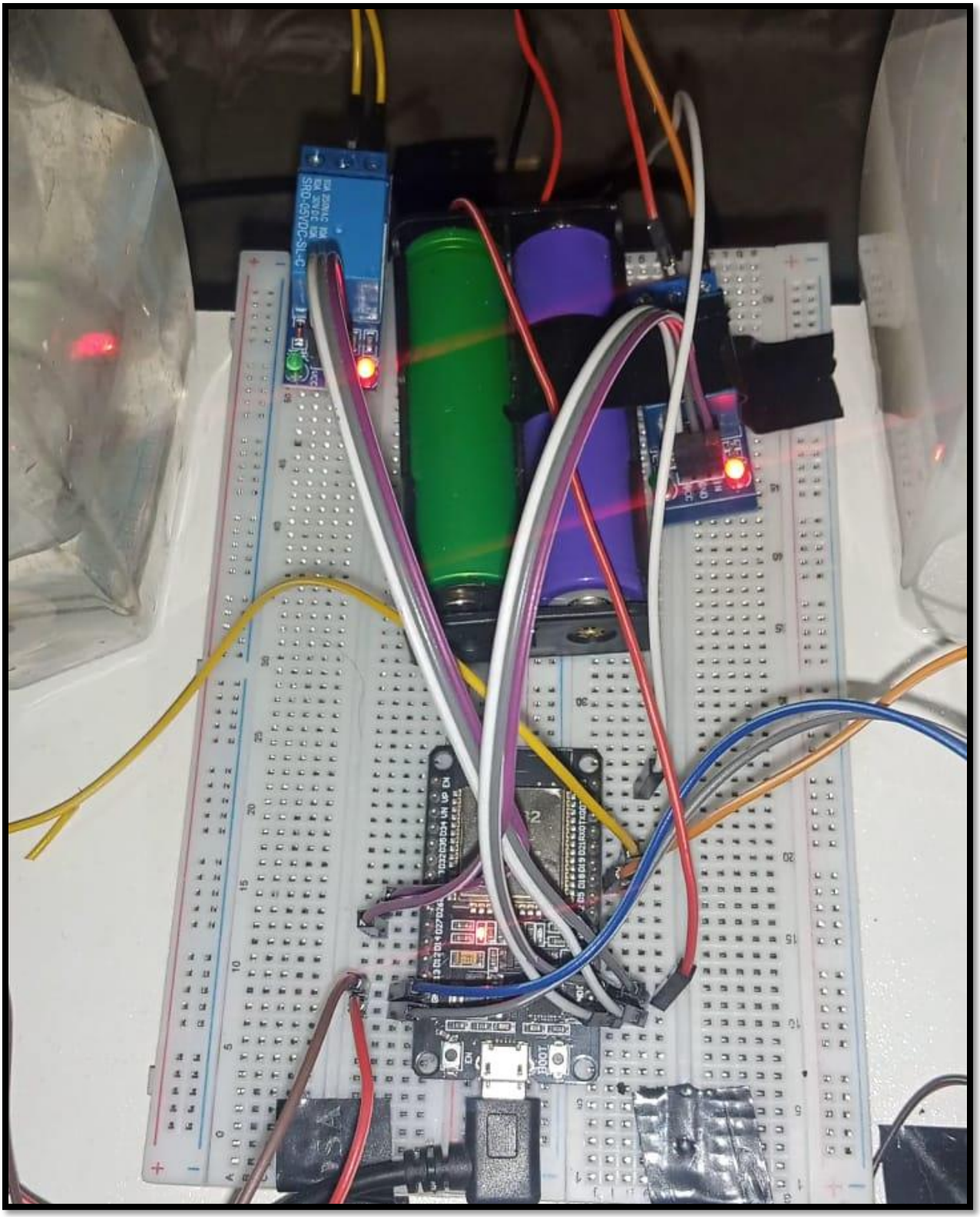
Stove:

- Control → GPIO 23
- Power → 12V supply

Software Components

- **Framework**: Arduino (PlatformIO)
- **Libraries**:
    - WiFi.h (ESP32 WiFi)
    - WebServer.h (HTTP server)
    - ESP32Servo.h (Servo control)
- **Frontend**: HTML, CSS, JavaScript
- **Communication**: HTTP REST API

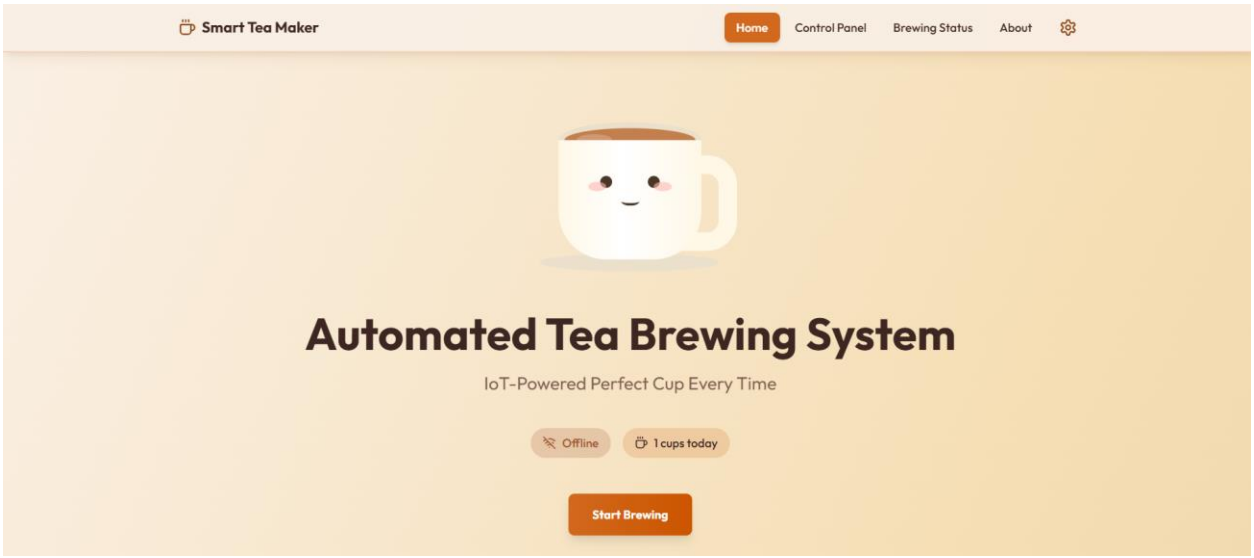
Screenshots Of Output

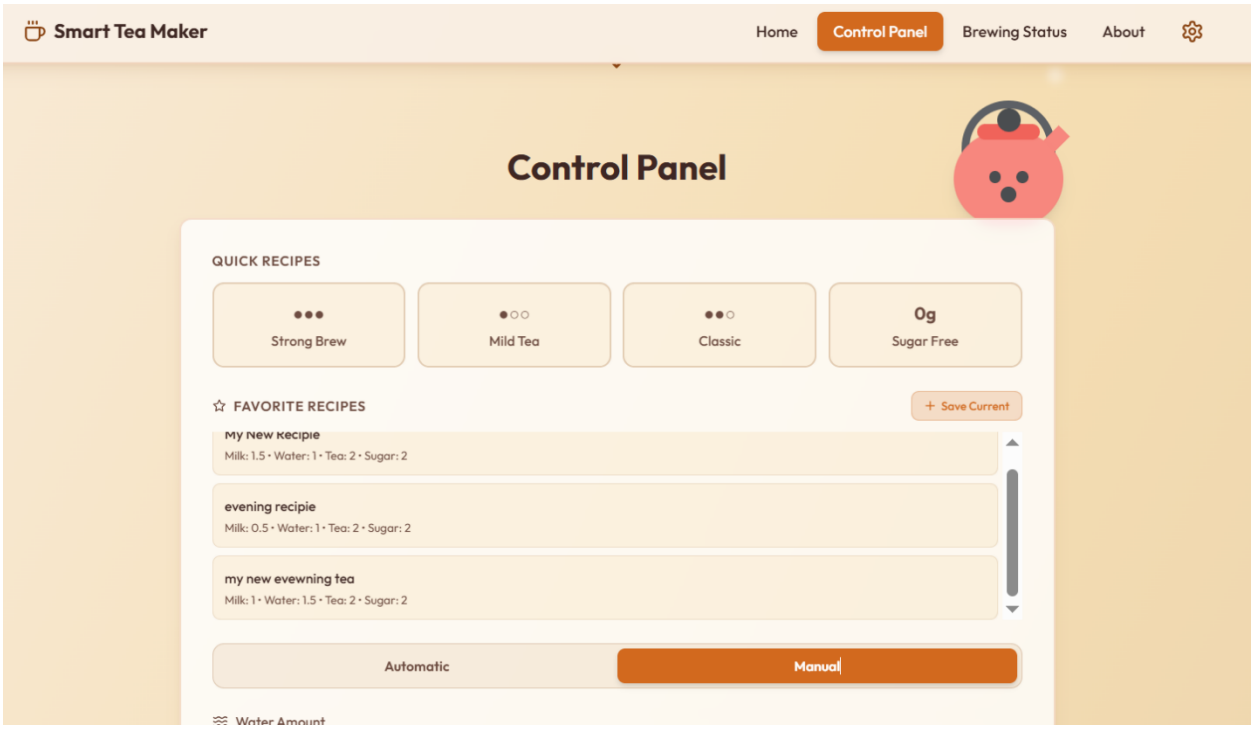

SRD-05VDC-SL-C
BOOT

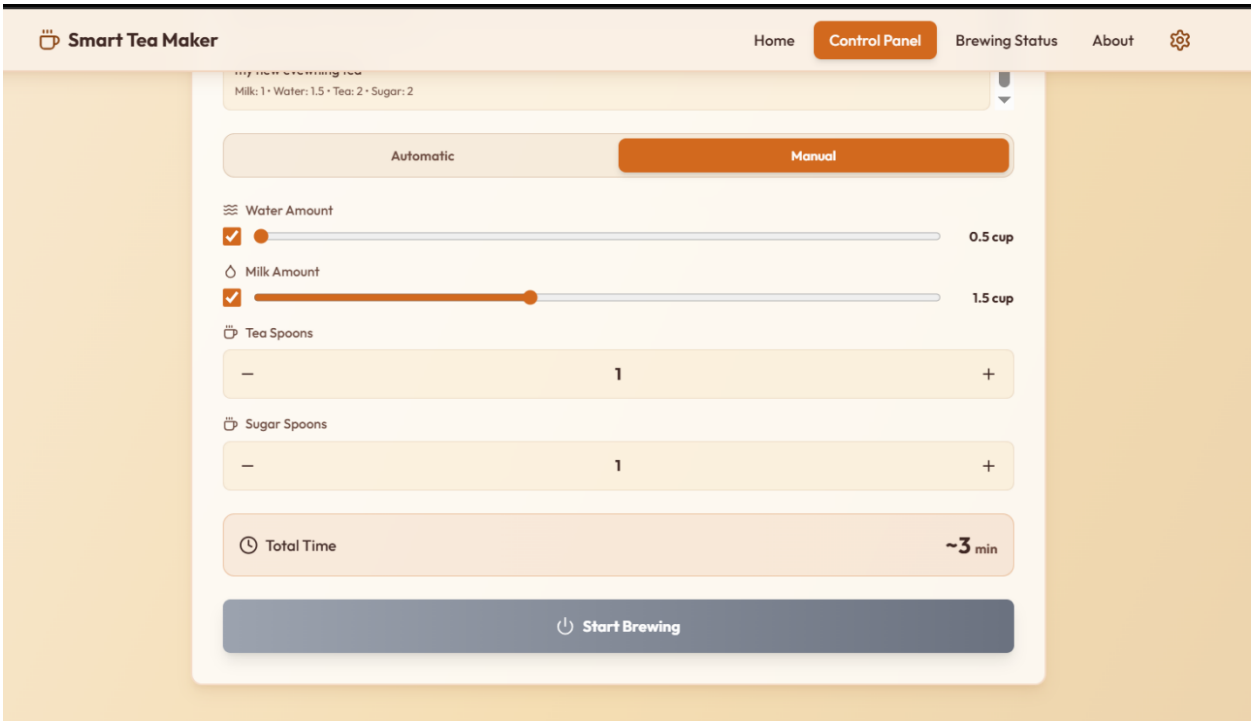
Automatic Stove ON/OFF feature

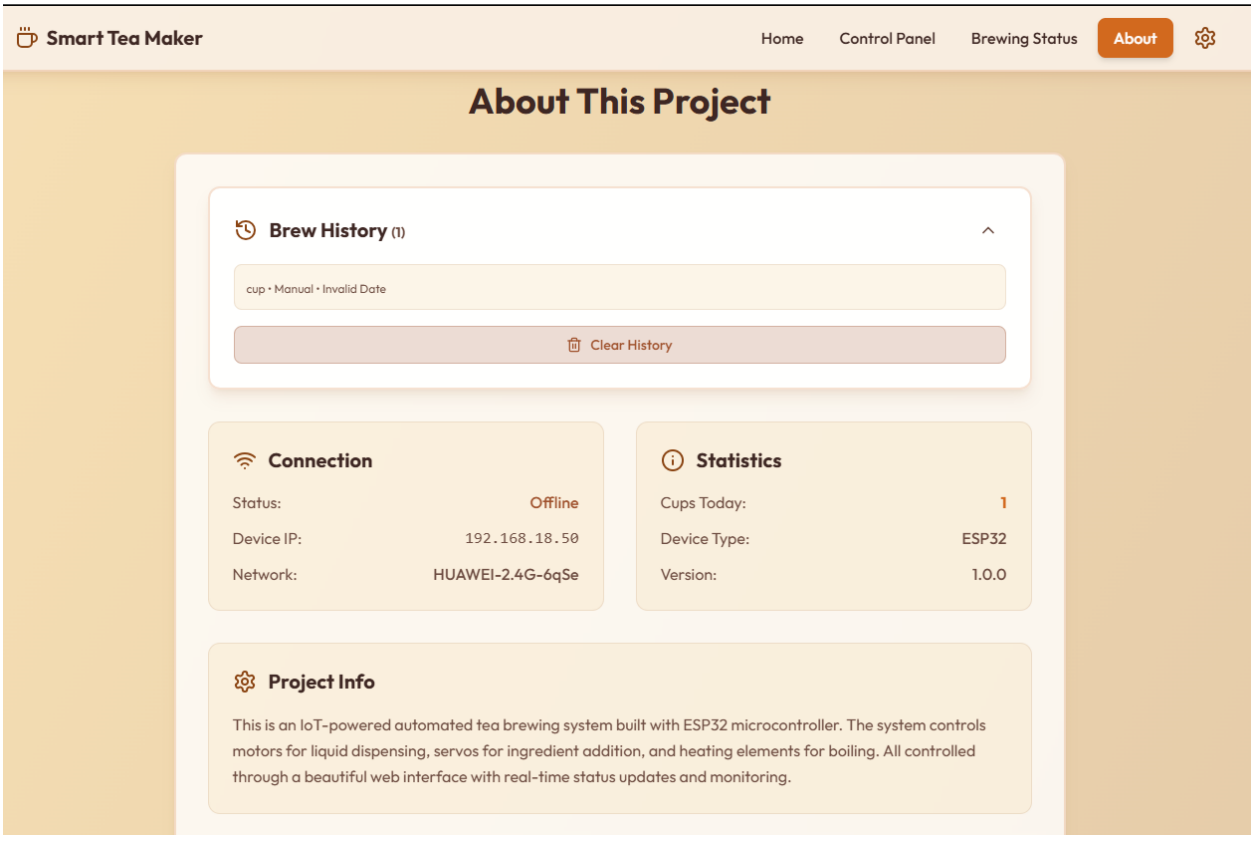
Stove turns ON and OFF automatically. When the ingredients are collected on pan stove is turned on. When the tea is ready, stive turns OFF automatically. For this purpose we attached **SSR (Solid state relay).**

Dashboard Screenshot






Smart Tea Maker
Home
Control Panel
Brewing Status
About
Milk: 1 • Water: 1.5 • Tea: 2 • Sugar: 2
Automatic
Manual
Water Amount
0.5 cup
Milk Amount
1.5 cup
Tea Spoons
1
Sugar Spoons
1
Total Time
~3 min
Start Brewing


Smart Tea Maker
Home
Control Panel
Brewing Status
About
About This Project
Brew History (1)
cup • Manual • Invalid Date
Clear History
Connection
Status: Offline
Device IP: 192.168.18.50
Network: HUAWEI-2.4G-6qSe
Statistics
Cups Today: 1
Device Type: ESP32
Version: 1.0.0
Project Info
This is an IoT-powered automated tea brewing system built with ESP32 microcontroller. The system controls motors for liquid dispensing, servos for ingredient addition, and heating elements for boiling. All controlled through a beautiful web interface with real-time status updates and monitoring.

Network Configuration

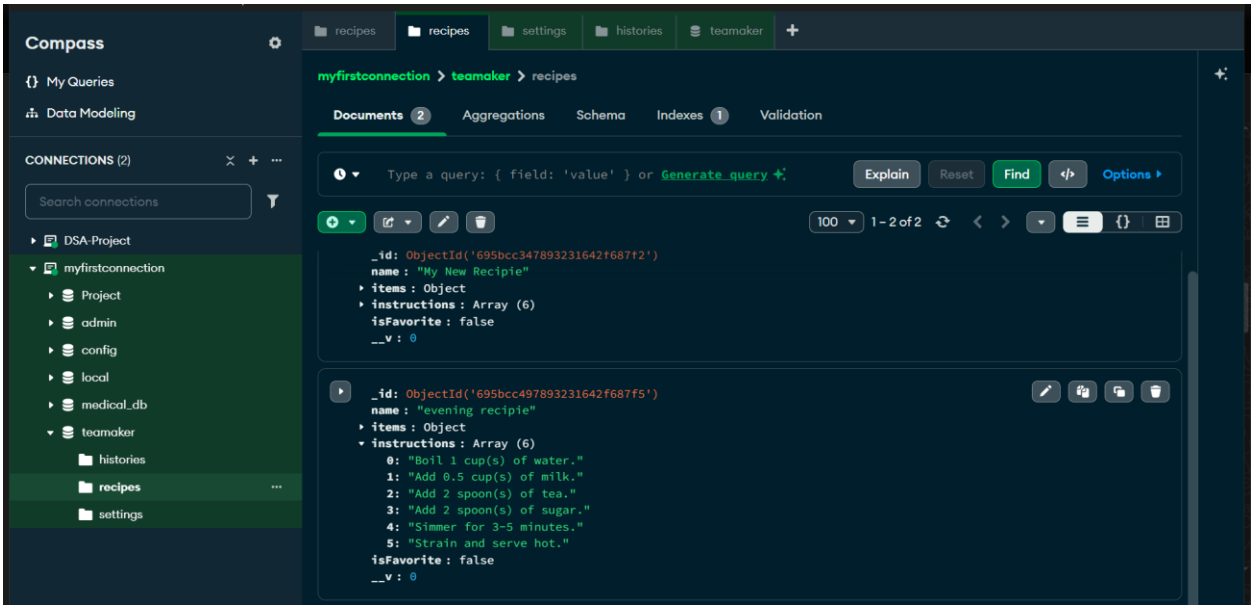
```
#include <Arduino.h>
#include <WiFi.h>
#include <WebServer.h>
#include <ESP32Servo.h>

const char *ssid = "HUAWEI-2.4G-6qSe";
const char *password = "wyeRy2eZ";

IPAddress local_IP(192, 168, 18, 50);
IPAddress gateway(192, 168, 18, 1);
IPAddress subnet(255, 255, 255, 0);
```

Methodology

1. **Power On**: Connect ESP32 to power supply
2. **Connect to WiFi**: Wait for device to connect (check serial monitor)
3. **Open Web Interface**: Navigate to device IP in browser
4. **Select Recipe**: Choose from quick recipes or customize
5. **Start Brewing**: Click "Start Brewing" button
6. **Monitor Progress**: Watch real-time status updates
7. **Collect Tea**: When status shows "Completed"

Cloud Database Integration by MongoDB

We integrated MongoDB Atlas cloud database to store recipes, brewing history, and user settings permanently. The database contains collections for recipes (with ingredients, instructions, and favorite flags), histories (past brewing sessions), and settings (user preferences). MongoDB Compass was used for database management and visualization during development.



MQTT Integration

```
/* WIFI CONFIG */
const char *ssid = "HUAWEI-2.4G-6qSe";
const char *password = "wyeRy2eZ";


/** MQTT CONFIG */
const char* mqtt_server = "192.168.0.115";
WiFiClient espClient;
PubSubClient client(espClient);

/* HARDWARE PINS */
const int RELAY1_PIN = 27;
const int RELAY2_PIN = 26;
const int STOVE_PIN  = 23;
const int SERVO1_PIN = 18;
const int SERVO2_PIN = 19;
```

```
/* MQTT STATUS PUBLISH */
void publishStatus(String state, long remaining)
{
  client.publish("tea/status/state", state.c_str());

  String json = "{\"state\":\"" + state +
                "\",\"remaining\":" + String(remaining) +
                ",\"motor1\":" + String(motor1Duration) +
                ",\"motor2\":" + String(motor2Duration) +
                ",\"servo1\":" + String(servo1Rotations) +
                ",\"servo2\":" + String(servo2Rotations) +
                ",\"stove\":"  + String(stoveDuration) + "}";

  client.publish("tea/status/full", json.c_str());
}
```

```
/* MQTT CALLBACK ***/
void mqttCallback(char* topic, byte* payload, unsigned int length)
{
  String message;
  for (int i = 0; i < length; i++)
    message += (char)payload[i];

  if (String(topic) == "tea/cmd/stop")
  {
    Serial.println("MQTT STOP RECEIVED");
    if (currentState == MOTOR1_RUNNING) digitalWrite(RELAY1_PIN, HIGH);
    if (currentState == MOTOR2_RUNNING) digitalWrite(RELAY2_PIN, HIGH);
    if (currentState == STOVE_RUNNING)  digitalWrite(STOVE_PIN, LOW);

    servo1.write(0);
    servo2.write(0);

    currentState = IDLE;
    publishStatus("idle", 0);
  }
}
```

```
/** MQTT RECONNECT ***/

void reconnectMQTT()
{
  while (!client.connected())
  {
    Serial.print("Connecting to MQTT...");
    if (client.connect("ESP32_TeaMaker"))
    {
      Serial.println("connected!");
      client.subscribe("tea/cmd/#");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" retrying...");
      delay(2000);
    }
  }
}
```

MQTT running successfully

```
E:\MQTT-setup\mosquitto>netstat -ano | findstr :1883
  TCP    0.0.0.0:1883           0.0.0.0:0              LISTENING       5260

E:\MQTT-setup\mosquitto>taskkill /PID 5260 /F
SUCCESS: The process with PID 5260 has been terminated.

E:\MQTT-setup\mosquitto>net start mosquitto

The Mosquitto Broker service was started successfully.
```

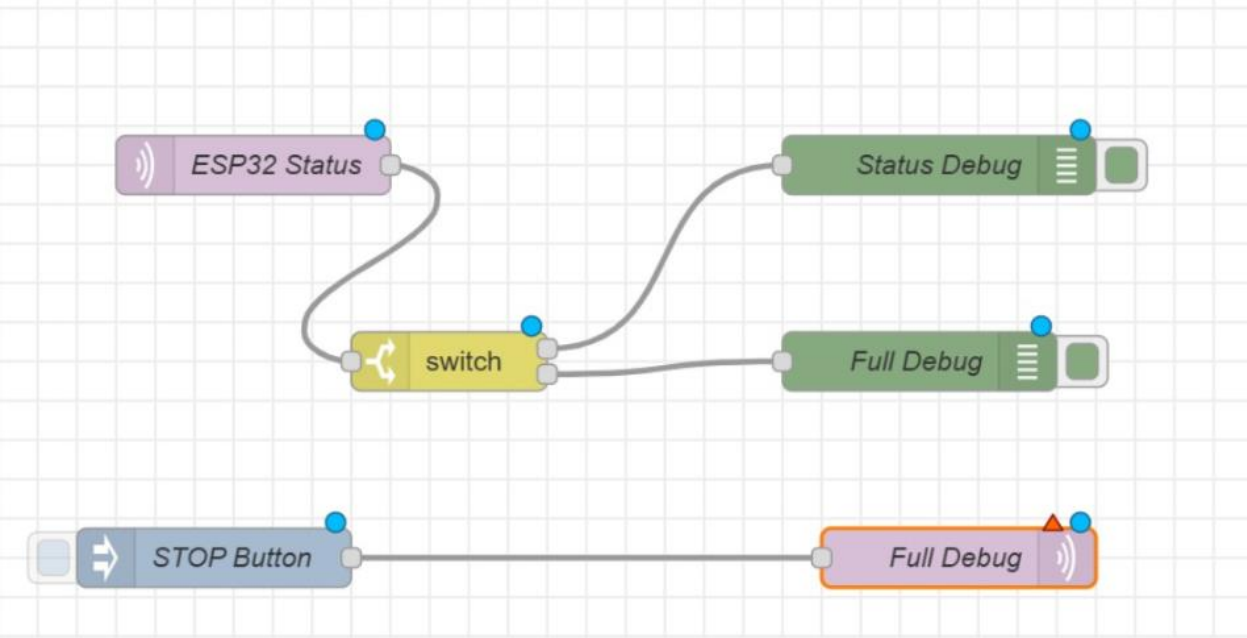
Node RED Integration

Node RED successfully configured and running.

```
E:\MQTT-setup\mosquitto>node-red
5 Jan 20:07:07 - [info]

Welcome to Node-RED
===================

5 Jan 20:07:07 - [info] Node-RED version: v4.1.2
5 Jan 20:07:07 - [info] Node.js  version: v18.20.4
5 Jan 20:07:07 - [info] Windows_NT 10.0.19045 x64 LE
5 Jan 20:07:10 - [info] Loading palette nodes
5 Jan 20:07:40 - [info] Settings file  : C:\Users\Imran Sb\.node-red\settings.js
5 Jan 20:07:40 - [info] Context store  : 'default' [module=memory]
```

```
5 Jan 20:07:40 - [info] Starting flows
5 Jan 20:07:40 - [info] Started flows
5 Jan 20:07:40 - [info] Server now running at http://127.0.0.1:1880/
5 Jan 20:08:41 - [info] Installing module: node-red-dashboard, version: 3.6.6
5 Jan 20:09:05 - [info] Installed module: node-red-dashboard
5 Jan 20:09:06 - [info] Dashboard version 3.6.6 started at /ui
5 Jan 20:09:07 - [info] Added node types:
5 Jan 20:09:07 - [info]  - node-red-dashboard:ui_base
```

Nodes Configured



Issues we faced

Motors Not Running

Command sent but motors don't activate.

Solutions:

- Check relay connections (GPIO 26, 27).
- Verify power supply to relays.
- Test relays manually (we connected relay to battery to check if it is working).
- Check motor power connections.

Servos Not Rotating

No movement from servo motors.

Solutions:

- Check servo power (5V) and GND connections.
- Verify signal wire connections (GPIO 18, 19).
- Check servo mechanical obstruction.
- Replace servo if faulty.

Incorrect Timing

Operations run shorter than expected.

Solutions:

- Verify millis() function working correctly.
- Check for code modifications affecting timing.

Results

Successful Automation: The system successfully automates the entire tea-making process from ingredient dispensing to heating, requiring zero manual intervention during operation.

Accurate Ingredient Dispensing:

- Water pump: Consistent dispensing with accuracy.
- Milk pump: Precise volume control.
- Tea servo: Reliable rotation for measured teaspoons.
- Sugar servo: Consistent sweetness levels.

Web Interface Performance:

- Responsive design works on mobile and desktop devices.
- Real-time status updates with < 1 second delay.
- Recipe saving and retrieval working flawlessly.

Timing Accuracy:

- Motor operations: Accurate to within 1 second.
- Heating cycle: Consistent timing with automatic shutoff.
- Total brewing time: 2-5 minutes depending on recipe.

Safety Features:

- Emergency stop responds immediately.

Conclusion

The Automated Tea Maker project successfully demonstrates the practical application of IoT and embedded systems in automating everyday tasks.

Future Scope

The Automated Tea Maker has significant potential for enhancement and expansion:

Temperature Sensor Integration:

- Implement precise temperature control for optimal brewing.

Water Level Sensors:

- Implement ultrasonic sensors for container level monitoring.
- Alert users when containers need refilling.
- Display remaining capacity on dashboard.

Voice Control Integration:

- Integrate with Google Assistant.
- Alexa skill development.
- Voice commands for tea preparation.

Scheduling System:

- Set tea preparation times in advance.
- Recurring schedules (daily morning tea).
- Calendar integration.