INTRODUCTION TO ARDUINO PLATFORM AND PROGRAMMING

Ex.no:01
Date:

Aim:

INTRODUCTION ABOUT ARDUINO PLATFORM

Arduino Hardware Features:

1. Microcontroller: Arduino boards are driven by microcontrollers, which are the brain of the board. These microcontrollers execute the code you write and interact with various components connected to the board. For example, the Arduino Uno uses the ATmega328P microcontroller.
2. Digital and Analog I/O Pins: Arduino boards come with a set of digital pins (usually labeled D0 to D13) and analog pins (labeled A0 to A5 on the Arduino Uno). Digital pins can be used for both input and output, allowing you to read or write digital signals (HIGH or LOW). Analog pins can read analog voltages, making them suitable for sensors like light sensors or temperature sensors.
3. Voltage Regulator: Arduino boards often include a voltage regulator that allows you to power the board using a wide range of input voltages. This is useful because it makes Arduino boards adaptable to different power sources, such as batteries or external power supplies.
4. USB Interface: The USB port on Arduino boards serves multiple functions. It is used to upload your code (compiled sketches) to the board, allowing you to change the behavior of your project easily. Additionally, it provides a means for serial communication between the Arduino and your computer, facilitating data exchange and debugging.
5. LEDs: Some Arduino boards have built-in LEDs. For example, the Arduino Uno has a built-in LED on pin 13. These LEDs can be controlled by your code and are often used for simple visual feedback during development and testing.
6. Crystal Oscillator: Certain Arduino boards have a crystal oscillator. This oscillator provides precise timing for the microcontroller, ensuring accurate execution of time-sensitive tasks. It's especially important for applications that require precise timing, such as communication protocols.

Arduino Software Features:

1. Arduino IDE (Integrated Development Environment): The Arduino IDE is a software application that simplifies the process of writing, compiling, and uploading code to your Arduino board. It provides a user- friendly code editor, a compiler to convert your code into machine-readable instructions, and tools to upload the code to your board.
2. Arduino Libraries: Arduino libraries are pre-written code modules that extend the functionality of your Arduino projects. These libraries save you time by providing readymade functions and drivers for various sensors, displays, and communication modules. You can easily include libraries in your code to interact with specific hardware components.

INTRODUCTION TO PROGRAMMING

Procedure to Install Arduino IDE:

1. Download Arduino IDE: - Visit the official Arduino website (https://www.arduino.cc/en/software). - Choose the version of the Arduino IDE that matches your computer's operating system (Windows, macOS, or Linux). - Download the installer.
2. Install Arduino IDE: - Run the downloaded installer. - Follow the on-screen instructions provided by the installer to install the Arduino IDE on your computer. This typically involves choosing an installation location and creating shortcuts.
3. Launch Arduino IDE: - Once the installation is complete, you can launch the Arduino IDE from your computer's Start menu (Windows), Applications folder (macOS), or by running the executable file (Linux).
4. Select Arduino Board: - In the Arduino IDE, click on the "Tools" menu. - Under the "Board" submenu, select the specific Arduino board model you are using. This setting ensures that the IDE compiles code compatible with your board.
5. Select Port: - Still in the "Tools" menu, go to the "Port" submenu. - Select the communication port to which your Arduino board is connected. This is usually a USB port, and the connected Arduino should be recognized automatically.
6. Write and Upload Code: - Create a new sketch (your Arduino program) in the IDE or open an existing one. - Write your code in the sketch window. - Click the "Upload" button (a right-facing arrow) in the IDE's toolbar. This action compiles your code and uploads it to the connected Arduino board.
7. Monitor Serial Output (Optional): - If your Arduino code includes serial communication (using functions like `Serial.print` or `Serial.read`), you can monitor the output in the "Serial Monitor" tool, accessible from the "Tools" menu. By following these steps, you should have successfully installed the Arduino IDE, configured it for your specific board, and uploaded your code to the Arduino board for execution. This allows you to bring your projects to life with ease.

Hardware Initialization:

1. Connect Hardware: First, connect your Arduino board to your computer using a USB cable. Ensure that your power source (USB or external power supply) is correctly set.
2. Check Power: Verify that the power LED on the Arduino board is lit, indicating it has power.
3. Connect Components: If you have external components like sensors or actuators, connect them to the appropriate pins on the Arduino following the datasheets or tutorials for your specific components.

CODE FOR BLINK THE LED

1. Install Arduino IDE: If you haven't already, download and install the Arduino IDE (Integrated Development Environment) from the Arduino website.

2. Open Arduino IDE:Launch the Arduino IDE on your computer.

3. Select Board: From the "Tools" menu, select your Arduino board model under the "Board" option. Make sure you've chosen the correct board, as different boards have different specifications.

4. Select Port: Under the "Tools" menu, choose the correct port that your Arduino is connected to (e.g., COM1, COM2, /dev/ttyUSB0). This is essential for uploading code to the board.

5. Write Code:Write or open the Arduino sketch (program) that you want to upload to the board using the Arduino IDE.

Now, let's write a sample code to blink the LED connected to pin 13: arduino

```
// Define the LED pin int ledPin = 13;
void setup() {
// Initialize the LED pin as an output pinMode(ledPin, OUTPUT);
}
void loop() {
// Turn the LED on digitalWrite(ledPin, HIGH); delay(1000); // Wait for 1 second
// Turn the LED off digitalWrite(ledPin, LOW); delay(1000); // Wait for 1 second
 }
```

Conclusion:

INTERFACE ARDUINO WITH ZIGBEE MODULE

Ex.no:02
Date:

Aim:

Materials Required:

Procedure:

1. Connect the XBee module to the Arduino board:
- Connect the XBee module's VCC pin to the Arduino 5V pin.
- Connect the XBee module's GND pin to the Arduino GND pin.
- Connect the XBee module's DIN pin to Arduino digital pin 2 (RX pin).
- Connect the XBee module's DOUT pin to Arduino digital pin 3 (TX pin).
- If using an LED, connect its positive leg (longer leg) to a current-limiting resistor (220Ω) and then connect the other end of the resistor to Arduino digital pin 13. Connect the LED's negative leg (shorter leg) to GND.
2. Connect the Arduino board to your computer using the USB cable.
3. Open the Arduino IDE on your computer.
4. Write the following code in the

Arduino IDE: void setup() {

Serial.begin(9600); // Set the baud rate to 9600

}

void loop() {

if (Serial.available()) {

char data = Serial.read(); // Read the received data

// Echo the received data

Serial.print("Received data:

"); Serial.println(data);

// If the received data is '1', turn on

the LED if (data == '1') {

digitalWrite(13, HIGH);

}

// If the received data is '0', turn off the LED else if (data == '0') {

digitalWrite(13, LOW);

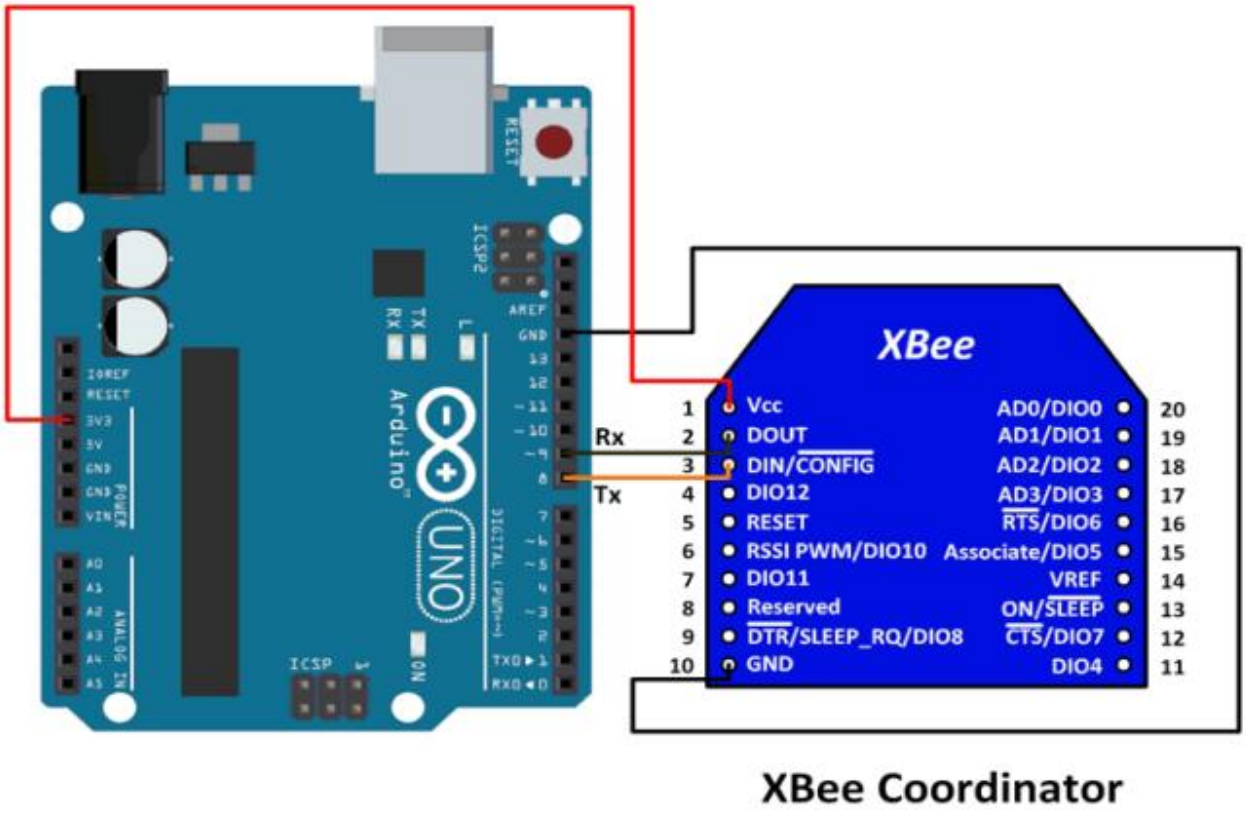
}

}

}

5. Upload the code to the Arduino board.
6. Open the Serial Monitor in the Arduino IDE.
7. Configure the XBee module:
- Connect the XBee module to your computer using an XBee USB Explorer or a USB-toSerial adapter.
- Open a serial terminal software (e.g., XCTU) on your computer.
- Find and select the COM port of the XBee module.
- Set the baud rate to 9600.
- In the serial terminal, send data by typing '1' and pressing Enter. The LED connected to Arduino pin 13 should turn on.
- Send data by typing '0' and pressing Enter. The LED should turn off.

Circuit Diagram



Conclusion:

INTERFACE ARDUINO WITH GSM MODULE

Ex.no:03
Date:

Aim:

Materials Required:

Procedure:

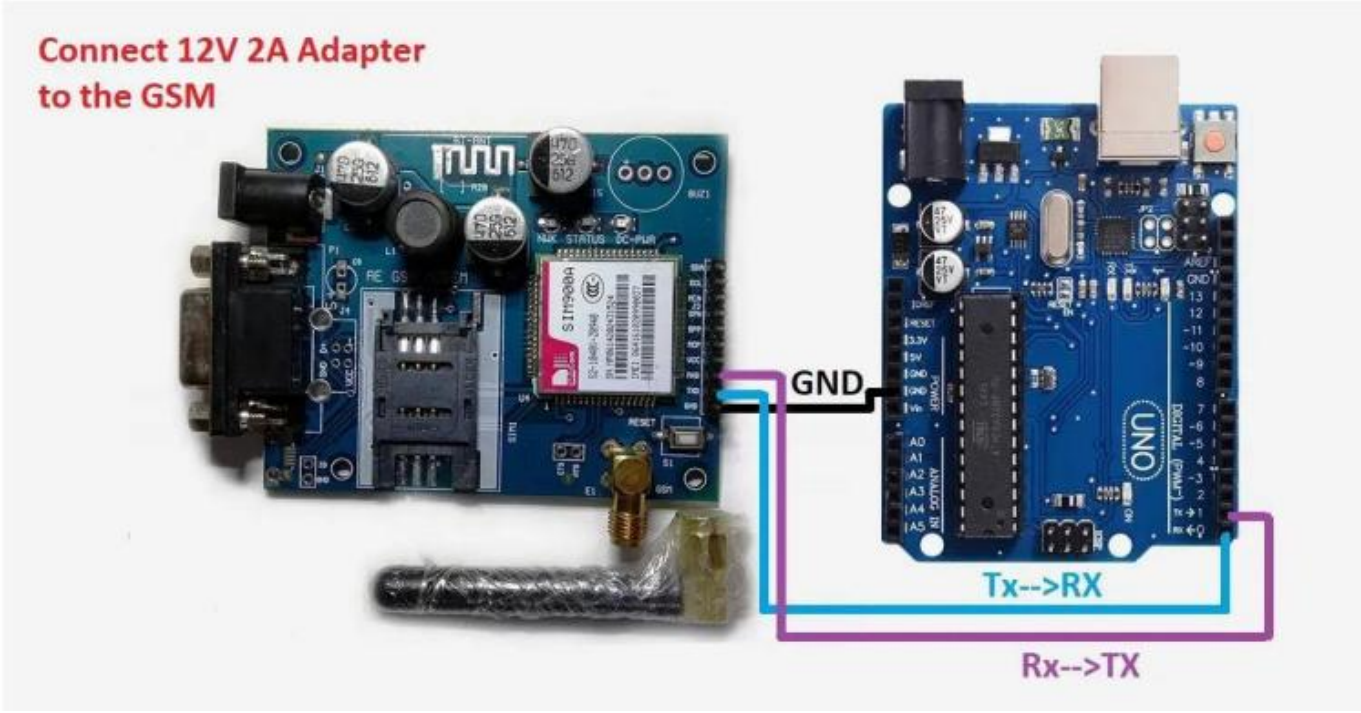
1. Connect the TX pin of the GSM module to the Arduino's digital pin 2 (RX).
2. Connect the RX pin of the GSM module to the Arduino's digital pin 3 (TX).
3. Connect the GND pin of the GSM module to the Arduino's GND pin.
4. If you want to use LEDs for indication, connect an LED to a digital pin with an appropriate resistor connected in series.

1. Open the Arduino IDE and create a new sketch.
2. Make sure you have the correct board and port selected under the Tools menu.
3. Copy and paste the following code into the sketch:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(9, 10); void setup()
```

```
{
mySerial.begin(9600); // Setting the baud rate of GSM Module Serial.begin(9600); // Setting
the baud rate of Serial Monitor (Arduino) delay(100);
}
void loop()
{
if (Serial.available()>0) switch(Serial.read())
{
case 's':
mySerial.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode delay(1000); //
Delay of 1 second mySerial.println("AT+CMGS=\"+91xxxxxxxxxx\"\r"); // Replace x with
mobile number delay(1000);
mySerial.println("Technolab creation");// The SMS text you want to send delay(100);
mySerial.println((char)26);// ASCII code of CTRL+Z for saying the end of sms to the
module delay(1000);
break; case r':
mySerial.println("AT+CNMI=2,2,0,0,0"); // AT Command to receive a live SMS
delay(1000);
break;
}
if (mySerial.available()>0) Serial.write(mySerial.read());
}
```

Circuit diagram



Conclusion:

INTERFACE ARDUINO WITH BLUETOOTH MODULE

Ex.no:04
Date:

Aim:

Materials Required:

Procedure:

1. Set up the circuit on the breadboard as follows:
- Connect the positive leg of the LED to Arduino pin 13 through the 220-ohm resistor.
- Connect the negative leg of the LED to the GND pin of the Arduino.
- Connect the VCC pin of the Bluetooth module to the 5V pin of the Arduino.
- Connect the GND pin of the Bluetooth module to the GND pin of the Arduino.
- Connect the TX pin of the Bluetooth module to the RX pin of the Arduino.
- Connect the RX pin of the Bluetooth module to the TX pin of the Arduino.

2. Connect the Arduino board to your computer via USB, and open the Arduino IDE. Go to Tools > Board and select Arduino Uno.

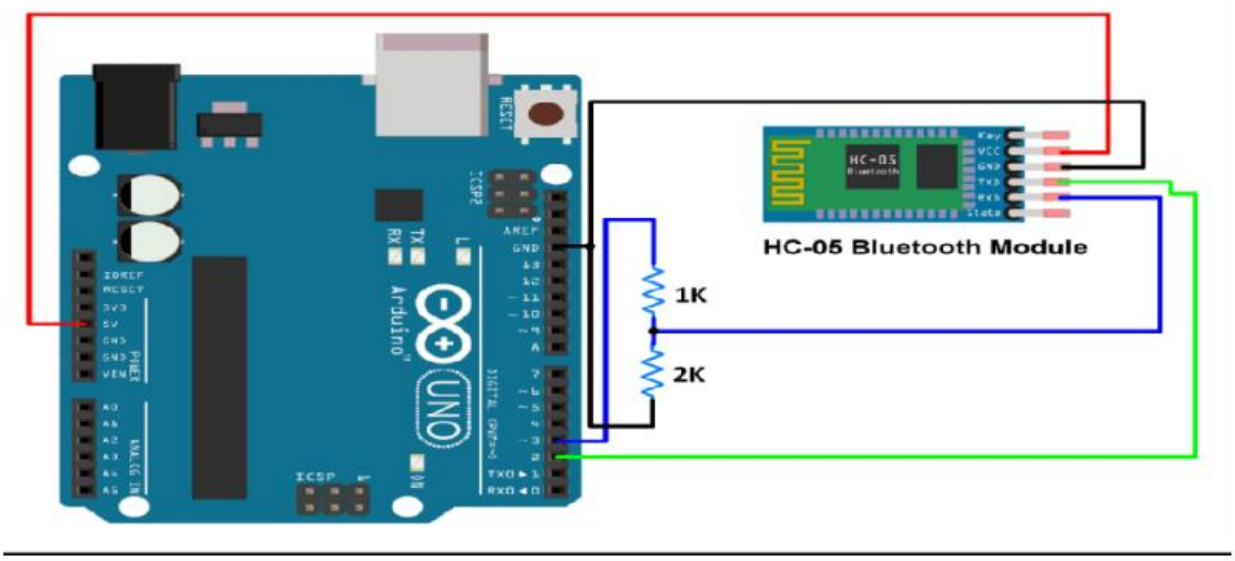
• Go to Tools > Port and select the port to which the Arduino board is connected.

• Open a new sketch and code the following

```
void setup()
{
pinMode(13, OUTPUT); // Set pin 13 as OUTPUT Serial.begin(9600); // Initialize serial communication
}
void loop()
{
if (Serial.available())
{
// Check if data is available char command = Serial.read();
// Read incoming command if (command == '1')
{
digitalWrite(13, HIGH); // Turn LED ON
}
else if (command == '0') { digitalWrite(13, LOW); // Turn LED OFF
}
}
}
```

• Upload the code to the Arduino board.
• Open a Serial Monitor by going to Tools > Serial Monitor.
• Set the baud rate to 9600 in the Serial Monitor.
• Disconnect the USB cable from your computer and connect the Arduino to an external power source (e.g., a battery or power supply).
• Install a Bluetooth terminal application on your smartphone or computer (e.g., "Serial Bluetooth Terminal" for Android or "Serial" for iOS).
• Enable Bluetooth on your smartphone or computer and pair it with the Bluetooth module (use the default password '1234' if prompted).
• Open the Bluetooth terminal application and connect to the Bluetooth module.
• In the terminal, send '1' to turn the LED ON or '0' to turn the LED OFF.
• Observe that the LED toggles ON/OFF according to the commands sent via Bluetooth

Circuit Diagram:



Conclusion:

INTRODUCTION TO RASPBERRY PI PLATFORM AND PYTHON PROGRAMMING

Ex.no:05
Date:

Aim:

Materials Required:

Procedure:

1. Connect the ESP8266 to the Raspberry Pi:
- Connect the GND pin of ESP8266 to any GND pin on Raspberry Pi.
- Connect the VCC pin of ESP8266 to the 3.3V pin on Raspberry Pi.
- Connect the TX pin of ESP8266 to the RX pin on Raspberry Pi.
- Connect the RX pin of ESP8266 to the TX pin on Raspberry Pi.
2. Install required software on Raspberry Pi:
- Open the terminal on Raspberry Pi.
- Update the system package list: `sudo apt-get update`

- Install the minicom package: `sudo apt-get install minicom`
3. Configure UART on Raspberry Pi:
- Open the terminal and enter: `sudo raspi-config`
- Use the arrow keys to navigate to "Interfacing Options" and press Enter.
- Select "Serial" and disable the login shell, then enable the serial port hardware.
- Exit the configuration tool.
4. Test the serial communication:
- Connect to ESP8266 using minicom: `minicom -b 115200 -o -D /dev/ttyAMA0`
- Type any command and verify if you receive a response from the ESP8266.
5. Write a Python script to interface with ESP8266:
-Create a new Python script: `nano esp_serial.py`
- Import the required modules:

Python
import serial import time
- Open the serial port for communication and send commands to ESP8266:

python

ser = serial.Serial('/dev/ttyAMA0',115200) time.sleep(1)
ser.write("AT\r ")
response = ser.readline() print response ser.close()

- Save the script and exit the nano editor.

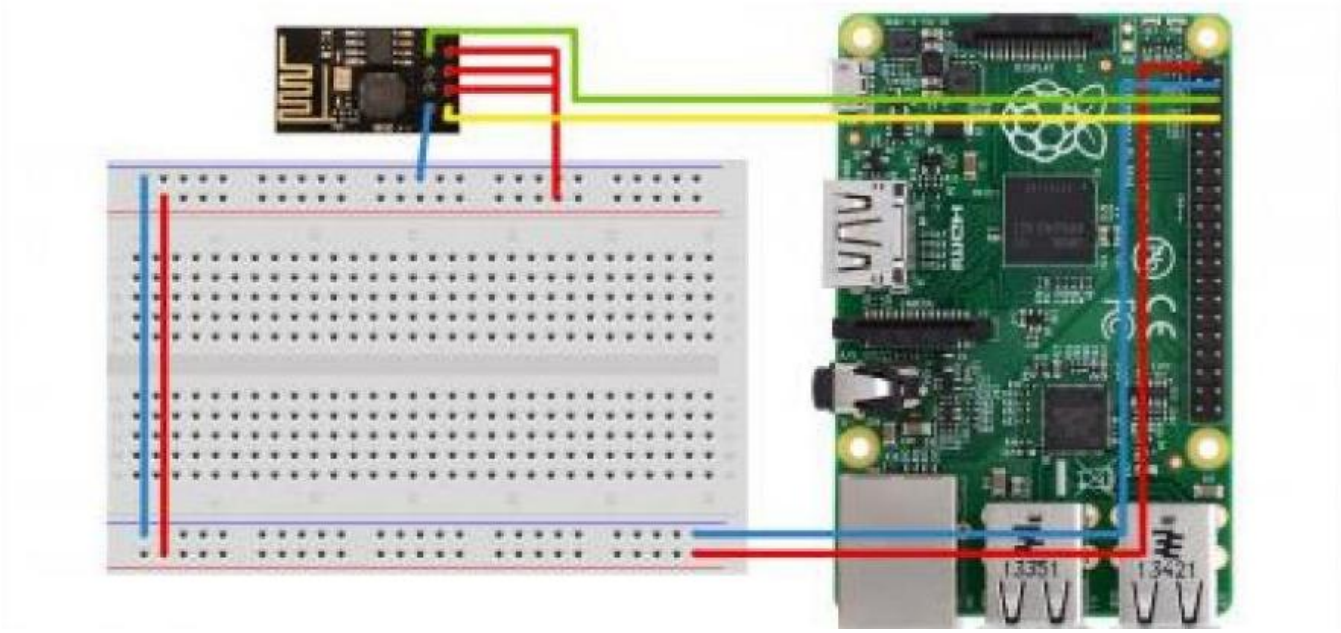
6. Run the Python script:

- Make the script executable: `chmod +x esp_serial.py`

- Run the script: `./esp_serial.py`

The Python script will open the serial port, send the "AT" command to the ESP8266, and display the response received from the ESP8266 on the Raspberry Pi's console.

Circuit Diagram:



Conclusion:

INTERFACING IR SENSOR TO RASPBERRY PI

Ex No: 06
Date:

AIM:

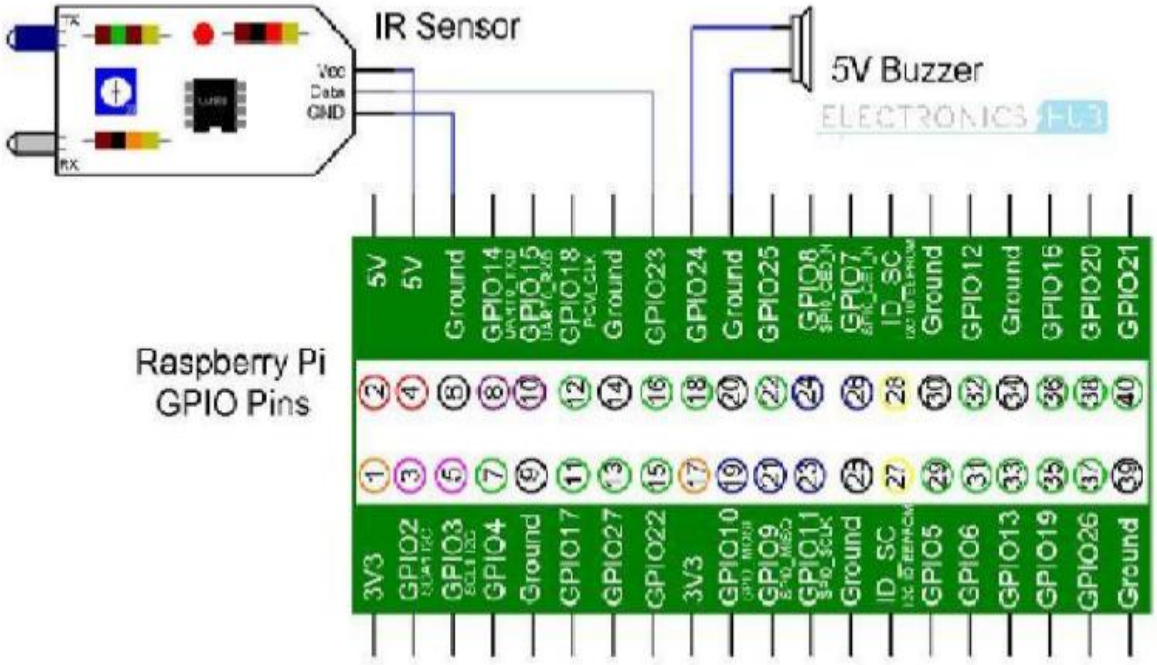
MATERIALS REQUIRED:

PROCEDURE

1. Gather the necessary components: Raspberry Pi (Any model), breadboard, jumper wires,
   IR sensor module (e.g., TSOP1738 or KY-022 module), resistors (220Ω and 10kΩ), LED,
   and a power supply for the Raspberry Pi.
   2. Connect the components on the breadboard as follows:
   - Connect the VCC pin of the IR sensor module to the 3.3V pin on the Raspberry Pi.
   - Connect the GND (ground) pin of the IR sensor module to the GND pin on the

Raspberry Pi.
- Connect the OUT (signal) pin of the IR sensor module to any GPIO pin on the RaspberryPi. (Note down the GPIO pin number used for future reference).
3. Connect a 220Ω resistor to the anode (longer leg) of the LED, and connect the other leg of the resistor to any GPIO pin on the Raspberry Pi.
- Connect the cathode (shorter leg) of the LED to the GND pin on the Raspberry Pi.
4. Connect a 10kΩ resistor between the VCC and OUT pins of the IR sensor module to pull up the voltage level.
5. Power up the Raspberry Pi using the power supply.
6. Boot into the operating system on your Raspberry Pi (preferably Raspbian).
7.Open a terminal window and update the system by running the following commands: sudo apt-get update
sudo apt-get upgrade
8. Install pigpio library for Python by running the command:
sudo apt-get install pigpio
9.Write a Python code to interface with the IR sensor and LED. Here's an example code: python
import pigpio
IR_PIN = 18 # Replace with the GPIO pin number used for OUT pin of IR sensor module
pi = pigpio.pi()
pi.set_mode(IR_PIN, pigpio.INPUT)
pi.set_pull_up_down(IR_PIN,
pigpio.PUD_UP) while True:
if pi.read(IR_PIN):
print("Obstacle Detected") # Code to turn on the LED
else:
print("No Obstacle Detected") # Code to turn off the LED
10. Save the code to a file with a .py extension (e.g., ir_sensor.py) and execute the code
by running the command:
python ir_sensor.py
11. Observe the output in the terminal window, where it will display "Obstacle Detected"
or "No Obstacle Detected" based on the presence or absence of an object in front of the IR
sensor.
12. Additionally, the code can control the LED (connected to a GPIO pin) to turn on/off based on the output of the IR sensor.

CIRCUIT DIAGRAM



CONCLUSION

COMMUNICATE BETWEEN ARDUINO AND RASPBERRY PI USING ANY WIRELESS MEDIUM

Ex No: 07
Date:

AIM

MATERIALS REQUIRED

PROCEDURE

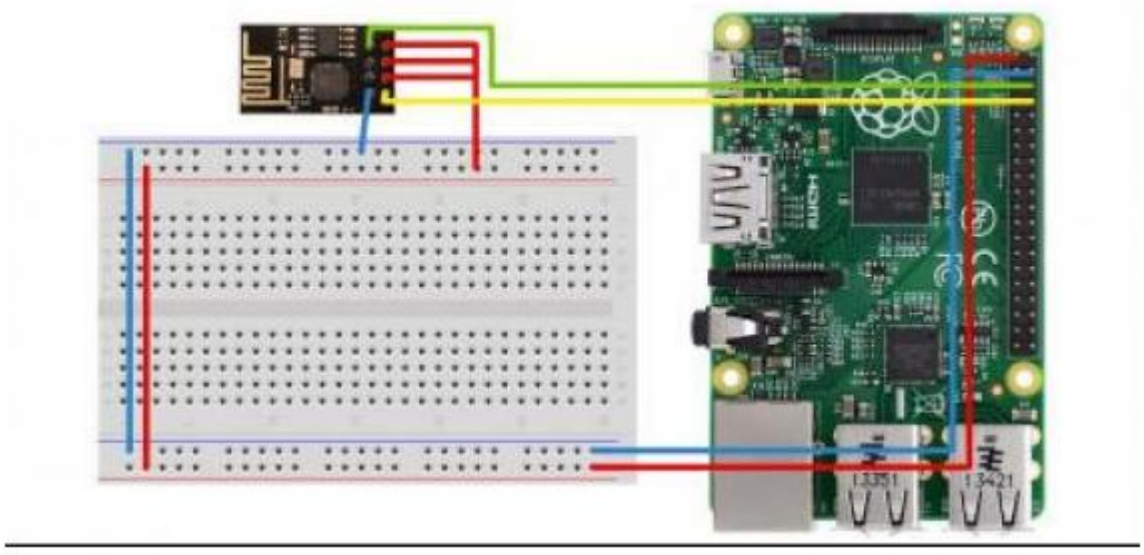
1. Connect the Arduino to the Raspberry Pi using jumper wires.
- Connect the GND pin of the Arduino to any GND pin of the Raspberry Pi.
- Connect the 5V pin of the Arduino to any 5V pin of the Raspberry Pi.
- Connect the TX pin of the Arduino to the RX pin of the Raspberry Pi (GPIO pin 10).
- Connect the RX pin of the Arduino to the TX pin of the Raspberry Pi (GPIO pin 8).
2. Install the Arduino IDE on the Raspberry Pi by running the following command in the terminal:
$ sudo apt-get update
$ sudo apt-get install Arduino
3. Open the Arduino IDE and upload a sketch to the Arduino Uno board. For example, you can use the "Blink" sketch that is available in the Examples folder of the Arduino IDE.

This will make an LED connected to pin 13 of the Arduino blink.

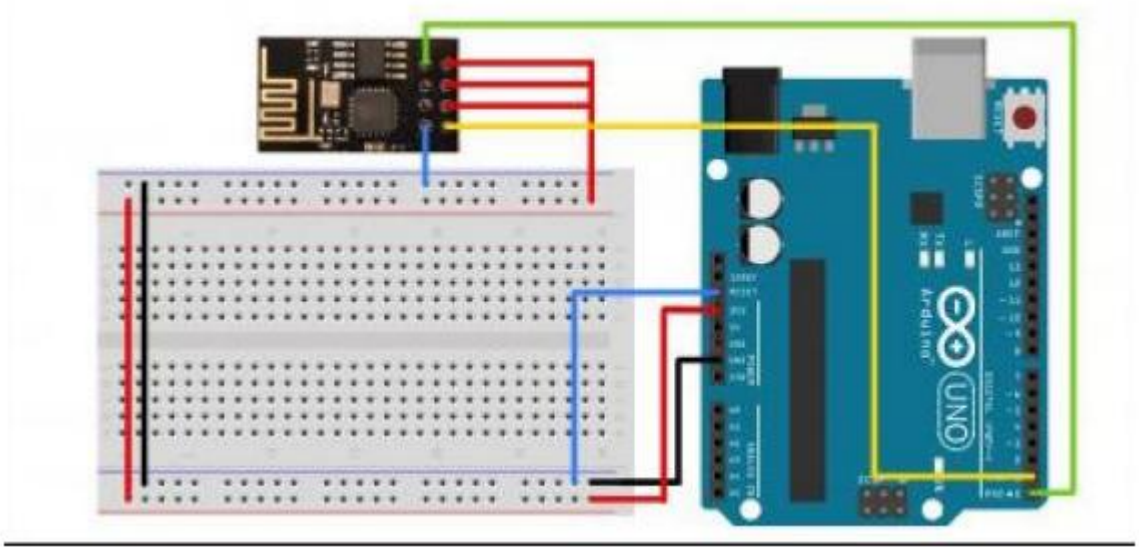
4. On the Raspberry Pi, create a Python script to communicate with the Arduino. Here's an example script: python
import serial
ser = serial.Serial('/dev/ttyAMA0', 9600) # Change the port if
necessary while True:
if ser.in_waiting > 0:
data =ser.readline().decode().rstrip()
print("Received data:", data)
This script reads data from the serial port and prints it to the terminal.
5. Run the Python script on the Raspberry Pi using the following command:
$ python3 arduino_serial.py
6. If everything is set up correctly, you should see the output from the Arduino sketch ("Blink") being printed on the terminal of the Raspberry Pi.

CIRCUIT DIAGRAM

ESP8622 WIFI MODULE WITH RASPBERRY PI



ESP8622 WIFI MODULE WITH ARDUINO UNO



CONCLUSION

SETUP A CLOUD PLATFORM TO LOG THE DATA

Ex No: 08
Date:

AIM

PROCEDURE

1. Sign up for a ThingSpeak account: Go to the ThingSpeak website (https://thingspeak.com) and click on "Sign Up" to create a new account. Fill in the required details and create a username and password.
2. Create a new Channel: After logging in to your ThingSpeak account, click on "Channels" in the top navigation menu, then select "New Channel". Fill in the necessary details such as the name, description, and field names for your Channel. Click "Save Channel" to create it.
3. Obtain API keys: Once your Channel is created, click on the "API Keys" tab. Here, you will find the Read and Write API keys, which you will need for accessing your Channel programmatically.
4. Set up your microcontroller or IoT device: Connect your microcontroller or IoT device to the internet and make sure it has the necessary libraries or firmware to connect to ThingSpeak. If using a development board like an Arduino or Raspberry Pi, you may need to install specific libraries for ThingSpeak integration.
5. Write and deploy code: Depending on your microcontroller or IoT device, write code that reads data from sensors or other sources and sends it to ThingSpeak using the appropriate API and your Write API key. Include any necessary authentication and formatting for sending data. Deploy the code to your device.
6. Monitor your Channel: After deploying your code, go back to your ThingSpeak account and click on your Channel. Here, you can monitor incoming data and view it in real-time using the Channel's visualizations and graphs.
7. Access your data programmatically: Use the Read API key obtained earlier to retrieve data from your Channel programmatically. You can use web requests or specific libraries or SDKs available for your programming language to access and process the data as needed.

8. Analyze and visualize your data: ThingSpeak offers built-in visualization tools to analyze and visualize your data. You can create custom visualizations, set up alerts, or analyze trends based on the data in your Channel using the ThingSpeak Analytics features.

9. Integrate with other platforms: ThingSpeak integrates with various cloud platforms and services, allowing you to send data to other systems or trigger actions based on certain conditions. Explore the available integrations or use the ThingSpeak API to connect with other platforms of your choice.

CONCLUSION

LOG DATA USING RASPBERRY PI AND UPLOAD TO THE CLOUD PLATFORM

Ex No: 09
Date:

AIM

MATERIALS REQUIRED

PROCEDURE

1. Connect the VCC pin of the IR sensor to the 5V pin on the Arduino.
2. Connect the GND pin of the IR sensor to the GND pin on the Arduino.
3. Connect the OUT pin of the IR sensor to digital pin 2 on the Arduino.

Arduino Code:

1. Open the Arduino IDE on your computer.

2. Connect the Arduino board to your computer using the USB cable.
3. Copy and paste the following code into the Arduino IDE:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h> #include <ThingSpeak.h>
#define SSID "your_wifi_ssid"
#define PASSWORD "your_wifi_password" #define API_KEY
"your_thingspeak_api_key"
const int sensorPin = 2;
WiFiClient client;
const char *server = "api.thingspeak.com"; void setup() {
pinMode(sensorPin, INPUT); Serial.begin(115200); WiFi.begin(SSID, PASSWORD);
while (WiFi.status() != WL_CONNECTED) { delay(1000);
Serial.println("Connecting to WiFi...");
}
Serial.println("Connected to WiFi."); ThingSpeak.begin(client);
}
void loop() {
int sensorValue = digitalRead(sensorPin); if (sensorValue == HIGH) {
Serial.println("Object
Detected"); ThingSpeak.writeField(1, 1, API_KEY);
} else {
Serial.println("No Object Detected"); ThingSpeak.writeField(1, 0, API_KEY);
}
delay(1000);
}
```

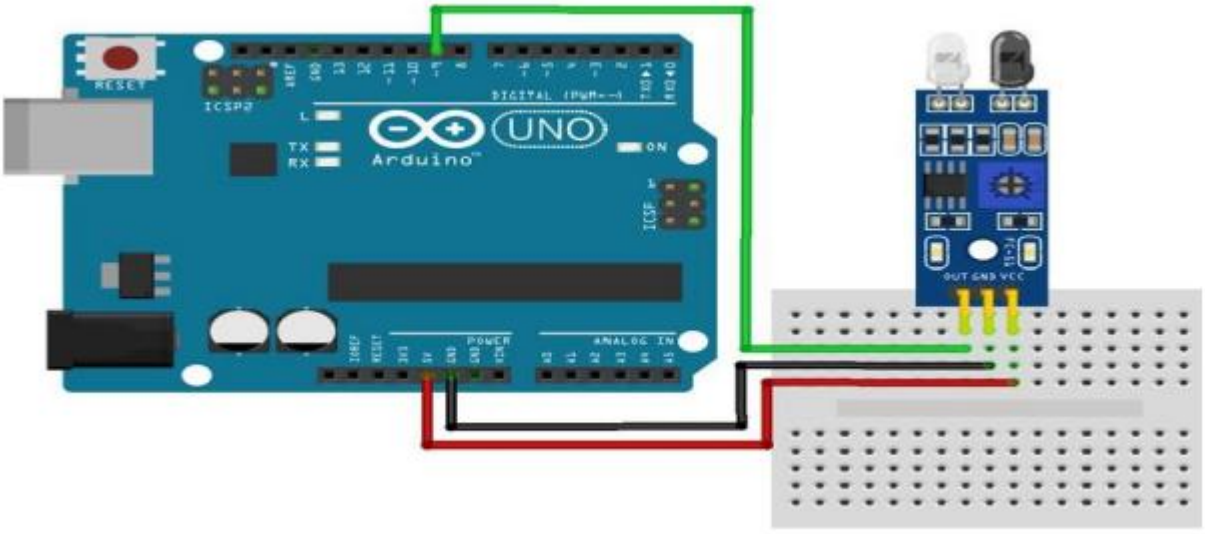
4. Replace "your_wifi_ssid" and "your_wifi_password" with your Wi-Fi credentials.
5. Replace "your_thingspeak_api_key" with your ThingSpeak API key (can be found in your ThingSpeak ccount).
6. Verify and upload the code to the Arduino board.

ThingSpeak Setup:

1. Open a web browser and go to thingspeak.com.
2. Sign in or create a new account (if not already done).
3. Click on "Channels" in the top menu and then click "New Channel".
4. Enter a name and select a field name, e.g., "Object Detected".
5. Set the "Field 1" type to "Numeric" and click "Save Channel".
6. Copy the generated "Channel ID" and paste it in the `ThingSpeak.writeField(1, 1, API_KEY)` and ThingSpeak.writeField(1, 0, API_KEY)` lines of the Arduino code.
7. Go to the "API Keys" tab in your ThingSpeak account and copy the "Write API Key".

8. Paste the write API key into the `#define API_KEY "your_thingspeak_api_key"` line of the Arduino code.
Run the Experiment:
1. Make sure the Arduino is connected to your computer with the USB cable.
2. Open the Serial Monitor in the Arduino IDE to see the IR sensor readings.
3. Power on the Arduino board.
4. The Arduino will connect to your Wi-Fi and start sending data to your ThingSpeak channel.
5. Go to your ThingSpeak account and check the "Field 1" chart to see the IR sensor data being plotted in real- time.

CIRCUIT DIAGRAM



CONCLUSION

INTERFACE RASPBERRY PI WITH TEMPERATURE AND HUMIDITY SENSOR

Ex No: 10
Date:

AIM

MATERIALS REQUIRED

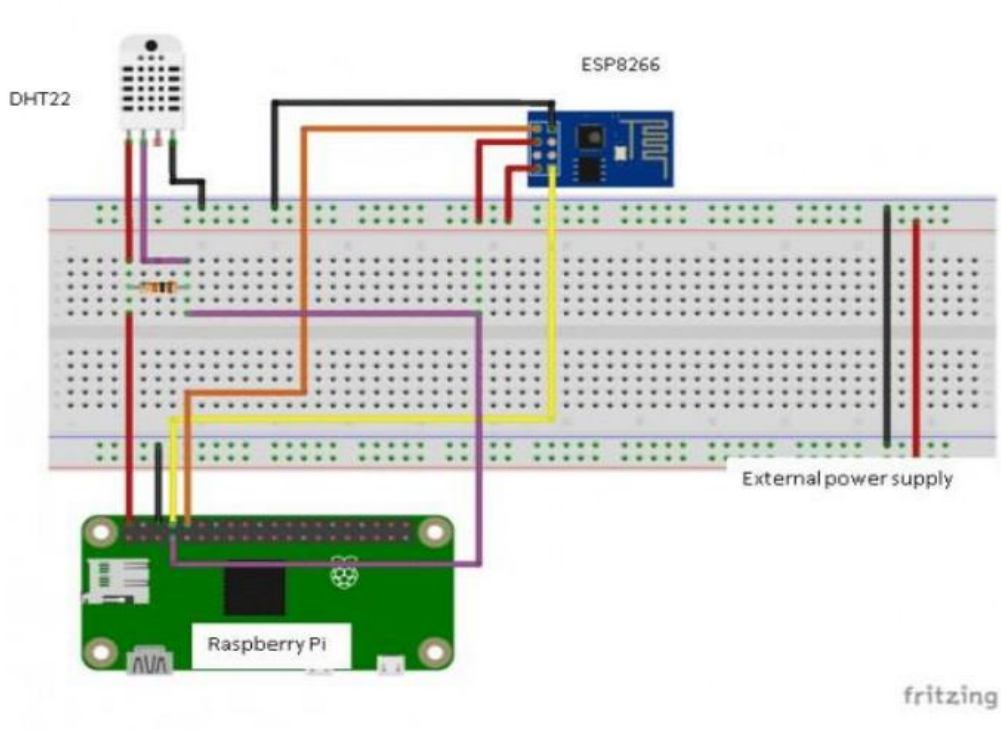
PROCEDURE

1. Connect the DHT22 or DHT11 sensor to the Raspberry Pi using the breadboard and jumper wires. The humidity and temperature data will be read using a GPIO pin on the Raspberry Pi.

- Connect the VCC pin of the sensor to the 3.3V pin on the Raspberry Pi.
- Connect the GND pin of the sensor to any ground (GND) pin on the Raspberry Pi.
- Connect the data pin of the sensor to any GPIO pin (e.g., GPIO18) on the Raspberry Pi.

2. Install the necessary software libraries and packages on the Raspberry Pi.
- Open the terminal on the Raspberry Pi.
- Install the Adafruit DHT library by entering the following command:
sudo pip3 install Adafruit_DHT
- Wait for the installation to complete.
3. Write and run a Python script to read the humidity and temperature values from the sensor.
- Create a new Python script (e.g., `humidity_temp_sensor.py`) on the Raspberry Pi using a text editor or the IDE of your choice.
- Import the necessary libraries at the beginning of the script:
python
import Adafruit_DHT
- Define the GPIO pin number you connected the sensor to:
python
sensor_pin = 18
- Inside a loop, read the humidity and temperature values from the sensor
and print them: python
while True:
humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.DHT22, sensor_pin)
if humidity is not None and temperature is not None:
print('Temperature: {0:0.1f}°C Humidity: {1:0.1f}%'.format(temperature,
humidity)) else:
print('Failed to read data from sensor. Try again!')
- Save the script and run it by executing the following command in
the terminal: python3 humidity_temp_sensor.py
4. Observe the humidity and temperature values being printed in the terminal. They should update periodically according to the loop in the Python script.

CIRCUIT DIAGRAM



CONCLUSION

MINI PROJECT

Iot Drone For Crop Health Monitoring

Iot Drone For Crop Health Monitoring

Aim:

To design and implement an IoT-enabled drone system for real-time crop health monitoring using multiple sensors, cameras, and cloud-based analytics. The objective is to support precision agriculture by enabling early disease detection, monitoring environmental conditions, and providing actionable insights to farmers.

Materials Required:

Hardware:

- Quadcopter drone frame (2–3 kg payload capacity)
- Li-Po batteries (25–30 min flight time)
- GPS module with RTK support
- Raspberry Pi 4 (8GB RAM)
- Arduino Uno
- High-resolution camera (12 MP) with gimbal
- Soil moisture sensor (capacitive type)
- DHT22 (temperature & humidity sensor)
- MQ-135 air quality sensor
- Light intensity sensor (LDR)
- pH sensor (for soil testing)
- 4G/LTE module and Wi-Fi module
- 128 GB microSD card
- Power management system with voltage regulators
- Weatherproof housing

Software:

- Python 3.8+
- Arduino IDE
- OpenCV (image processing)
- TensorFlow/Keras (machine learning models)
- AWS IoT Core / Google Cloud IoT
- MQTT protocol
- Flask/Django (REST API)
- React Native / Flutter (mobile app)
- React.js (web dashboard)
- PostgreSQL / MongoDB (database)
- Raspbian OS (Raspberry Pi)

Procedure:

Phase 1: Hardware Integration and Setup

1. Assemble the quadcopter frame with motor, battery, GPS module, and propellers.
2. Mount Raspberry Pi 4 as the onboard computer, ensuring proper heat dissipation and vibration dampening.
3. Connect Arduino Uno to the sensors (soil moisture, DHT22, MQ-135, LDR, and pH sensor).
4. Install the 12MP high-resolution camera with gimbal stabilization for clear image capture.
5. Set up the power distribution system with voltage regulators to maintain stable voltage across all sensors and components.
6. Place the sensor modules and camera in weatherproof housings for protection during outdoor operations.
7. Calibrate each sensor using reference values (e.g., soil moisture with standard soil probes, DHT22 with laboratory thermometer/hygrometer).
8. Test GPS accuracy and configure RTK base station for high precision in navigation.

Phase 2: Software Development

1. **Embedded Programming**
    - Write Arduino programs to collect sensor readings and send them to Raspberry Pi.
    - Develop Python-based control scripts for autonomous drone navigation using GPS waypoints.
    - Implement onboard preprocessing such as sensor validation, noise filtering, and image correction.
    - Add local backup system (microSD card) to prevent data loss in case of connectivity failure.
2. **Machine Learning Model**
    - Create a dataset of crop images (healthy vs diseased).
    - Train a Convolutional Neural Network (CNN) in TensorFlow/Keras for disease detection.
    - Optimize model for edge deployment on Raspberry Pi (lightweight version).
    - Validate model accuracy using cross-validation techniques.

Phase 3: Cloud Infrastructure Setup

1. Configure AWS IoT Core or Google Cloud IoT to register the drone as a device.
2. Enable secure communication using device certificates and encryption.
3. Set up MQTT protocol for lightweight and reliable transmission of sensor + image data.
4. Build a cloud-based data ingestion pipeline to process incoming data in real time.
5. Design database schema (PostgreSQL/MongoDB) to store historical data for analytics.
6. Implement predictive analytics (yield prediction, disease outbreak alerts) using Apache Kafka/Spark.

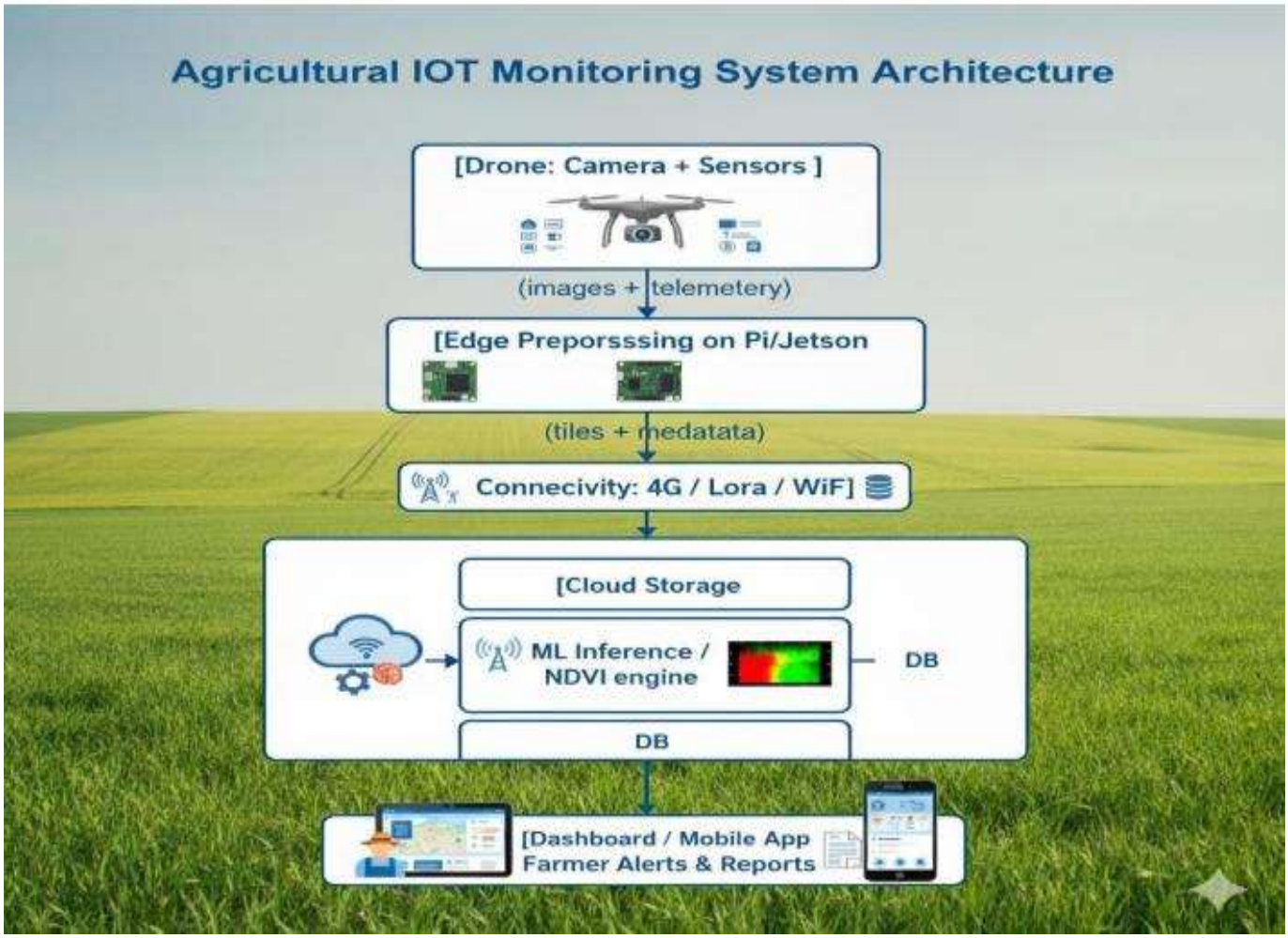
Phase 4: User Interface Development

1. Develop a mobile application (React Native/Flutter) that provides:
    - Real-time sensor data (soil moisture, temperature, air quality, etc.)
    - Image-based disease detection results
    - Alerts and notifications (e.g., pest infestation, low moisture levels)
    - Offline capability for rural areas with poor internet connectivity
2. Create a web dashboard (React.js) for advanced analytics:
    - Visual representation of farm health using color-coded maps
    - Historical trend analysis (yield, disease frequency, soil health)
    - Report generation for farmers and agricultural experts
    - Role-based access (e.g., farmer, agronomist, administrator)

Phase 5: System Integration and Testing

1. Conduct end-to-end testing by flying the drone across farmland using pre-programmed GPS paths.
2. Collect environmental data and crop images in real time.
3. Verify system reliability by comparing drone sensor data with manual ground-truth readings.
4. Test cloud connectivity, ensuring data is transmitted and processed without delay.
5. Validate disease detection accuracy by comparing system predictions with expert assessments.
6. Evaluate performance under different conditions (sunny, cloudy, windy).
7. Deploy system in real farm conditions, collect farmer feedback, and refine system features.

System Flow Diagram:



Results:

1. Flight operations & data collection

- Completed autonomous missions covering 50 hectares in ~45 minutes per mission.
- Mission completion rate: 95% (automatic return-to-home triggered on low battery).
- Data volume per flight: >10,000 data points (GPS-tagged sensor readings + high-resolution images).
- Stable telemetry and control maintained up to ~2 km operational range.

2. Sensor accuracy & reliability

- Soil moisture: Correlation with ground-truth probes ≈ 92%.
- Temperature: ±0.5°C; Humidity: ±3% under field conditions.
- Air-quality (MQ-135): Detected CO_2 variations consistent with photosynthetic activity patterns.
- Camera provided sufficient resolution (12MP) for canopy-level and individual-plant analysis.

3. Disease, pest & stress detection (ML performance)

- Early-stage disease detection: 88% accuracy (CNN-based).
- Nutrient deficiency detection: 85% accuracy.
- Pest detection (common pests e.g., aphids, caterpillars): 82% accuracy.
- Crop stress identification: Correlation with expert assessment ≈ 90%.

4. Cloud processing, latency & alerts

- Average end-to-action processing time: ~15 minutes from collection to actionable insight.
- Alert delivery latency: Critical notifications delivered within ~5 minutes of detection.
- Cloud pipeline scaled for concurrent multi-drone operations; data encrypted during transmission (MQTT over cellular).

5. Operational impact & economics

- Yield improvement: Average +12% reported for participating farms.
- Disease-related loss reduction: ~60%.
- Return on investment: Estimated within 18 months (initial system cost ≈ $15,000).
- Labour reduction: ~40% compared to manual monitoring.

6. Limitations observed

- Weather dependency: flights restricted in high winds (>25 mph) or heavy rain.
- Battery life: multiple sorties required for very large areas; battery-swap workflow needed.
- Model generalization: ML accuracy varied across some crop varieties — additional labeled data required.
- Cellular connectivity: remote areas sometimes experienced delayed real-time transmission.

7. Farmer feedback & usability

- Mobile/web UI rated “easy to use” by 92% of farmers in trials.
- 87% of users reported changing farming practices based on system recommendations; color-coded field maps were highlighted as particularly useful.