Practical 1

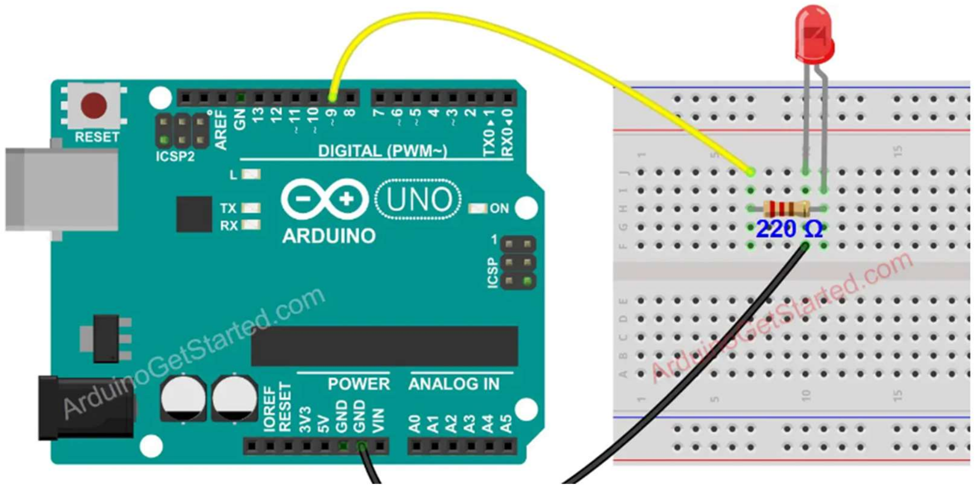
- **Study of Arduino board and Interfacing of LED (s) with Arduino.**

  An Arduino board is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards can be used to develop interactive electronic objects, taking inputs from a variety of sensors and controlling outputs such as motors, lights, and other actuators.

  The Arduino board is powered by a microcontroller, which is a small computer on a single integrated circuit. The microcontroller is programmed using the Arduino IDE, a software application that provides a simplified coding environment.

  Arduino boards have a variety of digital and analog input and output pins, which can be used to connect to sensors, actuators, and other devices. Arduino boards also have a number of built-in features, such as a USB port for communication with a computer and an LED for indicating the board's status.

  Interfacing of LED(s) with Arduino



  An LED (light-emitting diode) is a semiconductor device that emits light when an electric current flows through it. LEDs are commonly used in electronic devices as indicators, displays, and light sources.

  To interface an LED to an Arduino board, simply connect the shorter leg of the LED to the ground (GND) pin on the Arduino board and the longer leg of the LED to a digital output pin on the Arduino board. A resistor should be connected in series with the LED to limit the current flowing through it.

Once the LED is connected to the Arduino board, you can use the Arduino IDE to write a program to control the LED. For example, you could write a program to turn the LED on and off at regular intervals.

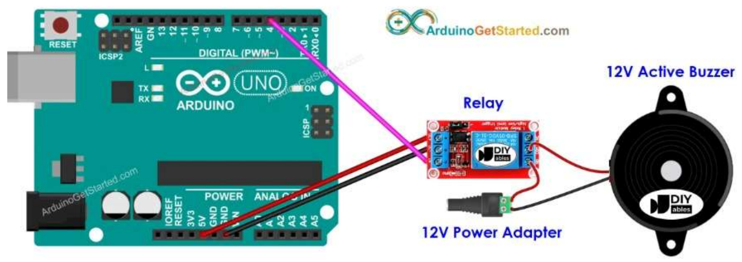
Code:

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Practical 2

- **Study and implementation of Buzzer, Switches, LCD, keypad, LDR, Ultrasonic sensors and PWM interfacing with Arduino.**

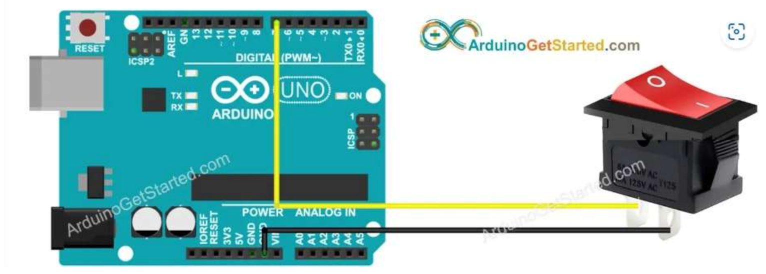
1. **Buzzer:**
   A buzzer is a simple audio output device that can produce sound. You can interface a buzzer with an Arduino by connecting one terminal to a digital pin and the other to GND.



Code:

```
int buzzerPin = 8;

void setup() {
  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  digitalWrite(buzzerPin, HIGH);
  delay(1000);
  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```

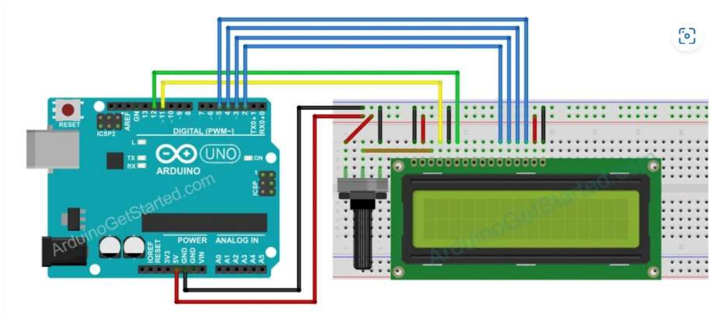
2. **Switches:**
   Switches are used for input. Connect one terminal to a digital pin and the other to GND.



Code:

```
const int switchPin = 2;

void setup() {
  pinMode(switchPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  int switchState = digitalRead(switchPin);

  if (switchState == LOW) {
    Serial.println("Switch is pressed");
  } else {
    Serial.println("Switch is released");
  }

  delay(100);
}
```

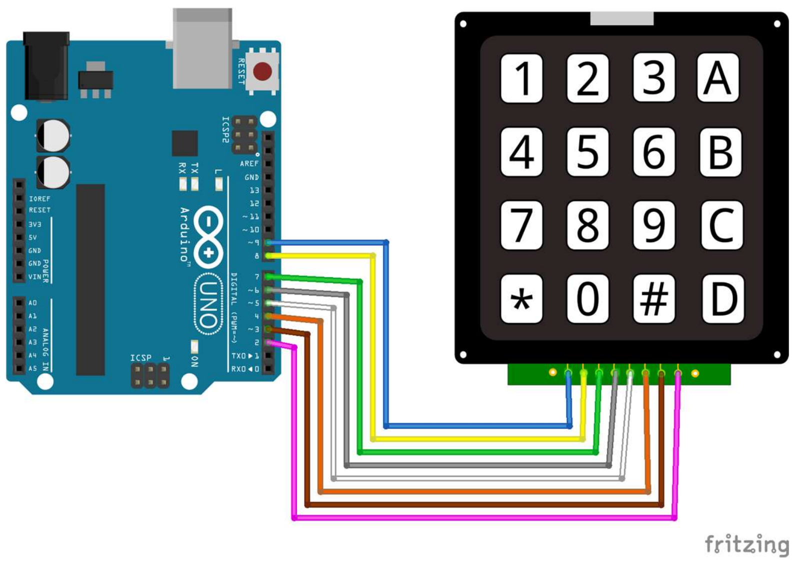
3. **LCD:**
   Interfacing an LCD requires a LiquidCrystal library.



Code:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Hello, Arduino!");
}

void loop() {
  lcd.setCursor(0, 1);
  lcd.print("Time: ");
  lcd.print(millis() / 1000);
  delay(1000);
  lcd.clear();
  delay(1000);
}
```

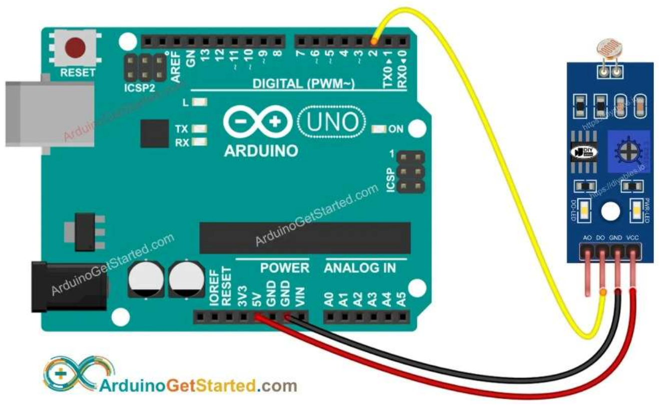
4. **Keypad:**
   To interface a keypad, you typically use a library like "Keypad.h." This allows you to detect key presses and use them in your projects for input.



Code:

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS,
COLS);

void setup() {
  Serial.begin(9600);
}

void loop() {
  char key = keypad.getKey();

  if (key) {
    Serial.println(key);
  }
}
```

5. **LDR:**
   An LDR changes its resistance based on light intensity. Connect one end to an analog pin and the other to GND.

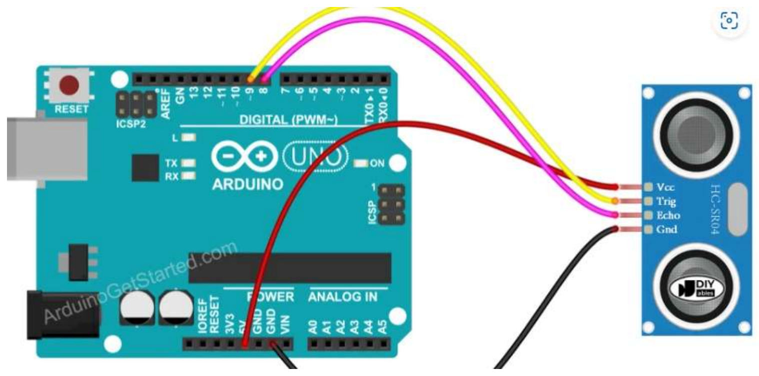


Code:

```
const int ldrPin = A0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  int ldrValue = analogRead(ldrPin);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);
  delay(1000);
}
```

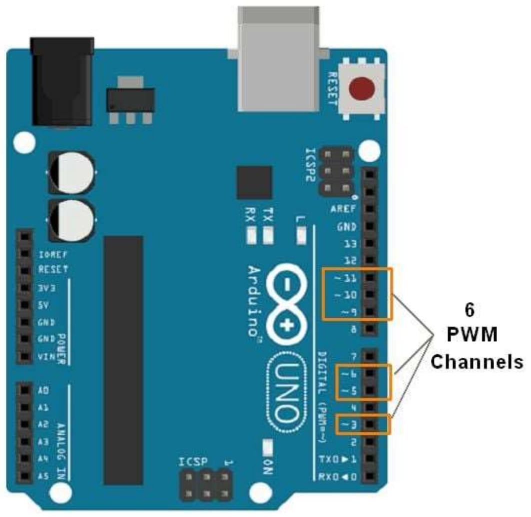
6. **Ultrasonic Sensors:**
   Ultrasonic sensors, like the HC-SR04, can measure distance by sending and receiving ultrasonic waves. Connect the trigger and echo pins to digital

pins on the Arduino. Use libraries to calculate distance based on the time it takes for the signal to bounce back.



Code:
```
const int trigPin = 9;
const int echoPin = 10;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  int distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(1000);
}
```

7. **PWM:**
PWM is used for controlling the intensity of various devices like LEDs, motors, and servos. Arduino has several PWM pins. You can use analogWrite**()** to control the PWM output.



Code:

```
int ledPin = 9;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  for (int brightness = 0; brightness <= 255; brightness++) {
    analogWrite(ledPin, brightness);
    delay(10);
  }

  for (int brightness = 255; brightness >= 0; brightness--) {
    analogWrite(ledPin, brightness);
    delay(10);
  }
}
```

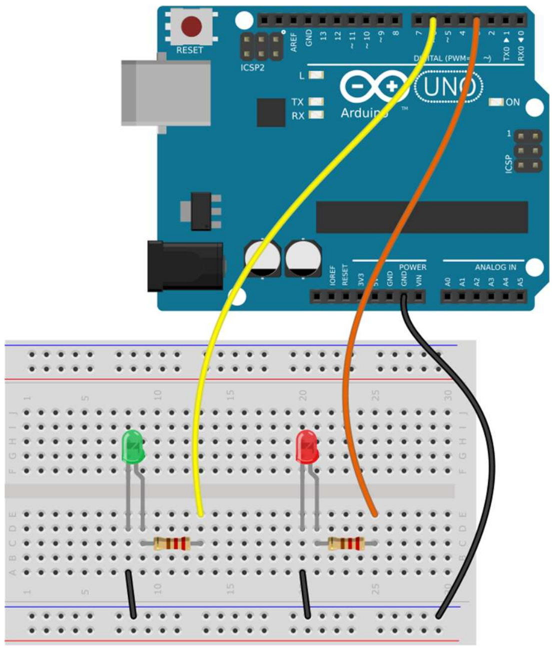
Practical 3

- **Study of serial communication and device control using serial communication with Arduino.**

Serial communication is a form of data transmission in which the data bits are sent one by one, in a sequential order. It is one of the most common and widely used communication protocols, and is used in a wide variety of applications, including computer networks, industrial control systems, and consumer electronics.

Arduino boards have built-in support for serial communication, which allows them to communicate with other devices, such as computers, sensors, and actuators. This makes serial communication a very powerful tool for controlling devices with Arduino.

To use serial communication with Arduino, you first need to initialize the serial port. This is done using the Serial.begin() function, which takes a baud rate as an argument. The baud rate is the speed at which data is transmitted over the serial port, and it is important to match the baud rate of the Arduino to the baud rate of the device you are communicating with.



Once the serial port is initialized, you can start sending and receiving data using the Serial.write() and Serial.read() functions. The Serial.write() function sends a byte of data over the serial port, while the Serial.read() function reads a byte of data from the serial port.

You can use serial communication to control devices in a variety of ways. For example, you can use serial communication to turn a servo motor to a specific position, or to turn on or off a relay. You can also use serial communication to read data from sensors, such as temperature sensors or light sensors.

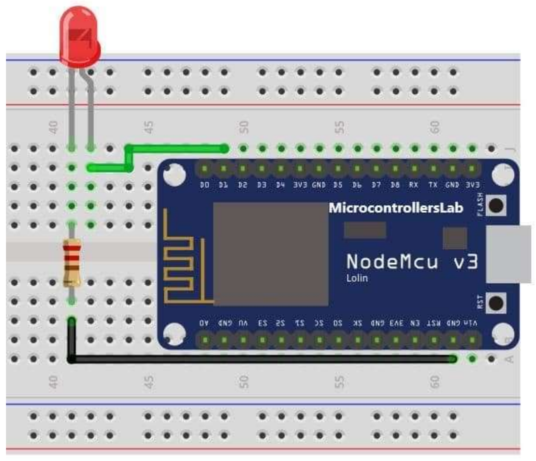
Here is a simple example of a sketch that uses serial communication to control an LED:

Code:
```
void setup() {
  Serial.begin(9600);

  pinMode(13, OUTPUT);
}

void loop() {
  if (Serial.available()) {
    char command = Serial.read();
    if (command == 'H') {
      digitalWrite(13, HIGH);
    }

    else if (command == 'L') {
      digitalWrite(13, LOW);
    }
  }
}
```

Practical 4

- **Study and implementation LED (s) Interfacing with NodeMCU.**

Interfacing LEDs with a NodeMCU (ESP8266) is a common and straightforward project in the world of IoT and microcontrollers. NodeMCU is an ESP8266-based development board that supports Wi-Fi connectivity and is compatible with the Arduino IDE. Below, I'll outline the steps to study and implement LED interfacing with NodeMCU:



1. Gather the Required Components:

- NodeMCU development board
- LEDs (any color)
- Resistors (220-330 ohms)
- Breadboard and jumper wires

2. Set Up Your Development Environment:

- Install the Arduino IDE on your computer if you haven't already.
- Install the necessary board support for the ESP8266 in the Arduino IDE by following the instructions provided in the Arduino ESP8266 core documentation.

3. Connect the Components:

- Place the NodeMCU on the breadboard and make sure it's powered (via USB or an external power supply).
- Connect the LEDs and resistors to the GPIO pins on the NodeMCU. For example, connect an LED to GPIO pin D2. Connect the anode (longer lead)

of the LED to D2 through a current-limiting resistor (220-330 ohms), and the cathode (shorter lead) to the ground (GND) pin on the NodeMCU.

4. Write the Arduino Code:

- Open the Arduino IDE, and create a new sketch.
- Write the code to control the LED(s). Here's a basic example to blink the LED connected to pin D2:

Code:

```
const int ledPin = D2;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

- Make sure to select the appropriate board (NodeMCU 1.0) and COM port in the Arduino IDE.
- Upload the code to the NodeMCU by clicking the "Upload" button.

5. Observe the LED Behavior:

- Once the code is uploaded successfully, you should see the LED on your NodeMCU board blinking on and off with a 1-second interval.

6. Experiment and Expand:

- You can modify the code to implement various LED patterns or control multiple LEDs.
- Experiment with different GPIO pins for LED interfacing.
- Add Wi-Fi capabilities to your NodeMCU to control the LEDs remotely via a web interface or a smartphone app.

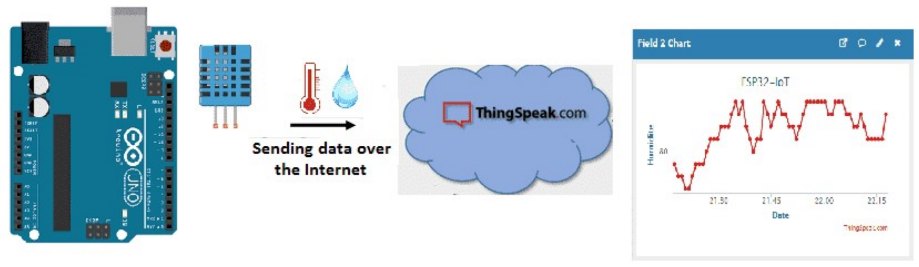
7. Documentation and Further Learning:

- Document your project, including the circuit diagram, code explanations, and any enhancements you make.
- Continue learning about the NodeMCU and explore more IoT projects, such as sensor interfacing, data logging, and remote control applications.

Practical 5

- **Implementation of Publishing data on thingspeak cloud.**

  Publishing data on ThingSpeak, a cloud-based IoT platform, is a great way to store, visualize, and analyze sensor data from your IoT devices. To implement publishing data on ThingSpeak, follow these steps:



1. Sign Up for a ThingSpeak Account:
- If you don't already have a ThingSpeak account, sign up for a free account at ThingSpeak.

2. Create a Channel:
- After logging in to your ThingSpeak account, click on "Channels" in the navigation bar, then click "My Channels" and "New Channel."
- Fill in the required fields, such as Name, Field 1, Field 2, etc., which represent the data fields you want to log.
- Click "Save Channel" to create your channel.

3. Note Your Channel API Key:
- Once your channel is created, go to the "API Keys" tab within your channel.
- Note down the "Write API Key." You'll need this key to send data to your channel.

4. Set Up Your Hardware:
- Connect your IoT device (e.g., Arduino with Wi-Fi capabilities, Raspberry Pi, ESP8266, etc.) and sensors as needed.
- Ensure your device has internet connectivity and can send HTTP requests.

5. Write Code to Publish Data:
- Depending on your hardware, you'll need to write code to read sensor data and send it to ThingSpeak. Here's an example using an Arduino with Wi-Fi (ESP8266) and the ThingSpeak Arduino library:

Code:
```
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";

const char* server = "api.thingspeak.com";
const String apiKey = "YourAPIKey";

void setup() {
  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() {
  float temperature = 25.5;
  float humidity = 60.0;

  ThingSpeak.setField(1, temperature);
  ThingSpeak.setField(2, humidity);

  int response = ThingSpeak.writeFields(channelNumber, apiKey);

  if (response == 200) {
    Serial.println("Data sent successfully");
  } else {
    Serial.println("Error sending data");
  }

  delay(15000);
}
```

- Replace "YourWiFiSSID", "YourWiFiPassword", "YourAPIKey", and channelNumber with your actual Wi-Fi credentials and ThingSpeak API key.

6. Upload and Run Your Code:
- Upload the code to your IoT device.

- The device will connect to Wi-Fi and start sending data to your ThingSpeak channel at regular intervals.

7. View and Analyze Data:

- Go to your ThingSpeak channel's webpage to view the data being published in real-time.
- You can use ThingSpeak's built-in visualization tools to create charts and graphs for your data.

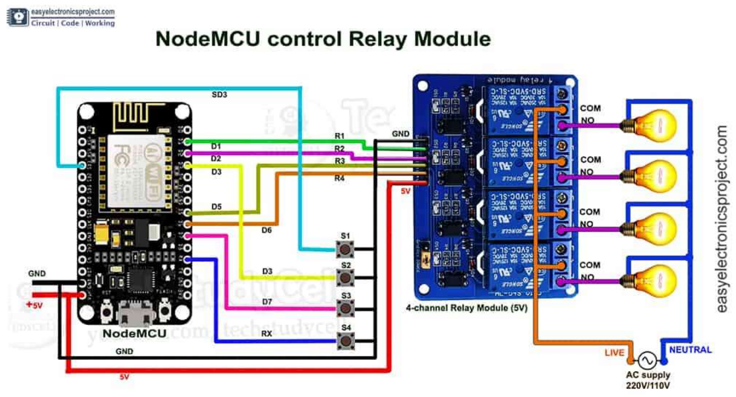
8. Automate and Enhance:

- You can further enhance your project by adding more sensors, sending notifications, or creating rules and actions based on your data using ThingSpeak's built-in features.

Practical 6

- **Controlling Appliances using NodeMCU MQTT over the Internet. (Adafruit Cloud)**

  Controlling appliances using a NodeMCU and MQTT (Message Queuing Telemetry Transport) over the Internet with Adafruit IO is a common and powerful IoT application. Adafruit IO is a cloud-based platform that makes it easy to send, receive, and visualize data from IoT devices. Here's a step-by-step guide to implementing this:


easyelectronicsproject.com
Circuit | Code | Working
NodeMCU control Relay Module
SD3
D1
D2
D3
D5
D6
D7
RX
R1
R2
R3
R4
GND
5V
+5V
S1
S2
S3
S4
NodeMCU
4-channel Relay Module (5V)
COM
NO
LIVE
NEUTRAL
AC supply 220V/110V
easyelectronicsproject.com

1. Set Up Adafruit IO Account:

- If you don't already have an Adafruit IO account, sign up at Adafruit IO.

2. Create an Adafruit IO Feed:

- After signing in, click "Feeds" on the left-hand menu.
- Create a new feed (e.g., "appliance-control").

3. Install Required Libraries:

- In the Arduino IDE, install the following libraries if you haven't already:
  - Adafruit MQTT Library
  - Adafruit IO Arduino Library
  - ESP8266WiFi Library (for NodeMCU)

4. Write the Arduino Code:

- Use the Arduino IDE to write code for your NodeMCU. Here's a basic example to control an appliance connected to a relay:

  Code:

```
#include <Adafruit_MQTT.h>
#include <Adafruit_MQTT_Client.h>
```

```
#include <ESP8266WiFi.h>

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";

#define IO_USERNAME "YourAdafruitIOUsername"
#define IO_KEY "YourAdafruitIOKey"
#define IO_FEED "appliance-control"

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, "io.adafruit.com", 1883,
IO_USERNAME, IO_KEY);

Adafruit_MQTT_Publish applianceFeed = Adafruit_MQTT_Publish(&mqtt,
IO_USERNAME "/" IO_FEED);

void setup() {
  Serial.begin(115200);
  delay(10);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  mqtt.connect();
}

void loop() {
  if (mqtt.connected()) {
    applianceFeed.publish("ON");
    delay(5000);
  } else {
    Serial.println("Failed to connect to MQTT");
    delay(5000);
  }

  mqtt.loop();
}
```

- Replace "YourWiFiSSID", "YourWiFiPassword", "YourAdafruitIOUsername", "YourAdafruitIOKey" with your actual credentials.

- Modify the appliance control logic as needed (e.g., turn on/off a relay, control a smart plug, etc.).

5. Upload and Run Your Code:
- Upload the code to your NodeMCU.
- Ensure your NodeMCU is connected to Wi-Fi and publishing data to Adafruit IO.

6. Set Up Adafruit IO Dashboard:
- Log in to Adafruit IO and create a dashboard.
- Add a toggle switch or a button to control the appliance based on the data published by your NodeMCU.

7. Control Your Appliance:
- Use the Adafruit IO dashboard to send commands to your NodeMCU and control the connected appliance over the Internet.

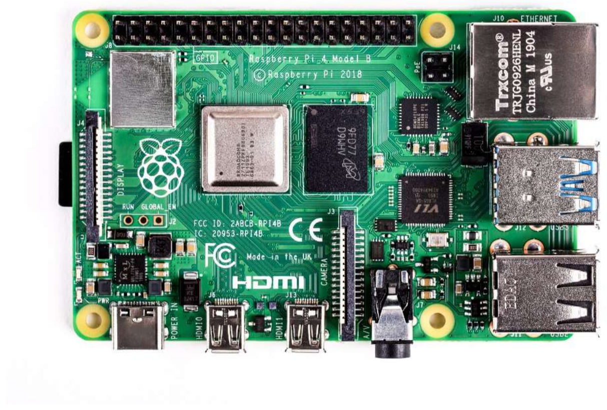
8. Enhance Your Project:
- You can add security features, such as authentication and encryption, to make your IoT project more secure.
- Implement feedback mechanisms to confirm the state of the appliance.
- Consider adding additional sensors or devices for more complex IoT applications.

Practical 7

- **Study and Setup Raspberry Pi board.**

To set up a Raspberry Pi board, you will need the following:

- Raspberry Pi board
- Power supply
- MicroSD card
- USB keyboard and mouse
- HDMI cable
- Monitor



Once you have gathered all of the necessary components, follow these steps to set up your Raspberry Pi board:

- Insert the microSD card into the Raspberry Pi board.
- Connect the power supply to the Raspberry Pi board.
- Connect the USB keyboard and mouse to the Raspberry Pi board.
- Connect the HDMI cable to the Raspberry Pi board and your monitor.
- Turn on the Raspberry Pi board.

When you first turn on your Raspberry Pi board, you will be prompted to select a language, set your country and time zone, and create a user account.

Once you have completed these steps, you will be logged in to your Raspberry Pi board.

The next step is to install an operating system on your Raspberry Pi board. The most popular operating system for Raspberry Pi is Raspbian. To install

Raspbian, you can download the Raspbian image from the Raspberry Pi website and write it to the microSD card using a tool like Etcher.

Once Raspbian is installed on your Raspberry Pi board, you can start using it to do all sorts of things, such as:

- Browse the internet
- Watch videos
- Play games
- Learn to code
- Program robots
- Build home automation projects

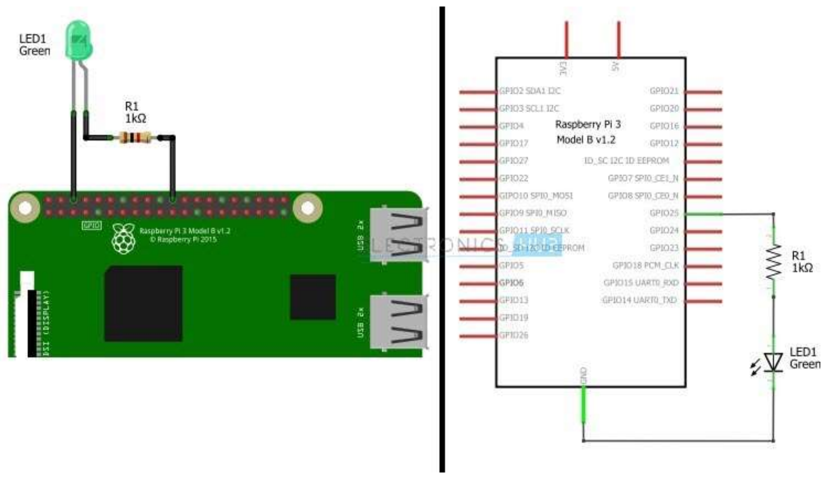
Here are some additional tips for setting up and using a Raspberry Pi board:

- Make sure to use a high-quality power supply. A cheap power supply can cause your Raspberry Pi board to become unstable or even crash.
- Use a heatsink to prevent your Raspberry Pi board from overheating.
- Keep your Raspberry Pi board up to date with the latest software updates.
- Don't be afraid to experiment. The Raspberry Pi board is a great platform for learning and creating.

Practical 8

- **Study and implementation of LED(s) Interfacing with raspberry pi.**

Studying and implementing LED interfacing with a Raspberry Pi is a fundamental step in learning how to control hardware using this popular single-board computer. To get started, you'll need a Raspberry Pi board, some LEDs, resistors, and basic knowledge of Linux and Python programming. Here are the steps to study and implement LED interfacing with a Raspberry Pi:



1. Gather the Required Components:
- Raspberry Pi board (e.g., Raspberry Pi 4, Raspberry Pi 3)
- LEDs (any color)
- Resistors (220-330 ohms)
- Breadboard and jumper wires
- MicroSD card with Raspberry Pi OS installed
- Power supply for the Raspberry Pi
- HDMI monitor, keyboard, and mouse (for initial setup)

2. Set Up Your Raspberry Pi:
- Insert the microSD card with Raspberry Pi OS into the Raspberry Pi.
- Connect the HDMI cable to a monitor or TV, plug in a USB keyboard and mouse, and connect the power supply.
- Follow the on-screen instructions to set up Raspberry Pi OS, including connecting to Wi-Fi (if needed) and updating the system.

3. Wiring the LED:
- Connect the LEDs to the Raspberry Pi GPIO pins using a breadboard and jumper wires. For example, you can connect an LED to GPIO pin 17 as follows:

- Connect the anode (longer lead) of the LED to a GPIO pin (e.g., GPIO 17).
- Connect the cathode (shorter lead) of the LED to a current-limiting resistor (e.g., 220-330 ohms).
- Connect the other end of the resistor to the Raspberry Pi's ground (GND) pin.

4. Write Python Code:

- Create a Python script to control the LED. You can use the **RPi.GPIO** library to interact with the Raspberry Pi's GPIO pins. Here's an example script to turn the LED on and off:

  Code:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

led_pin = 17

GPIO.setup(led_pin, GPIO.OUT)

try:
   while True:
      GPIO.output(led_pin, GPIO.HIGH)
      time.sleep(1)

      GPIO.output(led_pin, GPIO.LOW)
      time.sleep(1)

except KeyboardInterrupt:
   GPIO.cleanup()
```

5. Run Your Python Script:

- Save the Python script on your Raspberry Pi.
- Open a terminal and navigate to the directory where your script is located.
- Run the script using the following command:

```
python your_script.py
```

You should see the LED blinking on and off in a 1-second interval.

6. Experiment and Expand:

- You can modify the Python script to create different patterns or control multiple LEDs.

- Explore more advanced projects, such as using buttons or sensors to trigger LED actions.

7. Documentation and Further Learning:

- Document your project, including the wiring diagram, code explanations, and any challenges you faced.
- Continue learning about Raspberry Pi GPIO, Python programming, and electronics to expand your knowledge and skills.

Practical 9

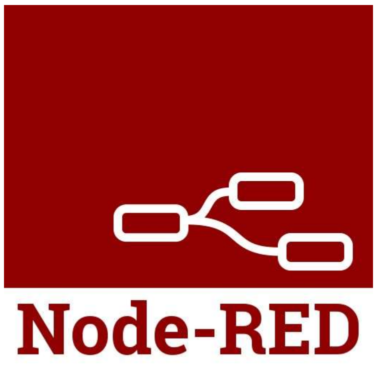
- **Study of Node-red programming tool.**

  Node-RED is a flow-based programming tool for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14, MQTT nodes can make properly configured TLS connections. In 2016, IBM contributed Node-RED as an open source OpenJS Foundation project.

  Node-RED is a popular tool for creating IoT applications because it is easy to use and requires no prior programming experience. It is also very versatile and can be used to create a wide variety of applications, from simple home automation projects to complex industrial control systems.

  Here are some of the benefits of using Node-RED:

  - It is easy to use, even for people with no prior programming experience.
  - It is very versatile and can be used to create a wide variety of applications.
  - It is open source and has a large community of users and developers.
  - It is built on Node.js, which is a powerful and popular JavaScript runtime environment.



  Here are some examples of how Node-RED can be used:

  - Create a home automation system to control your lights, thermostat, and other devices.
  - Build a dashboard to monitor the status of your devices and sensors.

- Create a data acquisition system to collect data from sensors and store it in a database.
- Develop a control system to control a robot or other machine.
- Integrate different systems and services, such as social media, email, and weather services.

If you are interested in learning more about Node-RED, there are many resources available online and in the Node-RED community. There are also many tutorials and examples that can help you get started.

Here are some tips for getting started with Node-RED:

- Start by installing Node-RED on your computer.
- Open the Node-RED editor in your web browser.
- Drag and drop nodes from the palette onto the workspace to create your flow.
- Connect the nodes together to create a sequence of operations.
- Deploy your flow to the Node-RED runtime.
- Test your flow to make sure it is working as expected.

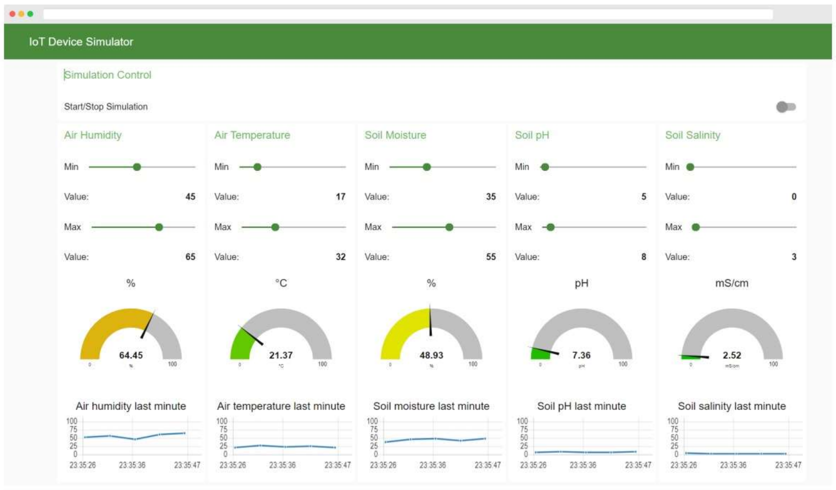
Once you have created a basic flow, you can start to add more complex features, such as using functions and variables. You can also use Node-RED to interact with hardware devices and APIs.

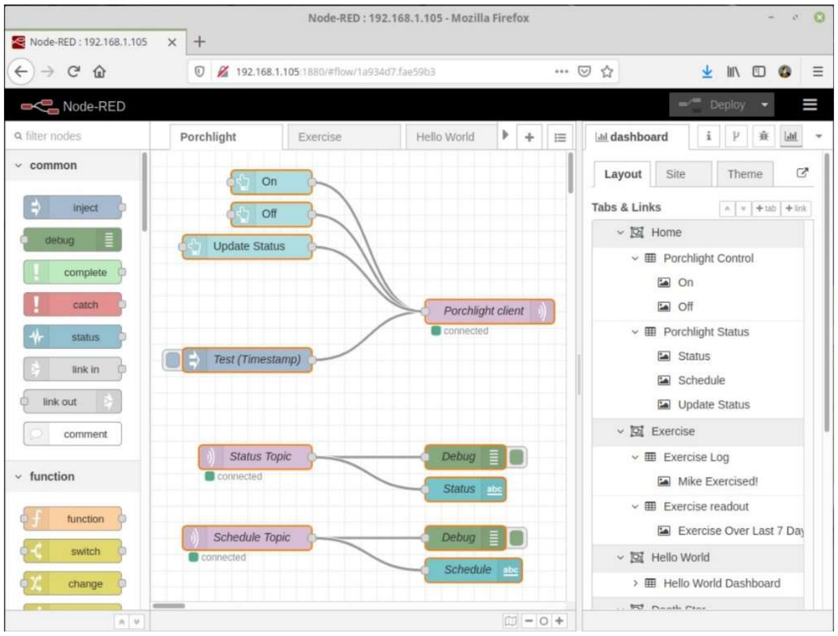
Node-RED is a powerful tool for creating IoT applications. It is easy to use and has a large community of users and developers. If you are interested in learning more about Node-RED, there are many resources available online and in the Node-RED community.

Practical 10

- **Study and implementation of processing data from different sensors and visualize data on Node-red dashboard.**

  Processing data from different sensors and visualizing that data on a Node-RED dashboard is a common use case for IoT and automation projects. In this guide, I'll walk you through the process of setting up Node-RED, connecting various sensors, and creating a dashboard to visualize the sensor data.





1. Set Up Node-RED:

If you haven't already installed Node-RED, follow the installation steps provided in the previous response.

2. Connect Sensors:

Connect your sensors to your hardware platform (e.g., Raspberry Pi) as needed. The specific connections depend on the type of sensors you're using (e.g., GPIO pins for digital sensors, ADC for analog sensors).

3. Install Node-RED Nodes:

Node-RED has a rich ecosystem of nodes created by the community. You may need to install specific nodes to interface with your sensors. To do this:

- Open the Node-RED user interface by navigating to **http://localhost:1880** in your web browser.
- Click on the hamburger menu in the top-right corner and select "Manage palette."
- In the "Install" tab, you can search for and install nodes relevant to your sensors (e.g., node-red-contrib-dht-sensor for DHT temperature and humidity sensors).

4. Create Sensor Flows:

- Drag and drop the sensor nodes onto the Node-RED canvas.
- Configure the nodes by double-clicking on them and specifying the sensor type and pin numbers (if applicable).
- Connect the sensor nodes to appropriate input nodes (e.g., MQTT input nodes) if you want to publish the sensor data to a topic.

5. Process and Publish Data:

- Use function nodes or other processing nodes to manipulate the sensor data as needed.
- If you want to send the data to a Node-RED dashboard, you can use MQTT output nodes to publish data to MQTT topics.

6. Set Up Node-RED Dashboard:

- Install the Node-RED Dashboard node by going to the "Manage palette" menu.
- Add a new dashboard tab by dragging the "ui_tab" node onto the canvas and configuring it.
- Add a dashboard group and a dashboard template node to your flow. Configure the template node to display the sensor data as you like.

7. Visualize Data:

- Use the template node's HTML and AngularJS features to customize how the sensor data is displayed on your Node-RED dashboard.

- Deploy your flow.

8. Access the Dashboard:

- Open a web browser and navigate to the dashboard URL, typically **http://localhost:1880/ui**. Replace localhost with your Raspberry Pi's IP address if you're accessing it remotely.

9. Observe and Analyze Data:

- You should now see your sensor data displayed in real-time on the Node-RED dashboard.
- Customize the dashboard by adding charts, gauges, buttons, and other UI elements to visualize the data in different ways.

10. Additional Features:

- You can add more sensors and create additional flows to process and visualize their data on the dashboard.
- Implement dashboard controls to interact with devices or set up alerts based on sensor data.

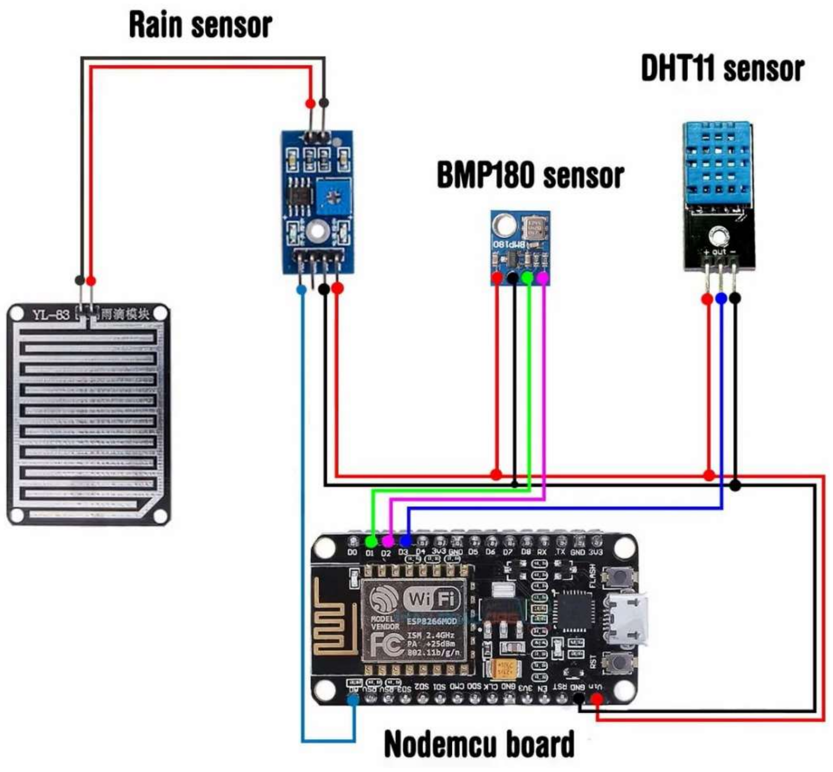
11. Security Considerations:

- When using Node-RED in a production environment, make sure to secure your Node-RED instance, enable authentication, and implement HTTPS for secure communications.

Practical 11

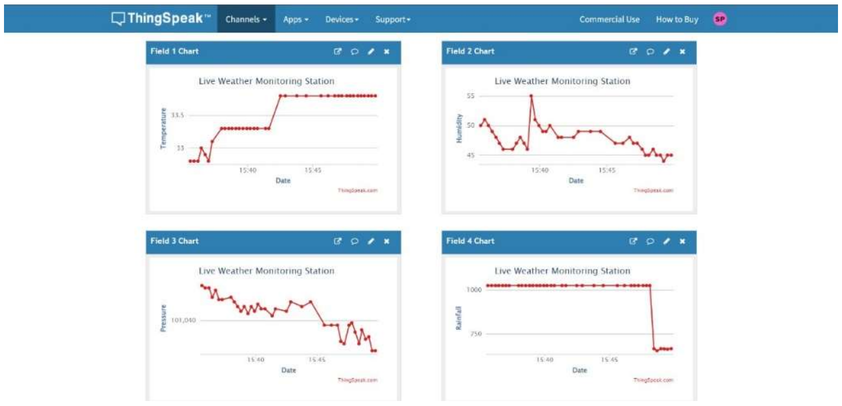
- **Write a sketch that will upload data (like temperature, Light status, etc) on thingspeak cloud.**

  To upload data to the ThingSpeak cloud using an Arduino, you'll need an Arduino board with an internet connection capability (e.g., ESP8266 or ESP32) and the ThingSpeak library installed in your Arduino IDE. Below is a sample Arduino sketch that uploads temperature and light status data to ThingSpeak. In this example, I'm assuming you have a DHT11 or DHT22 temperature and humidity sensor and a simple light sensor (e.g., LDR) connected to your Arduino. Ensure you have the necessary libraries installed for your sensors.



Code:

```
#include <WiFi.h>
#include <ThingSpeak.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>

const char* ssid = "YourWiFiSSID";
const char* password = "YourWiFiPassword";

const char* thingSpeakApiKey = "YourThingSpeakAPIKey";
const char* thingSpeakChannelID = "YourThingSpeakChannelID";

#define DHTPIN 2
#define DHTTYPE DHT11 // DHT11 or DHT22, change accordingly
DHT dht(DHTPIN, DHTTYPE);

const int lightSensorPin = A0;

void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  dht.begin();

  ThingSpeak.begin(client);
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  int lightValue = analogRead(lightSensorPin);
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
  } else {
    Serial.print("Temperature: ");
    Serial.println(temperature);
    Serial.print("Humidity: ");
    Serial.println(humidity);
    Serial.print("Light Value: ");
    Serial.println(lightValue);

    ThingSpeak.setField(1, temperature);
    ThingSpeak.setField(2, humidity);
    ThingSpeak.setField(3, lightValue);

    int response = ThingSpeak.writeFields(thingSpeakChannelID,
thingSpeakApiKey);

    if (response == 200) {
      Serial.println("Data sent to ThingSpeak successfully.");
    } else {
      Serial.println("Error sending data to ThingSpeak.");
    }
  }

  delay(15000);
}
```



Make sure to replace the placeholders in the code with your actual Wi-Fi credentials, ThingSpeak API key, and ThingSpeak channel ID.

Here's a brief overview of what the sketch does:

1. It connects to your Wi-Fi network.
2. It initializes the DHT sensor and reads temperature and humidity values.
3. It reads the light sensor value.
4. It uploads temperature, humidity, and light sensor values to your ThingSpeak channel.
5. The loop repeats every 15 seconds, uploading new data to ThingSpeak.

Practical 12

- **Case study on IoT Applications like**

  ● Home Automation: This home automation system based on IoT automates the functioning of household appliances over the Internet.

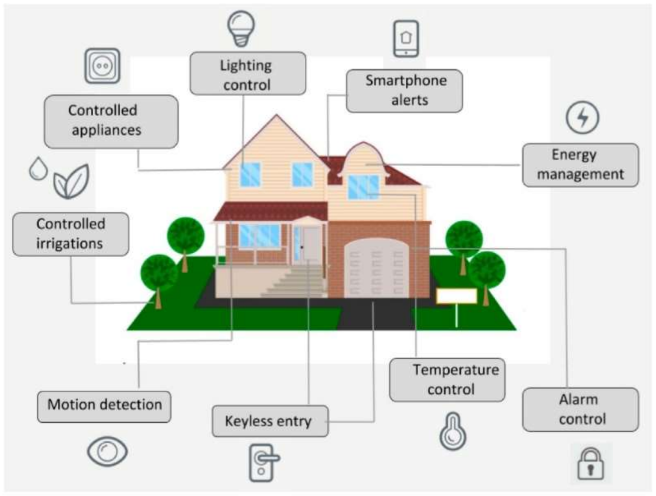
  ● Smart Agriculture System: This IoT-based system performs the routine agricultural tasks automatically and allows farmers to focus on more labor-intensive tasks.

  ● Smart Street light monitoring system: The of the major challenges related to street lights is that they are left on even during daylight hours or when there's no one on the street. An IoT-powered street light monitoring system can help us handle this challenge. Besides, the system will also ensure consumption monitoring, low power consumption, and instant faulty light detection.

  Implementation of mini projects on smart irrigation System, Smart dustbin, Intelligent Building, Smart harvesting, Smart Hospital, Smart classroom, Smart transportation/traffic, Smart Museum etc.

1. Home Automation:

Overview: Home automation systems based on IoT enable homeowners to control and manage their household appliances remotely, leading to convenience, energy efficiency, and increased security.



Key Features:

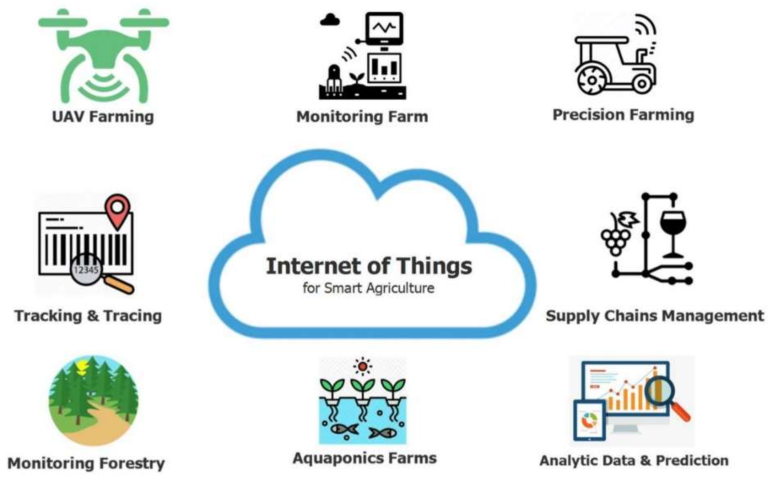
- **Remote Control:** Homeowners can use smartphones or web applications to control lights, thermostats, locks, and more, even when away from home.
- **Energy Efficiency:** IoT sensors can detect occupancy and adjust lighting and HVAC systems accordingly, reducing energy consumption.
- **Security:** IoT-based security cameras and sensors can provide real-time surveillance and send alerts in case of unusual activities.
- **Voice Control:** Integration with voice assistants like Amazon Alexa or Google Assistant allows users to control devices using voice commands.

Benefits:

- Energy savings through smart lighting and HVAC control.
- Improved security with real-time monitoring.
- Convenience and comfort through remote device control.
- Enhanced property value and appeal.

2. Smart Agriculture System:

Overview: Smart agriculture systems powered by IoT technology help farmers optimize their operations, increase crop yields, and reduce resource wastage.



Key Features:

- **Soil Monitoring:** IoT sensors measure soil moisture, temperature, and nutrient levels, enabling precise irrigation and fertilization.

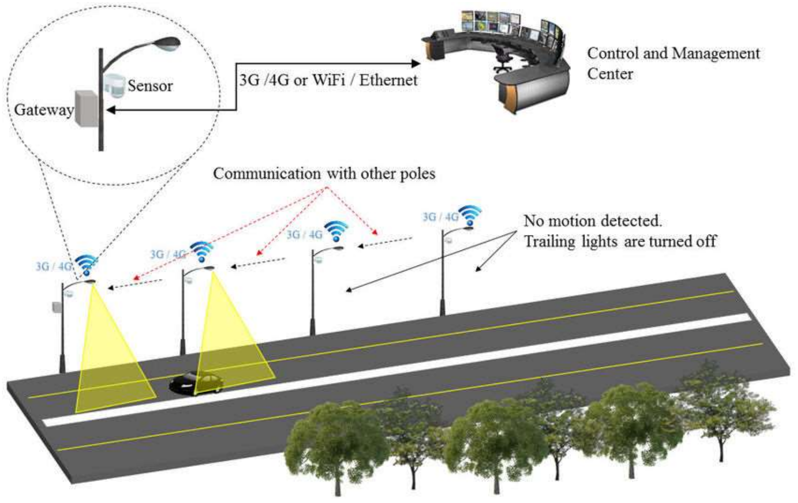
- **Weather Forecasting:** Integration with weather data provides insights for better crop management and protection.
- **Crop Health Monitoring:** Drones and IoT devices can assess crop health, detect pests, and identify disease outbreaks.
- **Automated Machinery:** IoT-connected machinery and tractors can perform tasks like plowing, seeding, and harvesting autonomously.

Benefits:

- Increased crop yield and quality.
- Reduced resource usage (water, fertilizers) and costs.
- Timely decision-making through real-time data.
- Sustainable farming practices.

3. Smart Street Light Monitoring System:

Overview: Smart street light monitoring systems using IoT technology enhance the efficiency of public lighting while reducing energy consumption and maintenance costs.


Sensor
Gateway
3G /4G or WiFi / Ethernet
Control and Management Center
Communication with other poles
3G / 4G
No motion detected.
Trailing lights are turned off

Key Features:

- **Daylight Sensing:** IoT sensors detect ambient light levels and adjust street light brightness accordingly.
- **Fault Detection:** IoT devices monitor the status of each street light and send alerts when a light is faulty.
- **Energy Consumption Monitoring:** Track energy usage and optimize lighting schedules to reduce electricity bills.

- **Remote Control:** Control street lights and schedules via a centralized platform.

Benefits:

- Energy savings by reducing unnecessary lighting during daylight hours.
- Lower maintenance costs with proactive fault detection.
- Enhanced public safety and reduced light pollution.
- Sustainability and reduced environmental impact.