Final Project Report – Smart Home Embedded System

Name : Maha Ibrahim

Course: Embedded Systems

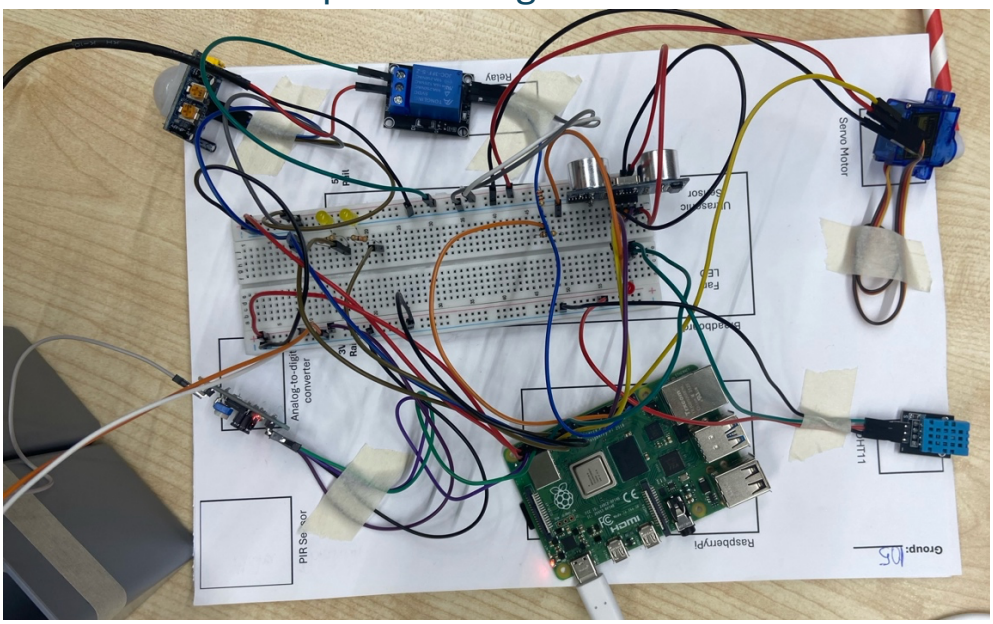
1.0 Objective of the Report

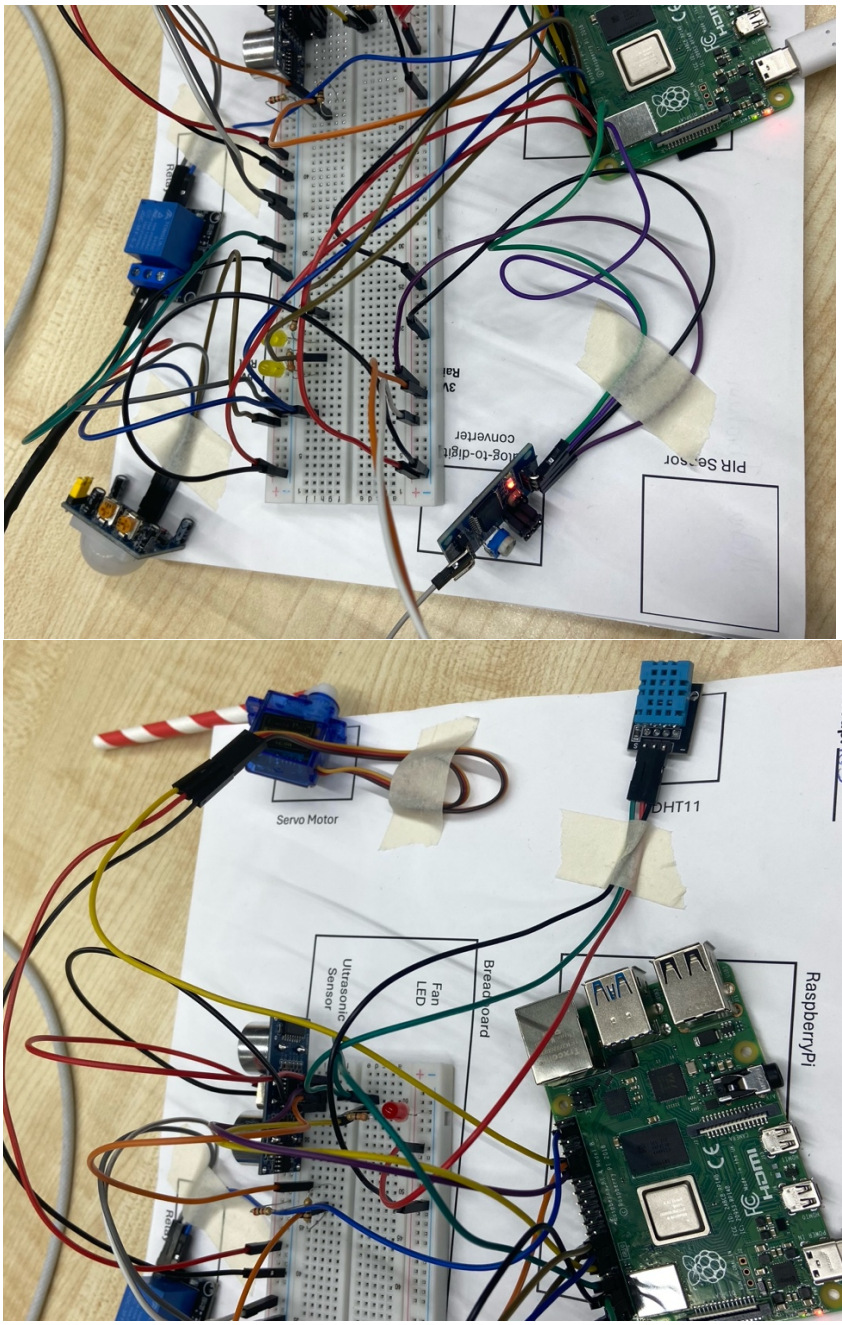
The objective of this project is to demonstrate the design and integration of a complete smart home embedded system using a Raspberry Pi. Multiple sensor and actuator subsystems were combined into a single IoT-based system controlled through MQTT. The project focuses on hardware–software integration, communication between distributed processes, and system-level behaviour rather than isolated components. The final system allows real-time monitoring and automated control through a central controller and a Grafana dashboard.

2.0 System Overview

The implemented smart home system integrates several independent subsystems: climate monitoring, motion-based lighting, distance-based gate control, and water tank monitoring with automated pump control. Each subsystem publishes sensor data to specific MQTT topics, while a single controller script subscribes to all topics and applies automation rules. Sensor data is stored in InfluxDB and visualised using Grafana for monitoring and debugging. This architecture ensures modularity, scalability, and loose coupling between components.

2.1 Hardware Setup and Wiring





Complete Smart Home Embedded System Hardware Setup

Connected Components

The Raspberry Pi serves as the central controller. A DHT11 sensor measures temperature and humidity, while a PIR sensor detects motion to control room lighting LEDs. An ultrasonic sensor measures distance for gate control using a servo motor. Water level monitoring is implemented using a resistive water level sensor connected through a PCF8591 analog-to-digital converter. A relay module is used to safely switch a 5 V DC water pump.

Power Distribution (3.3 V vs 5 V)

Both 3.3 V and 5 V power rails are used. The Raspberry Pi's 3.3 V supply powers low-voltage components such as the DHT11 sensor, PCF8591 ADC, and water level sensor to

ensure GPIO safety. The 5 V rail powers higher-current or 5 V devices, including the PIR sensor, ultrasonic sensor, relay module, servo motor, and water pump. All components share a common ground to ensure correct signal referencing.

Use of Relays, Servo Motors, and Resistors

A relay module is used to electrically isolate the Raspberry Pi from the water pump, preventing high current from flowing through GPIO pins. The servo motor controlling the gate is driven by a PWM signal from the Raspberry Pi and powered using the 5 V supply. Current-limiting resistors (330 Ω) are used in series with all LEDs to prevent excessive current and protect both the LEDs and the GPIO pins. In addition, a voltage divider is implemented on the ultrasonic sensor's ECHO pin to safely interface the 5 V output with the Raspberry Pi's 3.3 V GPIO input. The voltage divider consists of a 1 kΩ resistor (R1) connected between the ECHO pin and the GPIO input, and a 2 kΩ resistor (R2) connected between the GPIO input and ground, reducing the signal voltage to approximately 3.3 V and preventing damage to the Raspberry Pi.

Wiring Colour Coding

Power and Ground

- **Red wires**: Power supply lines
  - 5 V for ultrasonic sensor, servo motor
  - 3.3 V for DHT11 sensor
- **Black wires**: Ground (GND)
  - All components share a common ground with the Raspberry Pi
  - DHT11 , ultrasonic sensor, servo motor, PCF8591 ADC
- **Grey wire:**
  - GND for PIR Motion Sensor and Relay Module

- **White wire**:
  - 5V for relay module and 3.3 V for water level sensor
- **Purple wire**:
  - 3.3 V for PCF8591 ADC

- **Orange wires**:
  - GND for Water Level Sensor
- **Brown wire**:
  - 5 V for PIR sensor

Sensor Signal Connections

- **Green wire**:
  - DHT11 data signal → GPIO4
- **Blue wire**:
  - PIR motion sensor output → GPIO27
- **Purple wire**:

    - Ultrasonic TRIG signal → GPIO5
- **Orange wire**:
    - Ultrasonic ECHO signal → voltage divider → GPIO6
- **Grey wire**:
    - Water level sensor signal → PCF8591 AIN0

LED and Actuator Control Signals

- **Brown wires**:
    - Room LEDs control signals → GPIO22, GPIO23 (each via 330 Ω resistor)
- **Yellow wire**:
    - Fan indicator LED control → GPIO17 via 330 Ω resistor
- **Blue wire**:
    - Relay control input (pump ON/OFF) → GPIO26

I²C Communication (PCF8591 ADC)

- **Purple wire**:
    - SDA (data line) → GPIO2
- **Green wire**:
    - SCL (clock line) → GPIO3

Servo Motor Wiring

- **Yellow wire**:
    - Servo control signal → GPIO12
- **Red wire**:
    - Servo power (5 V)
- **Black wire**:
    - Servo ground

2.2 I²C Communication in the System

I²C (Inter-Integrated Circuit) is a communication protocol that allows devices to exchange data using only two wires: SDA for data and SCL for the clock signal. In this system, the Raspberry Pi acts as the master and communicates with connected modules by sending and receiving data over these two shared lines.

I²C is useful in embedded systems because it reduces the number of GPIO pins and wires needed, which keeps the hardware setup simple and organized. This is especially important when multiple sensors and modules are connected to a single Raspberry Pi.

Compared to using separate GPIO pins for each device, I²C allows multiple devices to share the same bus while still being individually addressed. This makes the system easier to expand and maintain.

In this implementation, the PCF8591 Analog-to-Digital Converter uses I²C to send water level sensor readings to the Raspberry Pi via GPIO2 (SDA) and GPIO3 (SCL). The PCF8591 has

its own I²C address, which allows it to communicate on the same bus without interfering with other devices.

2.3 Real-Time Behaviour of the System

The system does not operate as a hard real-time system because it runs on Linux, which is not a real-time operating system. Linux uses time-sharing and task scheduling, meaning process execution can be delayed due to other running tasks, and exact timing cannot be guaranteed.

Hard real-time systems require strict deadlines where missing a deadline causes system failure, while soft real-time systems allow small delays without serious consequences. In this project, missing an exact timing deadline does not cause critical errors, only minor delays in response.

Several parts of the system exhibit soft real-time behaviour, such as gate control using the ultrasonic sensor, and automatic pump activation based on water level. These actions respond within a short time but do not require millisecond-level precision.

Soft real-time behaviour is sufficient for this application because slight delays do not affect safety or functionality. The system remains reliable and responsive enough for a smart home environment.

2.4 Power Distribution and Observed System Behaviour

When the servo motor and water pump were activated, the system showed small disturbances such as brief delays or unstable sensor readings. This happened because motors draw a high current when starting, which can cause temporary voltage drops on the Raspberry Pi power lines.

Since all components share a common ground and some devices use the same 5 V supply, electrical noise and inrush current can affect sensitive sensors. To improve this, the pump and servo should be powered from a separate 5 V supply while keeping a shared ground, and additional filtering components could be added.

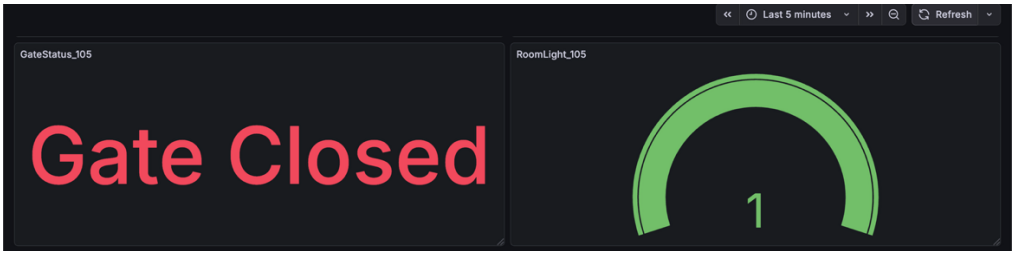
2.5 System Operation and Control Logic

During normal operation, each sensor runs as an independent MQTT publisher that sends data to a specific topic, such as temperature, motion, distance, or water level. A single controller script acts as an MQTT subscriber, listens to all topics, and applies automation rules, for example switching lights when motion is detected, opening the gate when an object is close, or starting the pump when the water level is low.

MQTT is used to decouple the system components, meaning sensors and actuators do not communicate directly with each other but only through the broker. This makes the system

modular, easier to debug, and allows each subsystem to run independently without affecting others.

2.6 Monitoring and Visualisation



Grafana Dashboard for Smart Home System

The Grafana dashboard visualises real-time data from all smart home subsystems, including temperature, humidity, water level, gate status, fan state, and room lighting state. Sensor values are shown using gauges and bar indicators, while actuator states such as the gate, fan, and lights are displayed as clear ON/OFF or status panels. This allows the user to quickly understand the current state of the entire system at a glance.

Time-series data is important because it shows how sensor readings and actuator states change over time, not just their current value. This makes it easier to monitor system behaviour, verify that automation rules are working correctly, and debug issues such as delayed responses or unexpected state changes.