Building the Internet of Things (IOT) using Raspberry Pi

TEAM MEMBER

Reg no: 510421104028

Name: DIVYA.A

Phase-3 Document submission

Project: *SMART PARKING*

INTRODUCTION:

- In this technology project you will begin building your projects by deploying IoT devices and then developing a python script on the IoT devices as per the project requirement.
- We have to develop the sensor system using raspberry pi.

OBJECTIVE:

- plan the design and deployment of IoT sensors in parking spaces to detect occupancy and availability.

- In PHASE 2 we discussed about the Image processing and their implementations.

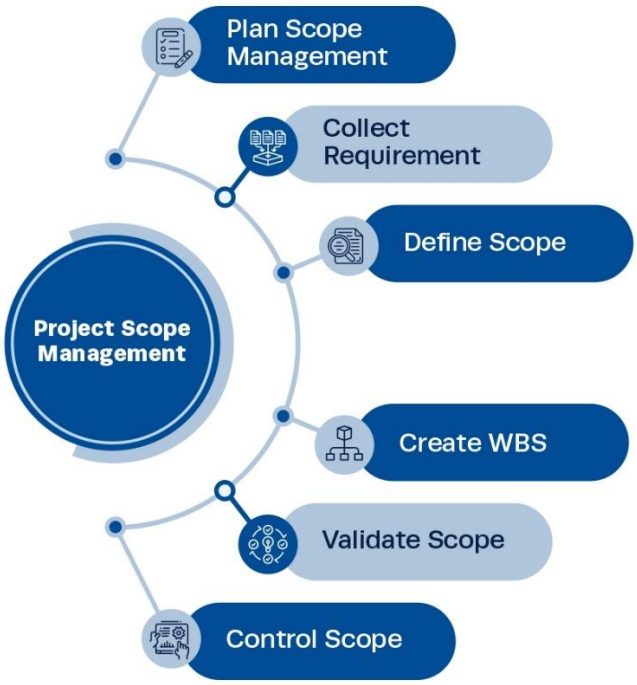
PHASE 3: Development Part 1

- Start building the IoT sensor system and raspberry pi integration
- Here is the steps for the following content:

Step 1: Define the Project Scope and Goals

- Decide what type of sensor(s) you want to use (e.g., temperature, humidity, motion).

- Determine what you want to achieve with the sensor data (e.g., monitor a room, track environmental conditions).



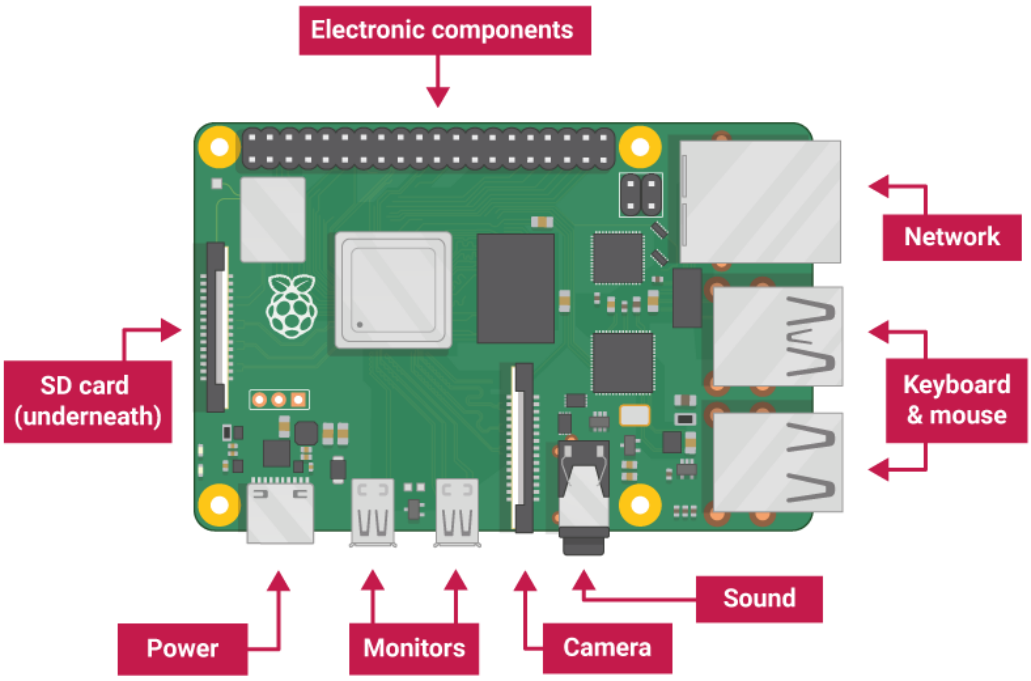
Step 2: Gather Materials

- Raspberry Pi (any model with Wi-Fi capabilities will work).
- Sensors (based on your chosen type).
- Power supply for Raspberry Pi.
- MicroSD card (8GB or more).
- Breadboard, jumper wires, and resistors (if required).
- Internet connection for the Raspberry Pi.

Step 3: Set Up Raspberry Pi

1.Install Operating System:

- Download and install a suitable OS (e.g., Raspberry Pi OS) on your microSD card.
- Insert the card into the Raspberry Pi.



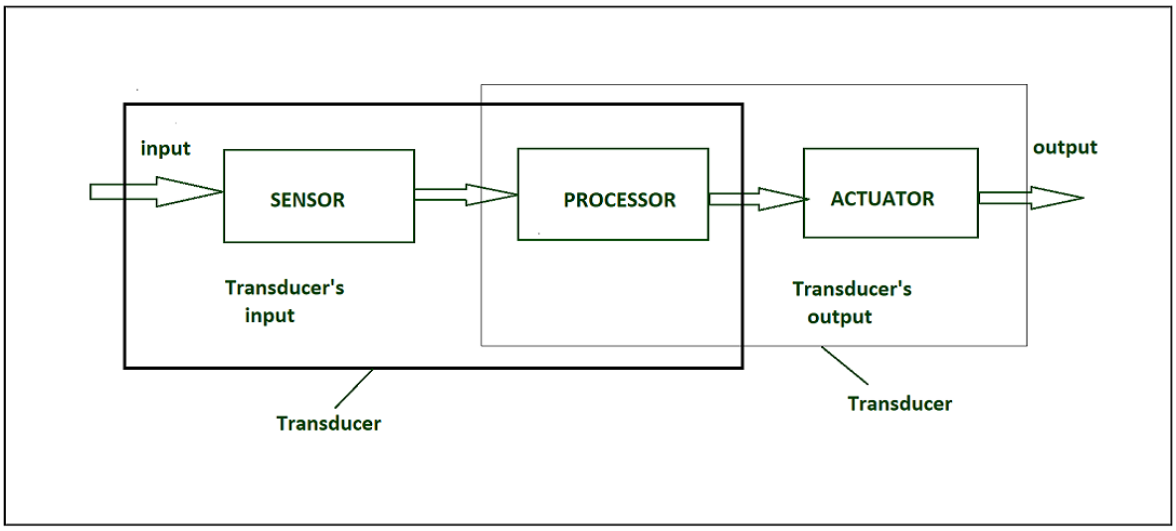
2.Boot and Configure:

- Power on the Raspberry Pi and follow the initial setup prompts.
- Connect to Wi-Fi and enable SSH for remote access.

Step 4: Connect Sensors

- Depending on the sensor(s) you chose, follow the datasheet or guide to connect them to the Raspberry Pi using the GPIO pins.

- Use breadboard and jumper wires if necessary.



Step 5: Write Code for Sensor Readings

- Choose a programming language (Python is commonly used) and write code to read data from the sensor(s).

- Use libraries or drivers provided by the sensor manufacturer.

- For example, if you're using a temperature sensor like the DHT11 with an Arduino, the code would look something like this:

Arduino Example (Temperature Sensor DHT11):

```
#include <DHT.h>

#define DHTPIN 2  // Pin where the sensor is connected
#define DHTTYPE DHT11  // Type of the sensor
```

```
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  delay(2000);  // Delay for 2 seconds

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
   }

  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");
}
```

Raspberry Pi Example (Temperature Sensor DHT22):

```
import Adafruit_DHT

DHT_PIN = 4   # Define the GPIO pin where the sensor is connected
DHT_SENSOR = Adafruit_DHT.DHT22

humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)

if humidity is not None and temperature is not None:
    print(f'Temperature={temperature:.2f}°C Humidity={humidity:.2f}%')
else:
    print('Failed to retrieve sensor data')
```

Step 6: Test Sensor Readings

Run the code to ensure that the Raspberry Pi can successfully read data from the sensor(s).

OUTPUT:

If you run this code on a Raspberry Pi with a DHT22 sensor connected to GPIO pin 4, and if the Adafruit DHT library is correctly installed, it will attempt to read data from the sensor.

If the sensor reading is successful, it will print out the temperature and humidity:

```makefile
Temperature=25.60°C  Humidity=53.40%
```

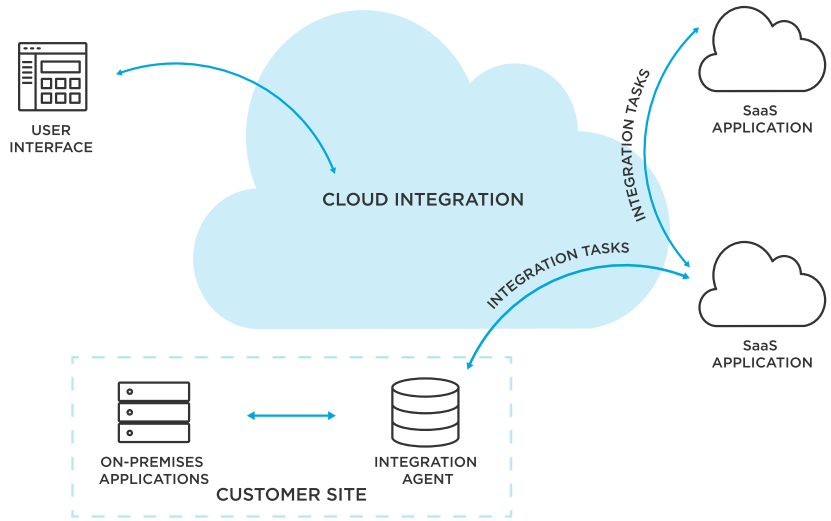
If there is an issue with the sensor or its connection, it will print:

```css
Failed to retrieve sensor data
```

indicating that it was unable to read data from the sensor. Remember, this code is designed specifically for Raspberry Pi with a DHT22 sensor connected to GPIO pin 4. If you're using a different setup, you might need to modify the pin number or use a different library depending on the sensor.

Step 7: Set Up Cloud Integration

- Choose a cloud platform for data storage and processing (e.g., AWS, Google Cloud, Azure, etc.).
- Create an account and set up a new project or application.

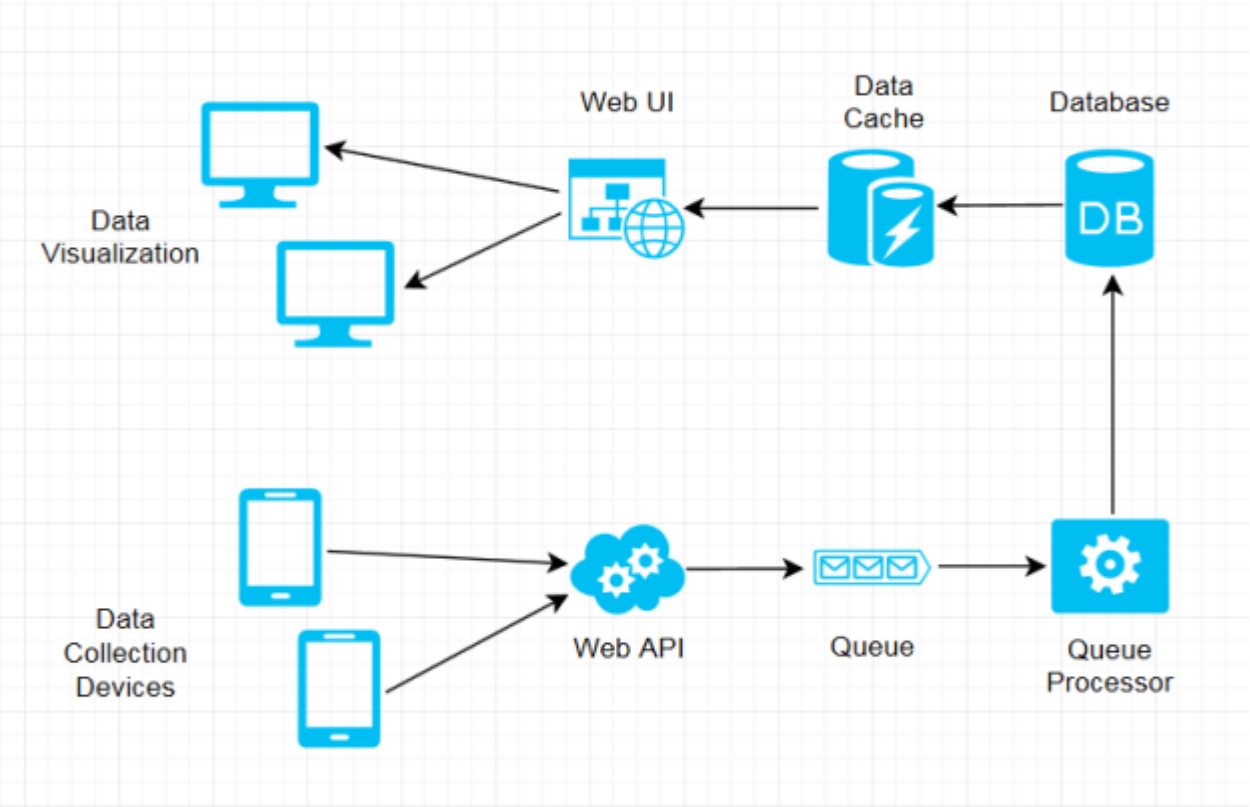


Step 8: Send Sensor Data to the Cloud

- Write code on the Raspberry Pi to send sensor readings to your chosen cloud platform.
- You may use MQTT, HTTP, or other protocols.

Step 9: Process and Visualize Data

- Set up cloud services to process and store the incoming data.
- This may involve using databases, message queues, or serverless functions.



Step 10: Monitor and Troubleshoot

- Continuously monitor your system for any issues.
- Use logging and monitoring tools to keep track of sensor readings and system health

Step 9: Documentation and Deployment

- Document your project thoroughly.
- Include information on hardware connections, software setup, and how to run the Python scripts.
- Once everything is working as expected, deploy your IoT devices in the desired locations.

CONCLUSION:

Remember, this is a high-level overview. Depending on the specific sensors and cloud platform you choose, the implementation details will vary. Always refer to the documentation provided by the manufacturers and cloud platforms for specific instructions.