CAD_PHASE5

Project Overview

The Smart Home Transformation Project aims to convert a conventional home into a modern, intelligent living space by incorporating various smart devices. These devices will enhance energy efficiency, home security, and overall convenience for the residents. The project leverages IoT technology and data processing capabilities to achieve these objectives.

Objectives

- Integration of smart devices for different areas of the home.
- Real-time data collection and processing.
- Automation for energy efficiency and home security.
- Voice control using an Alexa-like system for hall and bedroom.
- Bathroom enhancements with sensor taps, automatic hand dryers, and soap dispensers.
- Kitchen upgrades for gas leakage monitoring and appliance control (light, fridge, oven).

Milestones

1. Device Selection and Procurement
2. Integration and Connectivity Setup
3. Data Collection and Processing Implementation
4. Automation Logic Development
5. Voice Control Integration
6. Bathroom Enhancements
7. Kitchen Upgrades
8. Testing and Quality Assurance
9. Documentation and Training
10. Deployment and Monitoring

Innovations in Smart Home Automation

1. *Device Compatibility Assessments:*

One of the most significant innovations in smart device selection is the development of compatibility assessment tools. These tools, often available as mobile apps or online platforms, allow users to input their existing smart devices and preferred ecosystems (e.g., Apple HomeKit, Amazon Alexa, Google Assistant) to find devices that are compatible with their setup. These tools can also suggest additional devices that work seamlessly within the chosen ecosystem, simplifying the integration process.

2*. Artificial Intelligence-Powered Recommendations:*

Artificial intelligence (AI) plays a pivotal role in helping users select the right smart devices. AI-driven recommendation engines take into account a user's preferences, behavior, and the existing smart devices in their home to suggest new devices. These recommendations are based on user habits, the compatibility of devices, and emerging technologies, ensuring that users can make informed decisions without extensive research.

Innovations in System Architecture

1. *Edge Computing Architecture:*

Edge computing represents a significant shift from traditional cloud-based architectures. This innovation involves placing computing resources closer to the data source or "edge" of the network. It reduces latency and enhances real-time processing capabilities, making it ideal for applications such as IoT, autonomous vehicles, and augmented reality.

2.Serverless Computing:

Serverless computing, often associated with Function as a Service (FaaS), is an innovative architectural approach that abstracts infrastructure management from developers. Applications are built as a series of functions or microservices that are executed in response to events. This architecture simplifies deployment, scaling, and resource management.

Innovations in Voice Control Integration for Smart Homes

1. *Multimodal Voice Control:*

The integration of multimodal voice control allows users to combine voice commands with other inputs, such as touch, gestures, or visual cues. For example, users can point to a specific light while saying, "Turn this light off," or they can control a smart display using both voice and touch simultaneously. This innovation enhances the user experience and expands the possibilities for interaction.

2. *Custom Wake Words:*

Customizable wake words are an emerging innovation that allows users to personalize the way they activate their voice assistants. Instead of using generic wake words like "Alexa" or "Hey Google," users can choose custom wake words, increasing privacy and user engagement.

COMPONENTS USED:

Arduino UNO:

The Arduino UNO serves as the central processing unit, controlling and coordinating the functions of other components. It's a microcontroller board that runs the Arduino software.

The Arduino UNO is widely used in IoT projects due to its simplicity and versatility. It can be programmed using the Arduino IDE, making it accessible for both beginners and experienced developers. The presence of digital and analog pins allows for a variety of sensor and actuator connections.

Use Cases:
Home Automation: Control lights, fans, and appliances.
Data Logger: Monitor and log sensor data over time.
Robotics: Control the movement of robots.

4 Relay Module:

Relay modules act as an interface between the low-voltage Arduino and high-voltage devices. They are crucial for controlling appliances securely. Understanding the power requirements of connected devices and ensuring proper isolation are important considerations when using relay modules.

Use Cases:
Home Automation: Control high-voltage devices like lights or heaters.
Smart Switching: Turn on/off appliances remotely.
Industrial Automation: Control machinery or equipment.

LCD Display:

LCD displays come in different types, such as character LCDs or graphical LCDs. The choice depends on the project requirements. The display not only provides real-time information but also enhances user interaction. Custom characters and graphics can be displayed for a more intuitive user interface

Use Cases:
Weather Station: Display temperature, humidity, and weather conditions.
Smart Home Interface: Show real-time status and alerts.
Parameter Monitoring: Display critical data such as voltage or current.

Ultra sonic sensor:

Ultrasonic sensors are crucial for applications like smart parking systems, security systems, or robotics. They work based on the echo of ultrasonic waves, making them suitable for distance measurement. Calibration and accurate placement are essential for reliable readings.

Use Cases:
Parking Assist: Measure distance for parking assistance.
Security Systems: Detect intruders or unauthorized movement.
Robotics: Implement obstacle avoidance in robots.

DHT11 Temperature and humidity Sensor:

This provides accurate temperature readings and is often used in climate monitoring systems, weather stations, or home automation projects. Calibration may be necessary for precise temperature measurements, and the analog output can be converted to Celsius or Fahrenheit.

Use Cases:

Climate Monitoring: Monitor temperature in a greenhouse or room.
HVAC Systems: Control heating, ventilation, and air conditioning.
Health Monitoring: Measure body temperature for medical applications.

IR Proximity Detection Sensor:

Infrared (IR) sensors for proximity detection use infrared light to detect the presence or absence of an object within their sensing range. These sensors typically consist of an infrared emitter and a receiver. When an object is within the sensor's range, it reflects the emitted infrared light back to the sensor, triggering a response.

Use Cases:
Automatic Doors: IR sensors detect the presence of individuals, enabling doors to open automatically in public spaces like supermarkets.

Robot Obstacle Avoidance: Used in robotics for obstacle avoidance, IR sensors help robots navigate and change direction in response to detected obstacles.

BreadBoard:

A breadboard is a versatile prototyping tool in electronics, providing a platform for quickly building and testing electronic circuits without soldering. It consists of a grid of interconnected metal strips and holes, allowing components like resistors, capacitors, and integrated circuits to be easily plugged in and interconnected, making it ideal for rapid experimentation and design in electronics projects.

Use Cases:

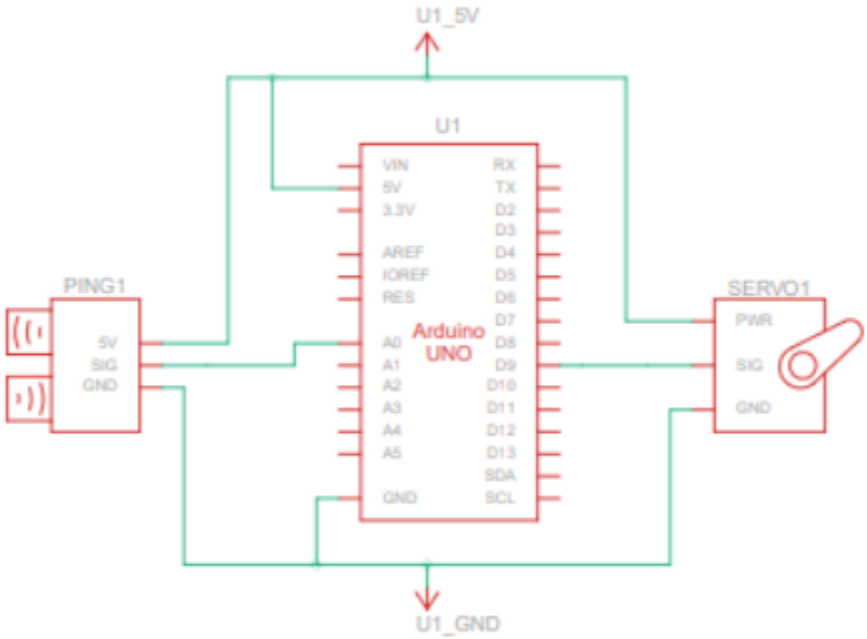
Prototyping Circuits: Breadboards are essential for rapid prototyping of electronic circuits, allowing engineers and hobbyists to experiment with components and designs without soldering.
Educational Tool: Widely used in electronics education, breadboards help students learn circuitry concepts and construct circuits for hands-on experimentation and understanding.

UV Sensor Connection:

UV Sensor VCC (Power): Connect the UV sensor's VCC (Power) pin to a 5V output on your Arduino. This provides power to the UV sensor.

UV Sensor GND (Ground): Connect the UV sensor's GND (Ground) pin to one of the GND (Ground) pins on your Arduino to complete the circuit.

UV Sensor Analog Output: Connect the analog output pin of the UV sensor (usually labeled as "OUT") to an analog input pin on your Arduino. In the code example provided, we used A0 as the analog input pin.



CODE 1:

```
#include <Servo.h>
```

```
Servo servo;  // Create a servo object
int servoPosition = 0;  // Initial servo position

const int uvSensorPin = A0;  // Analog pin for UV sensor
const int thresholdDistance = 5;  // Threshold distance in centimeters

void setup() {
  servo.attach(9);  // Attach the servo to pin 9
  servo.write(servoPosition);  // Set the initial servo position to 0 degrees
  pinMode(uvSensorPin, INPUT);  // Set UV sensor pin as input
  Serial.begin(9600);  // Initialize serial communication
}

void loop() {
  int uvSensorValue = analogRead(uvSensorPin);  // Read UV sensor value

  // Convert UV sensor value to a distance in centimeters (example
conversion)
  float distance = map(uvSensorValue, 0, 1023, 0, 50);

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance < thresholdDistance) {
    // Object detected within 5cm, move servo to 90 degrees
    servo.write(90);
  } else {
    // No object detected, move servo to 0 degrees
    servo.write(0);
  }

  delay(1000);  // Delay for stability
}
```

Explanation:

We include the necessary libraries, Servo and set up the servo object.
Define the analog pin where the UV sensor is connected and the threshold distance for object detection.
In the setup() function, we attach the servo to pin 9, set the initial servo position to 0 degrees, configure the UV sensor pin as input, and initialize serial communication for debugging.
In the loop() function, we read the UV sensor value and convert it to a distance in centimeters (the conversion may vary depending on your UV sensor).
We then check if the distance is less than the threshold (5cm). If an object is detected within 5cm, the servo moves to 90 degrees; otherwise, it returns to 0 degrees.
We add a delay for stability (you can adjust this as needed).

TMP36 Temperature Sensor Connections:

VCC (Power): Connect the VCC (Power) pin of the TMP36 to the 5V output on your Arduino. This supplies power to the sensor.

GND (Ground): Connect the GND (Ground) pin of the TMP36 to one of the GND (Ground) pins on your Arduino to complete the circuit.

Output: Connect the Output pin of the TMP36 to the analog input pin A0 on your Arduino. This is where the Arduino will read the analog voltage output from the sensor.

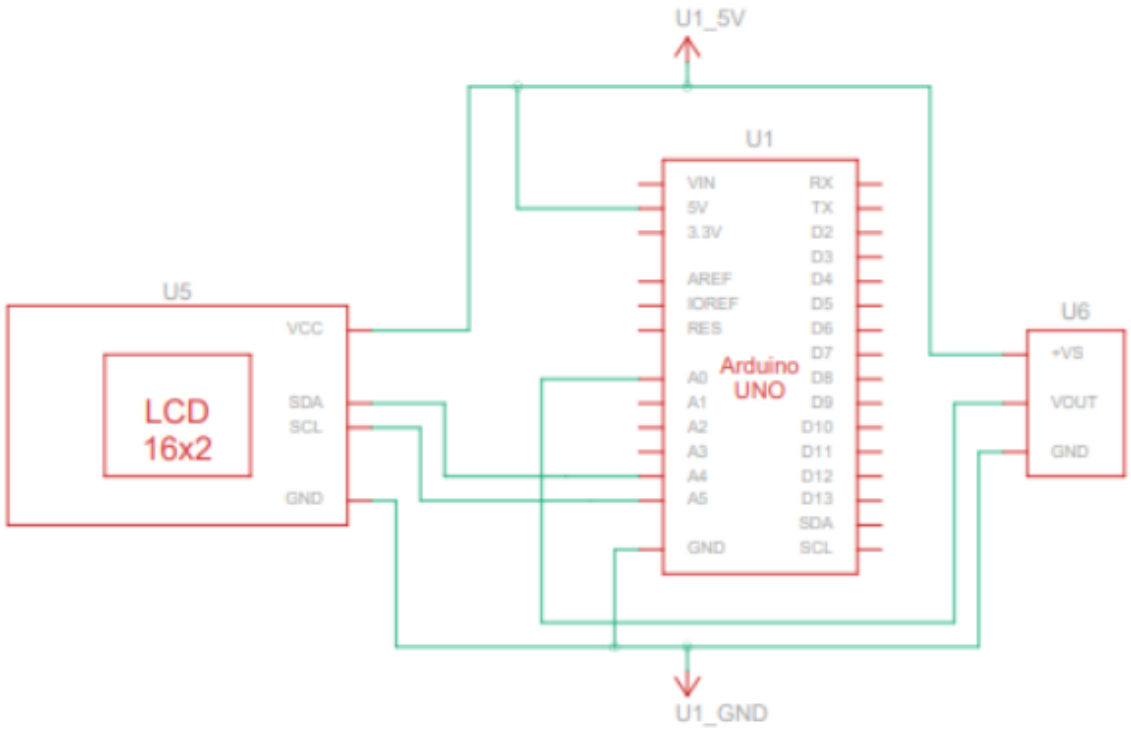
I2C LCD Connections:

SDA (Serial Data): Connect the SDA (Serial Data) pin of the I2C LCD to the corresponding SDA pin on your Arduino. On most Arduino boards, this is A4.

SCL (Serial Clock): Connect the SCL (Serial Clock) pin of the I2C LCD to the corresponding SCL pin on your Arduino. On most Arduino boards, this is A5.

VCC (Power): Connect the VCC (Power) pin of the I2C LCD to the 5V output on your Arduino for power.

GND (Ground): Connect the GND (Ground) pin of the I2C LCD to one of the GND (Ground) pins on your Arduino to complete the circuit.



CODE 2:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 columns and 2 rows
int tempPin = A0; // Analog input pin for TMP36 sensor

void setup() {
  lcd.init(); // Initialize the LCD
  lcd.backlight(); // Turn on the backlight
  Serial.begin(9600); // Initialize serial communication for debugging
}

void loop() {
  int sensorValue = analogRead(tempPin); // Read the TMP36 sensor value
  float voltage = (sensorValue / 1023.0) * 5.0; // Convert to voltage
  float temperatureC = (voltage - 0.5) * 100.0; // Convert to degrees Celsius

  lcd.clear(); // Clear the LCD display
  lcd.setCursor(0, 0); // Set the cursor to the first row
  lcd.print("Temperature: ");
  lcd.setCursor(0, 1); // Set the cursor to the second row
  lcd.print(temperatureC);
  lcd.print("C");

  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println("C");

  delay(1000); // Delay for one second between readings
}
```

Explanation:

We include the necessary libraries, including Wire for I2C communication and LiquidCrystal_I2C for the LCD.

We initialize the LCD with its I2C address (0x27), 16 columns, and 2 rows.

We define the analog input pin (tempPin) where the TMP36 temperature sensor is connected.

In the setup() function, we initialize the LCD, turn on the backlight, and start serial communication for debugging (you can view temperature values on the serial monitor if needed).

In the loop() function, we read the analog value from the TMP36 sensor and convert it to voltage.

We calculate the temperature in degrees Celsius using the TMP36 formula.

We clear the LCD and display the temperature on the first row (line 0) and the second row (line 1).

We also print the temperature value to the serial monitor for debugging.

Finally, we add a delay of one second to prevent the LCD from updating too quickly.

LED 1 and LED 2:

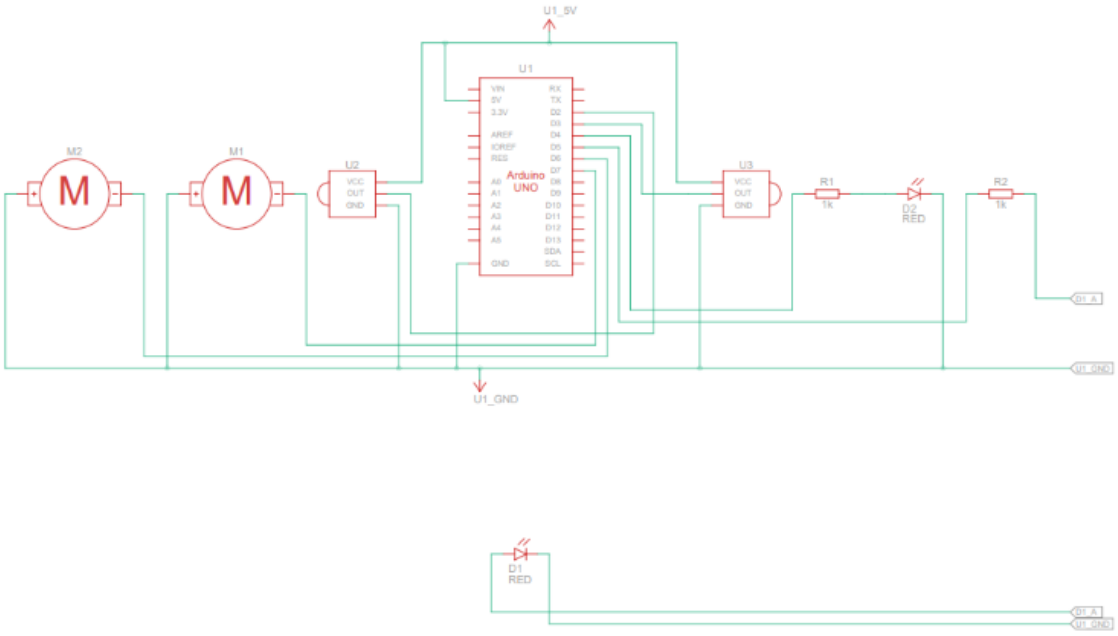
LED 1: Connect one terminal of LED 1 to a current-limiting resistor (e.g., 220-330 ohms), and connect the other end of the resistor to digital pin 4 (led1Pin) on your Arduino. Connect the remaining terminal of LED 1 to GND.

LED 2: Connect one terminal of LED 2 to a current-limiting resistor (e.g., 220-330 ohms), and connect the other end of the resistor to digital pin 5 (led2Pin) on your Arduino. Connect the remaining terminal of LED 2 to GND.

Motor 1 and Motor 2:

Motor 1: Connect one terminal of Motor 1 to the 5V output on your Arduino or an external power supply. Connect the other terminal of Motor 1 to a motor driver or an H-bridge module, and control its direction and speed using two digital pins. You'll need a motor driver for bi-directional control, and the specific connections will depend on the motor driver you are using.

Motor 2: Similar to Motor 1, connect one terminal of Motor 2 to the 5V output, and the other terminal to the motor driver. Use two digital pins for controlling Motor 2.



CODE 3:

```
const int irSensor1Pin = 2;  // IR sensor 1 connected to digital pin 2
const int irSensor2Pin = 3;  // IR sensor 2 connected to digital pin 3
const int led1Pin = 4;       // LED 1 connected to digital pin 4
const int led2Pin = 5;       // LED 2 connected to digital pin 5
```

```
const int motor1Pin = 6;     // Motor 1 connected to digital pin 6
const int motor2Pin = 7;     // Motor 2 connected to digital pin 7

void setup() {
  pinMode(irSensor1Pin, INPUT);
  pinMode(irSensor2Pin, INPUT);
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);
  pinMode(motor1Pin, OUTPUT);
  pinMode(motor2Pin, OUTPUT);
}

void loop() {
  int sensor1Value = digitalRead(irSensor1Pin);
  int sensor2Value = digitalRead(irSensor2Pin);

  if (sensor1Value == HIGH) {
    // Sensor 1 detects something, turn on LED 1 and Motor 1
    digitalWrite(led1Pin, HIGH);
    digitalWrite(led2Pin, LOW);
    digitalWrite(motor1Pin, HIGH);
    digitalWrite(motor2Pin, LOW);
  } else if (sensor2Value == HIGH) {
    // Sensor 2 detects something, turn on LED 2 and Motor 2
    digitalWrite(led1Pin, LOW);
    digitalWrite(led2Pin, HIGH);
    digitalWrite(motor1Pin, LOW);
    digitalWrite(motor2Pin, HIGH);
  } else {
    // No sensor detects anything, turn off both LEDs and Motors
    digitalWrite(led1Pin, LOW);
    digitalWrite(led2Pin, LOW);
    digitalWrite(motor1Pin, LOW);
    digitalWrite(motor2Pin, LOW);
  }
}
```

Explanation:

We define the pins for the two IR sensors (irSensor1Pin and irSensor2Pin), two LEDs (led1Pin and led2Pin), and two motors (motor1Pin and motor2Pin).

In the setup() function, we set the pins as either INPUT or OUTPUT depending on their function.

In the loop() function, we read the values from both IR sensors.

If sensor1Value is HIGH, it means that Sensor 1 has detected something. In this case, we turn on LED 1 and Motor 1 and turn off LED 2 and Motor 2.

If sensor2Value is HIGH, it means that Sensor 2 has detected something. In this case, we turn on LED 2 and Motor 2 and turn off LED 1 and Motor 1.

If neither sensor detects anything, we turn off both LEDs and both Motors.

IBM CLOUD:

A "smart home automation system on IBM Cloud" would typically refer to a setup where Internet of Things (IoT) devices and sensors within a smart home are connected to the IBM Cloud platform to enable centralized control, data analysis, and automation.

The specific capabilities and use cases may vary depending on the implementation and the services provided by IBM Cloud. It's essential to research the specific offerings and consult with IBM or an authorized reseller to understand how IBM Cloud can be leveraged effectively for your smart home automation needs.

Architecture

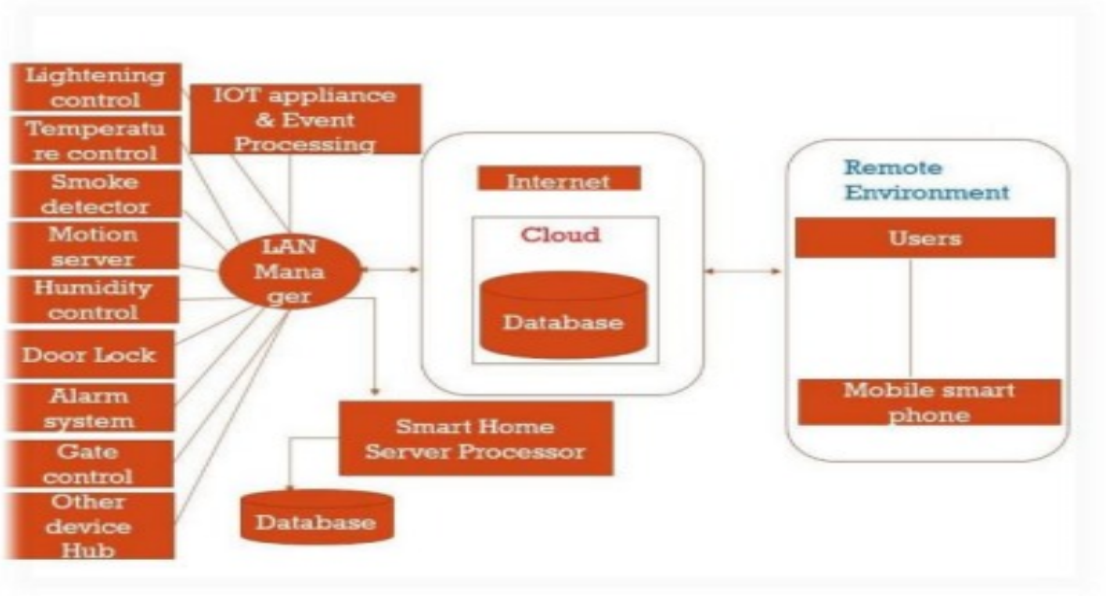


Fig 1: Architecture Diagram

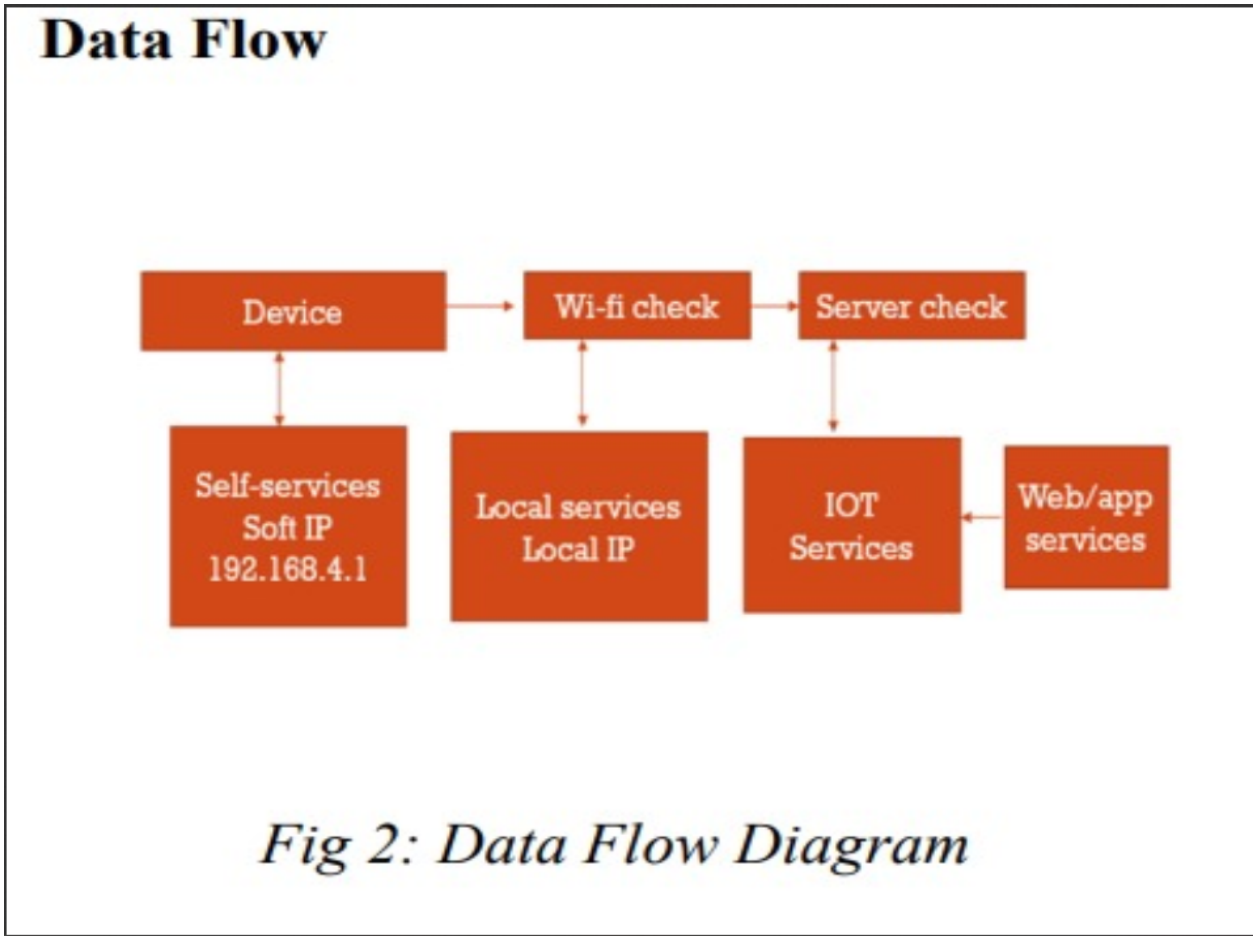
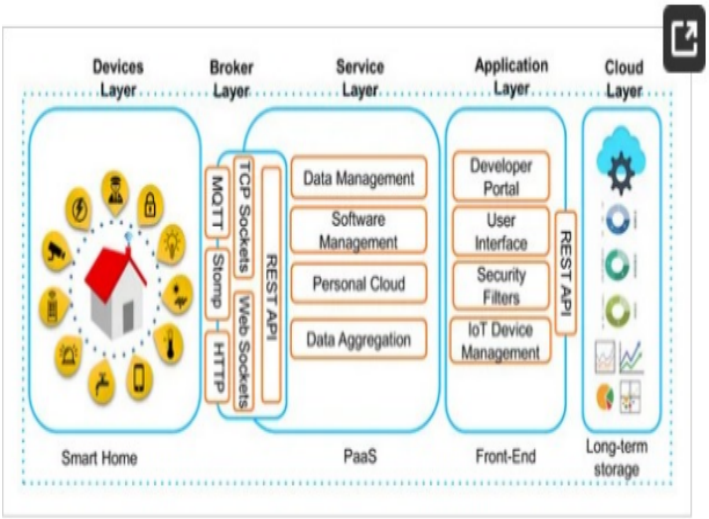
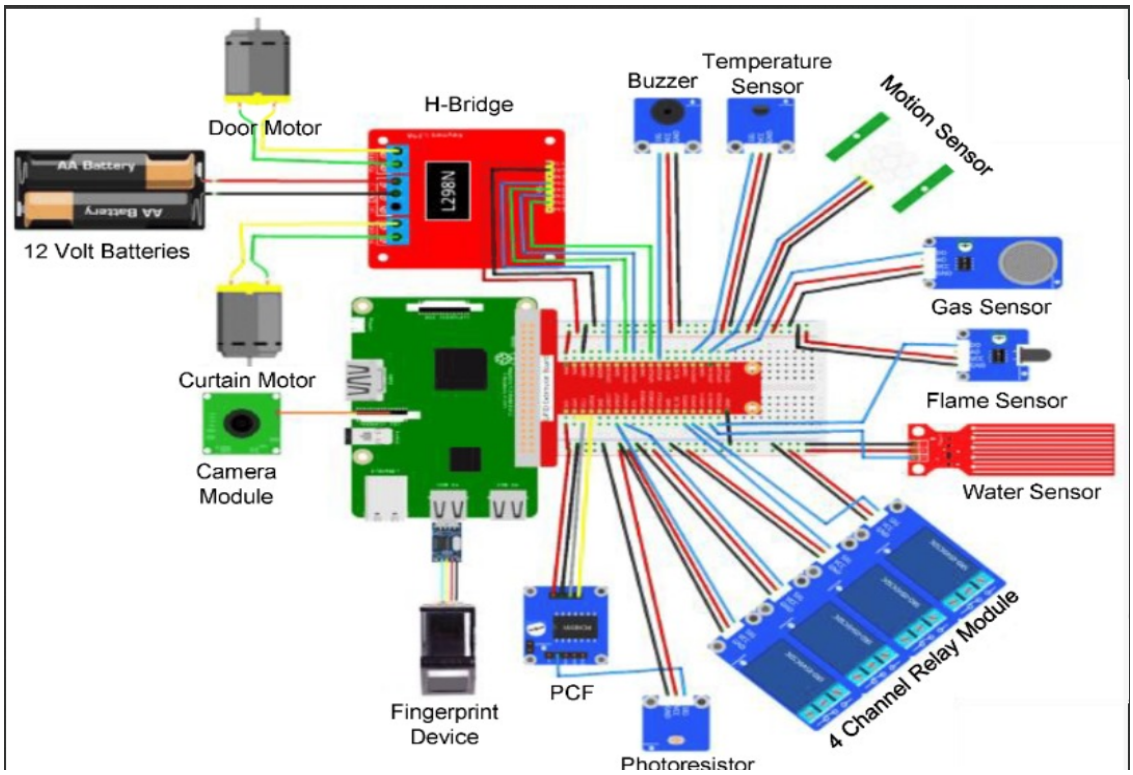
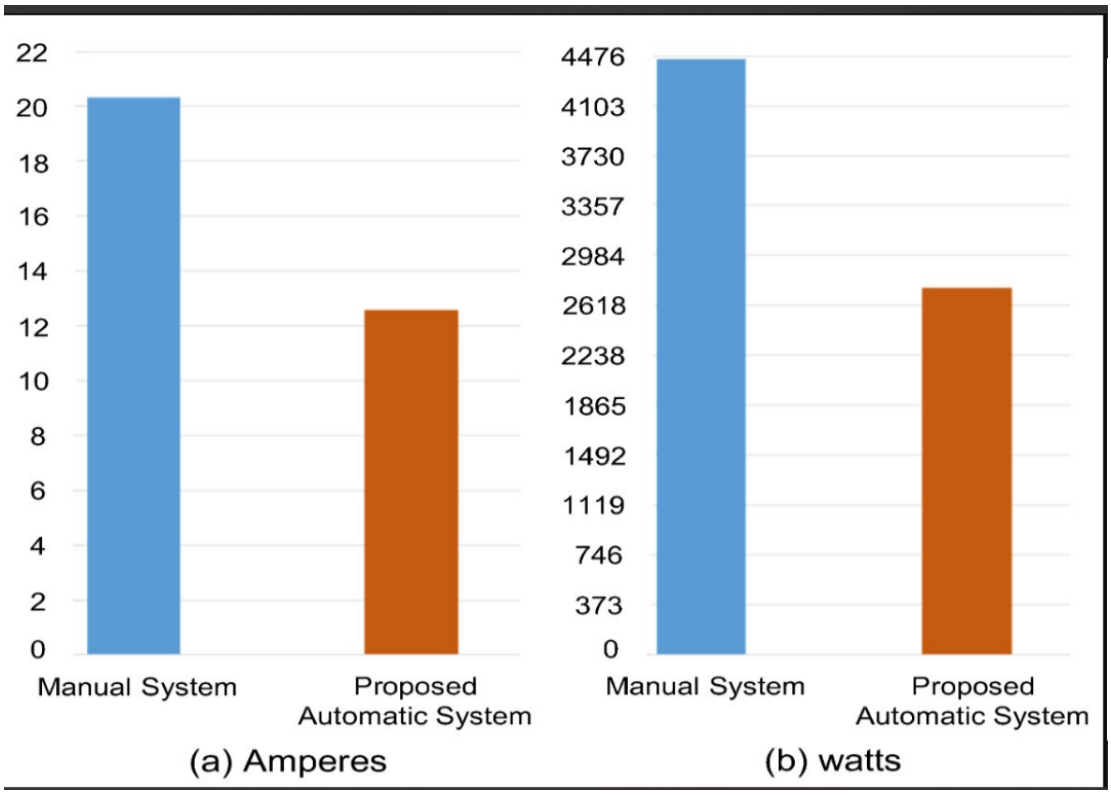


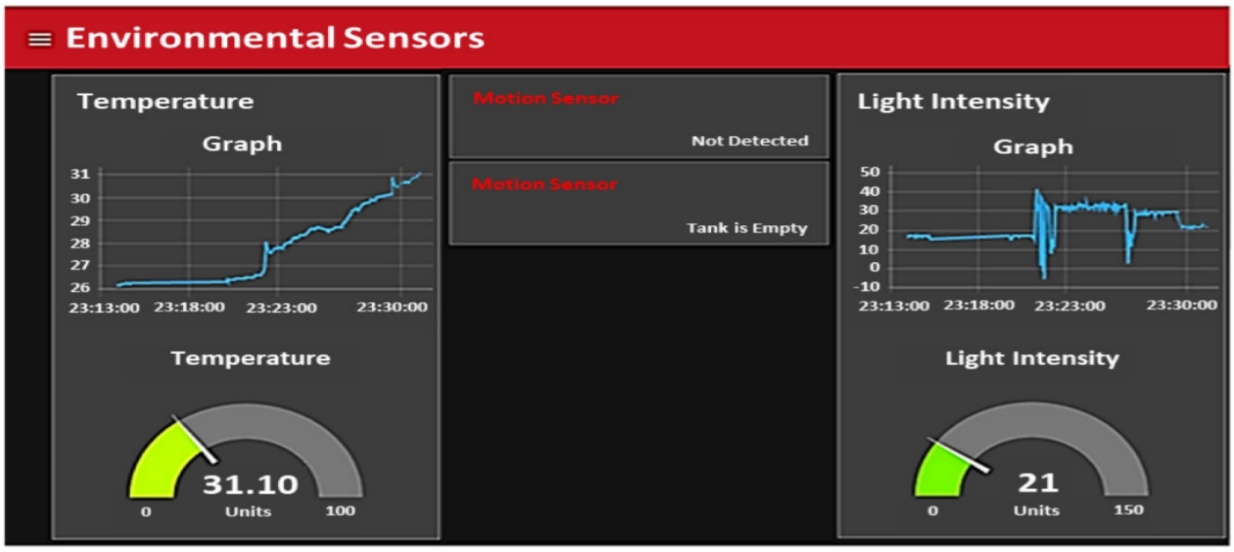
Fig 2: Data Flow Diagram



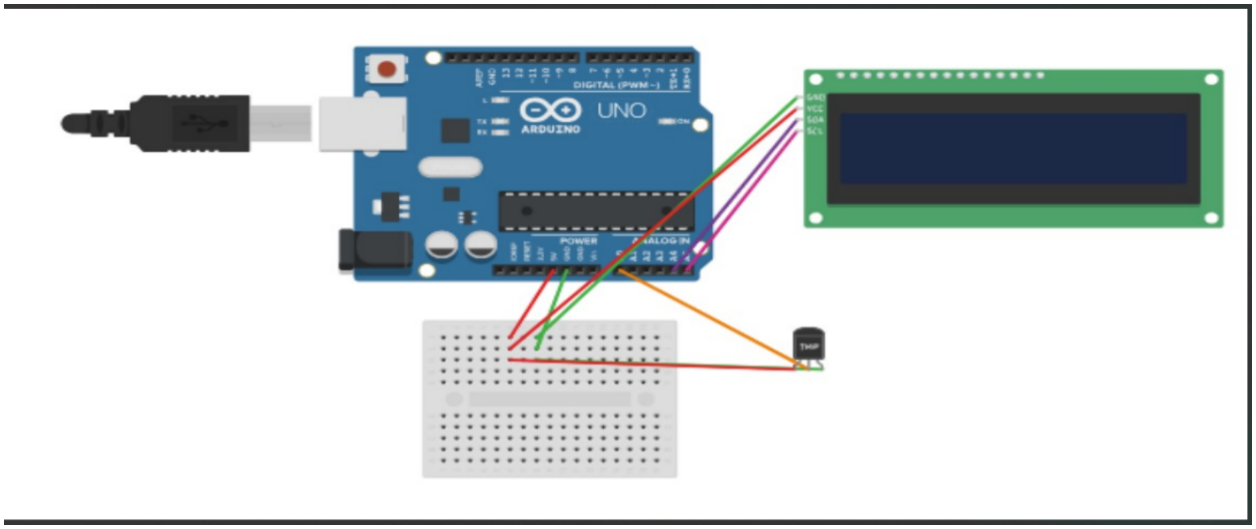
The proposed framework consists of 5 layers such as devices, service, broker, application and cloud layers.


Door Motor
H-Bridge
Buzzer
Temperature Sensor
Motion Sensor
AA Battery
AA Battery
12 Volt Batteries
L298N
Gas Sensor
Curtain Motor
Flame Sensor
Camera Module
Water Sensor
Fingerprint Device
PCF
4 Channel Relay Module
Photoresistor


22
20
18
16
14
12
10
8
6
4
2
0
Manual System
Proposed Automatic System
(a) Amperes
4476
4103
3730
3357
2984
2618
2238
1865
1492
1119
746
373
0
Manual System
Proposed Automatic System
(b) watts


Environmental Sensors
Temperature
Graph
31
30
29
28
27
26
23:13:00
23:18:00
23:23:00
23:30:00
Temperature
31.10
0
Units
100
Motion Sensor
Not Detected
Motion Sensor
Tank is Empty
Light Intensity
Graph
50
40
30
20
10
0
-10
23:13:00
23:18:00
23:23:00
23:30:00
Light Intensity
21
0
Units
150


Room Safety Sensors
Flame Detection
chart
100
80
60
40
20
0
20:45:00
20:50:00
20:59:00
Gauge
98.22
0
units
150
Detected
Smoke / Gas Detection
chart
100
80
60
40
20
0
20:45:00
20:49:00
20:56:00
Gauge
36.0473908688
0
units
150
Not Detected



CONCLUSION:

In conclusion, implementing IBM Cloud in smart home automation offers a range of valuable benefits and use cases. By leveraging the capabilities of IBM Cloud, homeowners can create a more intelligent, efficient, and secure living environment. These benefits include centralized control, data analytics, automation, scalability, enhanced security, integration with third-party services, remote access, customization, energy efficiency, support for multiple protocols, and a robust developer ecosystem. The specific advantages will depend on the individual implementation and services offered by IBM Cloud, making it essential for users to explore and understand the available options to tailor smart home automation to their needs and preferences.