Exercise #1 – Familiarization with Arduino and Perform Necessary Software Installation

Aim:

To study Arduino UNO hardware and Arduino IDE software.

Apparatus Required:

Sign Number	**Name of the Equipment**	**Quantity**
1	Arduino UNO	1
2	Computer with Arduino IDE	1
3	USB Cable	1

Theory:

Internet of Things (IoT):

The Internet of Things - (IoT) describes physical objects (Or groups of such objects) that are embedded with sensors, processing ability, software, and other technologies, and that connect and exchange data with other devices and systems over the Internet or other communications networks.

The field has evolved due to the convergence of multiple technologies, including ubiquitous computing, commodity sensors, increasingly powerful embedded systems, and machine learning. Traditional fields of embedded systems, wireless sensor networks, control systems, automation (Including home and building automation), independently and collectively enable the Internet of things. In the consumer market, IoT technology is most synonymous with products pertaining to the concept of the "Smart Home", including devices and appliances (Such as lighting fixtures, thermostats, home security systems and cameras, and other home appliances) that support one or more common ecosystems, and can be controlled via devices associated with that ecosystem, such as smartphones and smart speakers. The IoT can also be used in healthcare systems.

Microcontroller:

Basically, any product or device that interacts with its user has a microcontroller buried inside. Microcontrollers are "Special Purpose Computers", that differs from the PC on:

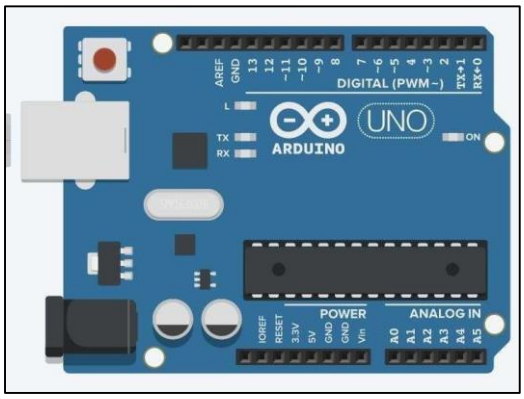
1. Microcontrollers are "Embedded" inside some other device (Often a consumer product) so that they can control the features or actions of the product.

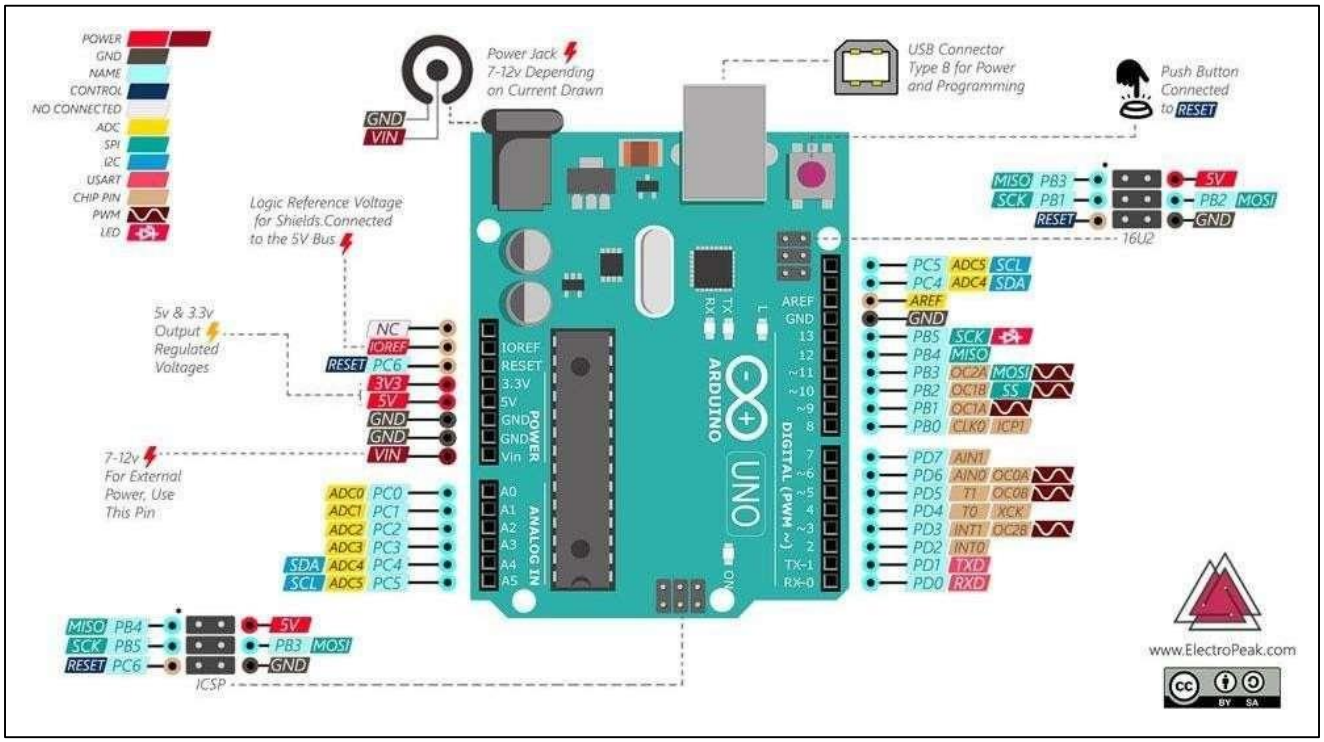
2. Microcontrollers are dedicated to one task and run one specific program. The program is stored in ROM (Read Only Memory) and generally does not change.

3. Microcontrollers are often low-power devices. A desktop computer is almost always plugged into a wall socket and might consume 50 watts of electricity. A battery-operated microcontroller might consume 50 milliwatts.

Arduino UNO:

An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.





General Pin Functions:

LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is ON, when the pin is low, itis OFF.

VIN: The input voltage to the Arduino board when it is using an external power source (As opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7 - 20V). Supplying voltage viathe 5V or 3.3V pins bypasses the regulator, and can damage the board.

3V3: A 3.3V supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

Reset: Typically used to add a reset button to shields that block the one on the board. Special Pin Functions:

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software

control (Using pinMode(), digitalWrite(), and digitalRead() functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (Disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labeled A0 through A5; each provides 10 bits of resolution (That is, 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the analogReference() function.

In addition, some pins have specialized functions:

Serial / UART: Pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connectedto the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

External interrupts: Pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or fallingedge, or a change in value.

PWM (Pulse Width Modulation): Pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite()

function.

SPI (Serial Peripheral Interface): Pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPIcommunication using the SPI library.

I²C (Inter IC Communication): Pin SDA (A4) and pin SCL (A5). Support communication using the Wire library.

AREF (Analog Reference): Reference voltage for the analog inputs.

Arduino IDE Software:

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++.[3] It is used to write and upload programs to Arduinocompatible boards, but also, with the help of third-party cores, other vendor development boards.

Procedure to Install Arduino IDE software:

1. Visit: http://arduino.cc/en/Main/Software

2. Download & install the Arduino environment (IDE) for Windows, Mac, or Linux.

3. Extract the ZIP file. (The extracted folder will contain both the Arduino program itself and also the drivers that allow the Arduino to be connected to your computer by a USB cable).

4. Connect the board to your computer via the UBS cable.

5. The power light on the LED will light up and you may get a “Found New Hardware” message from Windows.

6. Ignore this message and cancel any attempts that Windows makes to try and install drivers automatically for you.

7. Open Device Manager

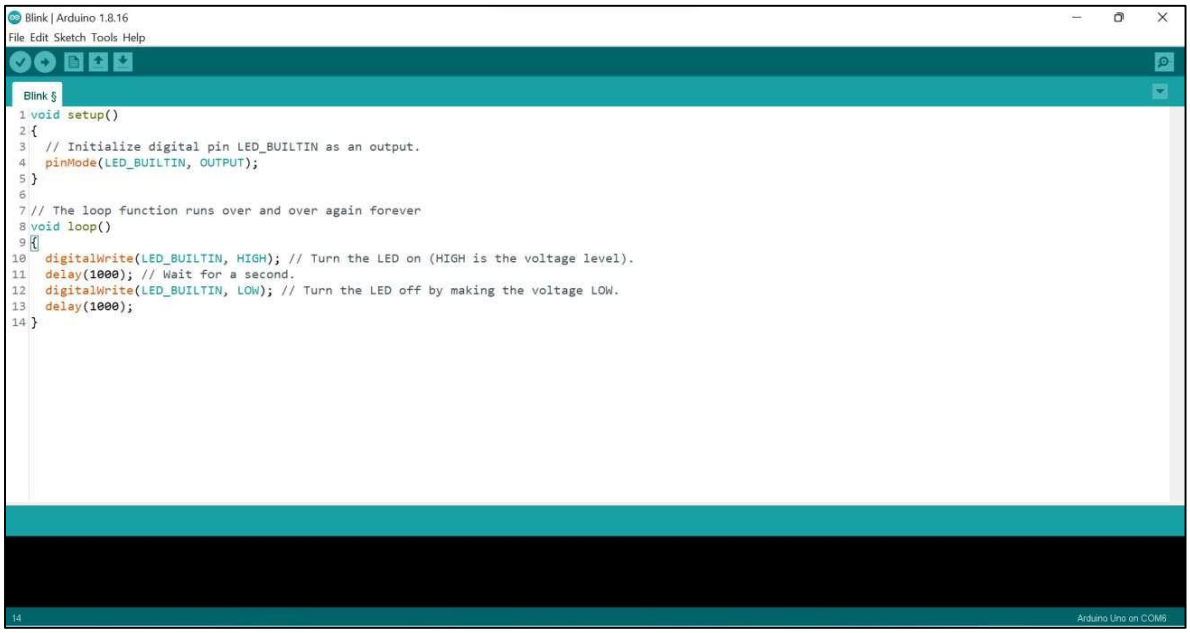
8. Under the section “Other Devices” you should see an icon for “Unknown Device”, write click on it and press update driver software.

9. Select the option: “Search Automatically for Drivers”.

10. You should be done by successfully installing the Arduino driver.

11. Launch the Arduino IDE.

12. Select your board (Tools -> Board ->UNO)



Result:

Thus, the Arduino UNO Hardware and Arduino IDE software were studied.

Exercise # 2a – Interfacing LED With Arduino UNO

Aim:

To interface LED with Arduino UNO.

Apparatus Required:

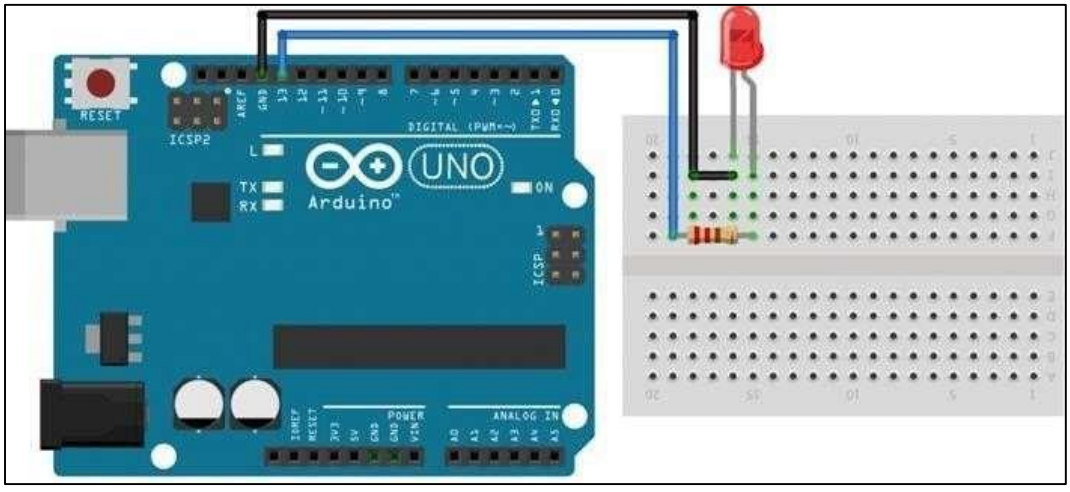
Sign Number	**Name of the Equipment**	**Quantity**
1	Arduino UNO	1
2	Computer with Arduino IDE	1
3	USB Cable	1
4	LED	1
5	330Ω Resistor	1
6	Breadboard	1
7	Jumper Wires	As Required

Theory:

Light Emitting Diode (LED) is a widely used standard source of light in electrical equipment. It has a widerange of applications ranging from your mobile phone to large advertising billboards.

Here, an LED is connected to one of Arduino's digital pins via 330Ω resistor. Whenever the respective pin is set HIGH,current flows via LED and hence it glows.

Circuit Diagram:



Code:

```
#define LED 13
void setup()
{
pinMode(LED, OUTPUT);
}
void loop()
{
digitalWrite(LED, HIGH); delay(1000); digitalWrite(LED, LOW); delay(1000);
}
```

Result:

Hence, a LED is interfaced with Arduino UNO and is made to blink successfully.

Exercise #2b – Interfacing Buzzer with Arduino UNO

Aim:

To interface a buzzer with Arduino UNO and write program to turn ON the buzzer at different frequencies.

Apparatus Required:

Sign Number	**Name of the Equipment**	**Quantity**
1	Arduino UNO	1
2	Computer with Arduino IDE	1
3	USB Cable	1
4	Buzzer	1
5	Breadboard	1
6	Jumper Wires	As Required

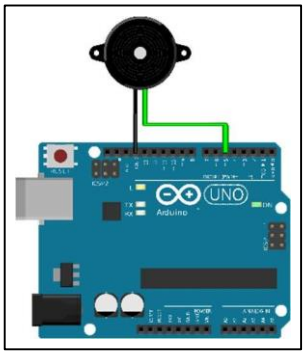
Theory:

A piezo electric buzzer is a simple device which can generate beeps and tones. Working principle of the device is piezoelectric effect. The main component of this device is a piezo crystal, a special material that change shape when a voltage applied to it.

Piezo buzzers are used for making beeps alarms and tones. They can be used in alarm systems, for keypad feedback, or some games. Light weight, simple construction and low price make it usable in various applications like car/truck reversing indicator, computers, call bells etc.

Simply change the frequency of the voltage sent to the piezo and it will start generating sounds by changing shape very quickly.

Circuit Diagram:



Code:

```
#define BUZZER 5

#define FREQUENCY 200

void setup()

{

pinMode(BUZZER, OUTPUT);

}

void loop()

{

tone(BUZZER,FREQUENCY);

delay(500); noTone(BUZZER); delay(500);

}
```

Result:

Hence, buzzer is successfully interfaced with Arduino UNO and is turned ON with different frequencies.

Exercise 3 – Interfacing DHT11 with Arduino and Print Temperature and Humidity

Aim:

To interface DHT11 temperature sensor with Arduino UNO and write a program to print temperature and humidity.

Apparatus Required:

Sign Number	Name of the Equipment	Quantity
1	Arduino UNO	1
2	Computer with Arduino IDE	1
3	USB Cable	1
4	DHT11 Temperature sensor	1
5	Breadboard	1
6	Jumper Wires	As Required

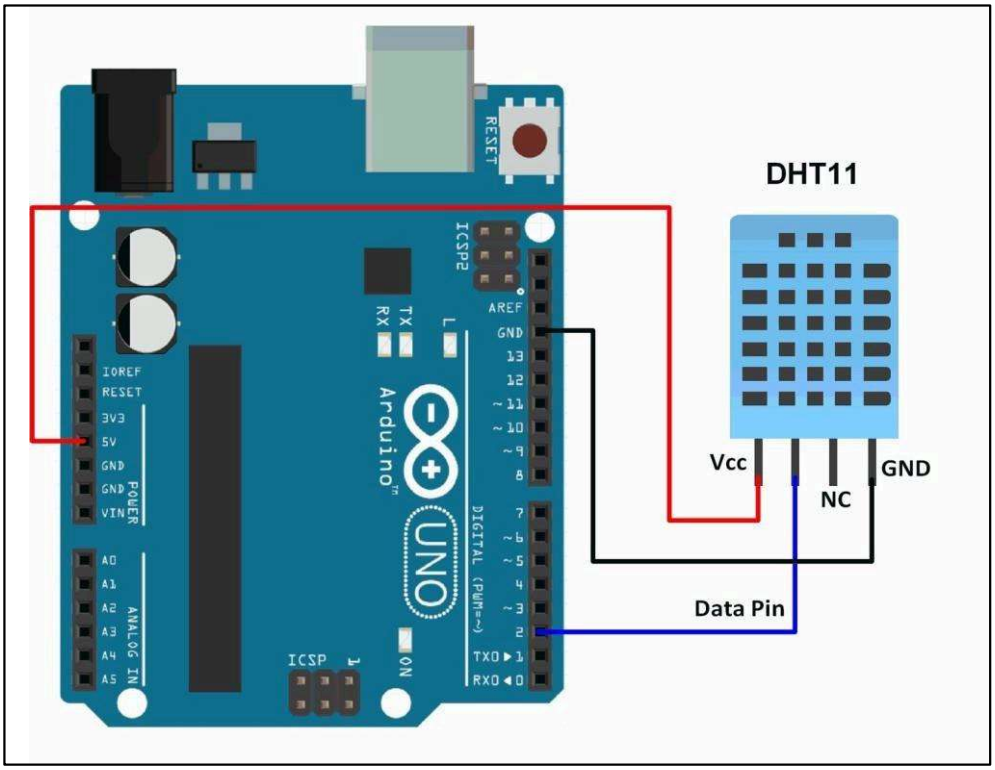
Theory:

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc. to measure humidity and temperature instantaneously. DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED.

DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and for measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz. That is, it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

Circuit Diagram:



Code:

Procedure:

1. Make connections as per the circuit diagram.
2. Open the Arduino IDE in your computer and write the above sketch.
3. Compile the sketch and upload it to Arduino UNO.
4. Once uploaded, open the Serial Monitor.
5. The temperature and humidity will be printed there.

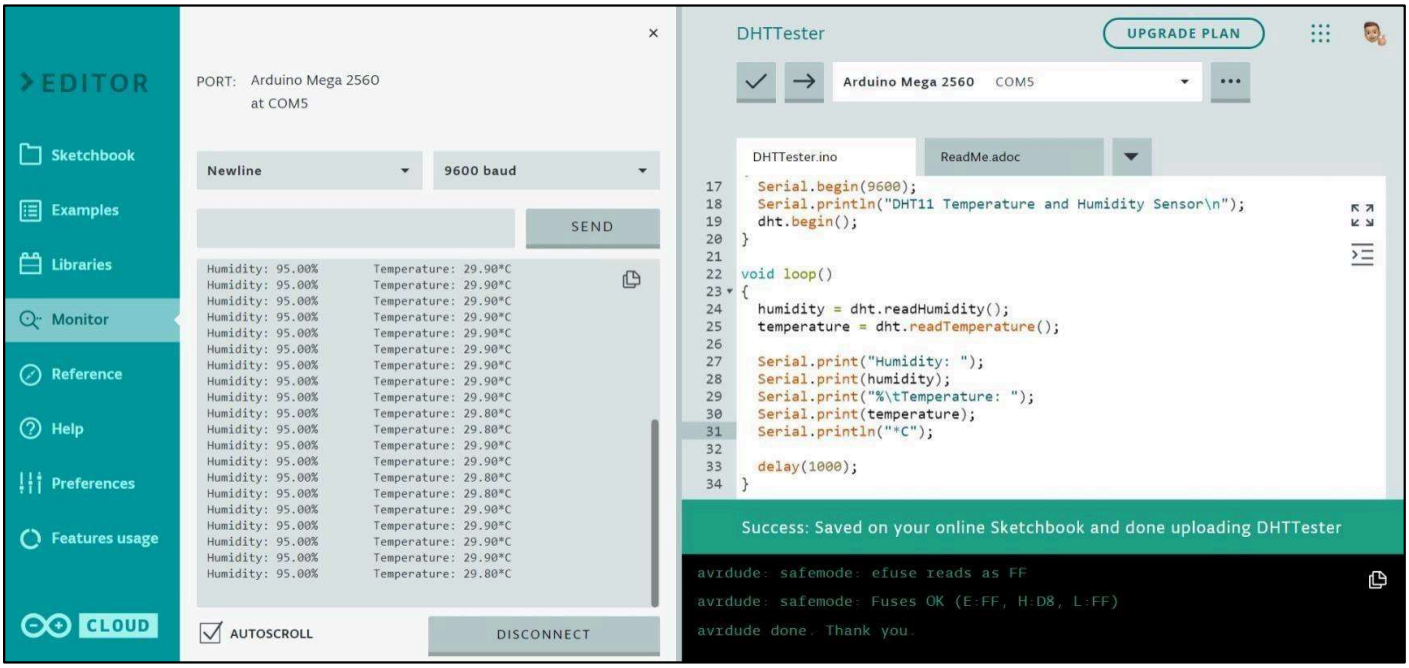
```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 7

#define DHTTYPE    DHT11     // DHT 11

DHT_Unified dht(DHTPIN, DHTTYPE); // define dht class

void setup() {

  // serial monitor setup
 Serial.begin(9600);

 dht.begin(); // sensor starting up

  // sensor takes time to start up so wait
  Serial.println("Sensor Starting up....");
 delay(2000); // 2 seconds wait time



}

void loop() {

 sensors_event_t event;
dht.temperature().getEvent(&event);

if (isnan(event.temperature)) {
Serial.println(F("Error reading temperature!"));
}
else {
Serial.print(F("Temperature: "));
Serial.print(event.temperature);
Serial.println(F("°C"));
}

 // Get humidity event and print its value.
dht.humidity().getEvent(&event);
 if (isnan(event.relative_humidity)) {
```

```
Serial.println(F("Error reading humidity!"));
 }
else {
Serial.print(F("Humidity: "));
Serial.print(event.relative_humidity);
Serial.println(F("%"));
}

delay(1000); // 1 sec delay between every reading
}
```

OUTPUT:

```
Temperature: 30.50°C

Humidity: 55.50%
```

Output:



Result:

Thus, the DHT11 temperature and humidity sensor is interfaced with Arduino UNO and the temperature and humidity data is printed successfully.

To interface Bluetooth/Wifi with Arduino and write a program to send sensor data to smartphone using Bluetooth/Wifi.

Aim

To interface Bluetooth (HC-05) or Wi-Fi (ESP8266) with Arduino and send sensor data to a smartphone.

Theory

Bluetooth and Wi-Fi are wireless communication technologies used for data transmission.Bluetooth (HC-05): Short-range wireless communication module for sending data to smartphones via serial communication.Wi-Fi (ESP8266): A wireless networking module that allows data transmission over a network using TCP/IP.Both modules can be used to transmit real-time sensor data (e.g., temperature, humidity) from Arduino to a smartphone.

Apparatus Required

Component	**Specification**
Arduino Board	Uno/Nano/Mega
Bluetooth Module	HC-05 or HM-10
Wi-Fi Module	ESP8266 (NodeMCU)
Sensor	Any analog sensor (e.g., temperature, LDR)
Jumper Wires	Male-Female, Male-Male
Smartphone	With Bluetooth/Wi-Fi App

Procedure

A. Bluetooth (HC-05)

1. Connect HC-05 module to Arduino as per the circuit diagram.
2. Upload the Bluetooth code to Arduino.
3. Open Bluetooth Terminal App on your smartphone.
4. Pair HC-05 (default password: 1234 or 0000).
5. Select the connected HC-05 in the app.
6. View the sensor data streaming on the app.

B. Wi-Fi (ESP8266)

1. Connect ESP8266 module to Arduino as per the circuit diagram.
2. Upload the Wi-Fi code to Arduino.
3. Open the Serial Monitor and note the IP address.
4. Connect your smartphone to the same Wi-Fi network.

5. Open a web browser and enter the IP address.
6. The webpage displays real-time sensor data.

Bluetooth Code (HC-05)

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(2, 3);  // RX, TX
int sensorPin = A0;
int sensorValue = 0;
void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);
   BTSerial.print("Sensor Data: ");
  BTSerial.println(sensorValue);
  delay(1000);
}
```

Wi-Fi Code (ESP8266)

```
#include <ESP8266WiFi.h>

const char* ssid = "Your_SSID";
const char* password = "Your_PASSWORD";
WiFiServer server(80);
int sensorPin = A0;
int sensorValue = 0;
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
    Serial.println("Connected to WiFi!");
  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection: close");
        client.println();
        client.println("<html><h1>Sensor Data</h1>");
        sensorValue = analogRead(sensorPin);
        client.print("<p>Sensor Value: ");
        client.print(sensorValue);
        client.println("</p></html>");
        delay(1000);
        client.stop();
      }
    }
  }}
```

Output

A. Bluetooth Output

Sensor Value: 345

Sensor Value: 350

Sensor Value: 355

Wi-Fi Output

+--------------------------------------+

| Sensor Data |

+--------------------------------------+

| Sensor Value: 345 |

| Sensor Value: 350 |

| Sensor Value: 355 |

+--------------------------------------+

Result

Thus the code for Bluetooth (HC-05) and Wi-Fi (ESP8266) with Arduino Successfully interfaced

To interface Bluetooth/Wifi with Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth/Wifi.

1. Setting Up Bluetooth/WiFi on Raspberry Pi

Bluetooth Setup

1. Ensure Bluetooth is enabled on your Raspberry Pi:

   ```
   sudo apt update
   sudo apt install bluetooth bluez blueman
   ```

2. Start the Bluetooth service:

   ```
   sudo systemctl start bluetooth
   ```

3. Check Bluetooth devices:

   ```
   hcitool scan
   ```

4. Pair and connect with your smartphone:

   ```
   bluetoothctl
   scan on
   pair <device_address>
   connect <device_address>
   ```

WiFi Setup

1. Check if WiFi is enabled:

   ```
   iwconfig
   ```

2. Connect to a WiFi network:

   ```
   sudo raspi-config
   ```

   - Select **Network Options > Wi-Fi**
   - Enter your SSID and password.

2. Reading Sensor Data from Raspberry Pi

you have a DHT11/DHT22 sensor connected to the Raspberry Pi.

1. Install dependencies:

   ```
   sudo apt install python3-pip
   pip3 install Adafruit_DHT
   ```

2. Read data from DHT11:

```
import Adafruit_DHT
sensor = Adafruit_DHT.DHT11
pin = 4  # GPIO pin
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
if humidity is not None and temperature is not None:
    print(f'Temp: {temperature}°C, Humidity: {humidity}%')
else:
    print('Failed to read sensor data')
```

3. Sending Sensor Data via Bluetooth

Use the bluetooth Python module to send data to a smartphone.

1. Install Bluetooth libraries:

```
sudo apt install python3-bluetooth
```

2. Server Code on Raspberry Pi:

```
import bluetooth
import Adafruit_DHT

sensor = Adafruit_DHT.DHT11
pin = 4

server_sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
server_sock.bind(("", bluetooth.PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]
bluetooth.advertise_service(server_sock, "RaspberryPiServer",
                            service_classes=[bluetooth.SERIAL_PORT_CLASS],
                            profiles=[bluetooth.SERIAL_PORT_PROFILE])

print(f"Waiting for connection on RFCOMM channel {port}...")

client_sock, client_info = server_sock.accept()
print(f"Connected to {client_info}")

try:
    while True:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        if humidity is not None and temperature is not None:
            data = f'Temp: {temperature}°C, Humidity: {humidity}%\n'
            client_sock.send(data)
        else:
            client_sock.send("Failed to read sensor data\n")
```

```
except:
    print("Connection closed")
    client_sock.close()
    server_sock.close()
```

3. **On Smartphone:**
    - Use a Bluetooth terminal app (e.g., **Serial Bluetooth Terminal**) to receive sensor data.

4. Sending Sensor Data via WiFi (Flask Server)

Use Flask to create a REST API that your smartphone can access.

1. Install Flask:

   ```
   pip3 install flask
   ```

2. Flask Server Code on Raspberry Pi:

   ```
   from flask import Flask, jsonify
   import Adafruit_DHT

   app = Flask(__name__)

   sensor = Adafruit_DHT.DHT11
   pin = 4

   @app.route('/sensor', methods=['GET'])
   def get_sensor_data():
       humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
       if humidity is not None and temperature is not None:
           return jsonify({'temperature': temperature, 'humidity': humidity})
       else:
           return jsonify({'error': 'Failed to read sensor data'}), 500

   if __name__ == '__main__':
       app.run(host='0.0.0.0', port=5000, debug=True)
   ```

3. **On Smartphone:**
    - Open a browser and enter http://<RaspberryPi_IP>:5000/sensor to see the sensor data.

Ex5 Mini Projects(any one for each group)

i. Home Automation system with mobile Integration.

ii. Weather Monitoring system using Raspberry Pi/Arduino

iii. Automatic plant watering/irrigation system using Raspberry Pi/Arduino.

iv. Vehicle Tracking System using Raspberry Pi/Arduino.

v. Intrusion detection System using Raspberry Pi/Arduino.

vi. Smart Parking System using Raspberry Pi/Arduino

1. Home Automation System with Mobile Integration

Control home appliances (lights, fans, door locks, etc.) using a mobile app.

- **Components:**
  - Raspberry Pi/Arduino
  - Relay module (to control appliances)
  - WiFi/Bluetooth module (ESP8266 or HC-05)
  - Mobile App (Android/iOS using Flutter or MIT App Inventor)

- **Implementation Steps:**
  1. Connect the appliances to relays.
  2. Use Raspberry Pi/Arduino to receive commands via WiFi/Bluetooth.
  3. Develop a mobile app to control appliances.
  4. Use MQTT or Firebase for cloud integration.

2. Weather Monitoring System using Raspberry Pi/Arduino

Collect real-time weather data (temperature, humidity, air pressure) and display it on a web/app interface.

- **Components:**
  - DHT11/DHT22 (Temperature & Humidity Sensor)
  - BMP180 (Pressure Sensor)
  - Raspberry Pi/Arduino
  - LCD Display (optional)
  - WiFi module (ESP8266)

- **Implementation Steps:**

1. Connect sensors to Raspberry Pi/Arduino.
2. Read sensor data and display it on an LCD or Web UI.
3. Store data on Firebase/ThingSpeak for remote monitoring.
4. Create a dashboard using Flask (for web) or an Android app.

3. Automatic Plant Watering/Irrigation System

Automatically water plants based on soil moisture levels.

- **Components:**

  - Soil Moisture Sensor
  - Relay Module
  - Water Pump
  - Raspberry Pi/Arduino
  - WiFi/Bluetooth module (ESP8266/HC-05)

- **Implementation Steps:**

  1. Connect the soil moisture sensor to Raspberry Pi/Arduino.
  2. If moisture is low, trigger the water pump via a relay.
  3. Send notifications to a mobile app about water status.
  4. Control watering remotely via a web/app interface.

4. Vehicle Tracking System

Track a vehicle's real-time location using GPS and send data to a mobile app.

- **Components:**

  - GPS Module (Neo-6M)
  - GSM Module (SIM800L)
  - Raspberry Pi/Arduino
  - Google Maps API (for visualization)

- **Implementation Steps:**

  1. Connect GPS and GSM modules to Raspberry Pi/Arduino.
  2. Retrieve GPS coordinates and send them via SMS or an online server.
  3. Develop a mobile app or web interface to display the location on a map.
  4. Enable real-time tracking using MQTT or Firebase.

5. Intrusion Detection System

Detect unauthorized entry using motion sensors and alert via SMS/email.

- **Components:**

  - PIR Motion Sensor
  - Camera Module (Optional)
  - Raspberry Pi/Arduino
  - Buzzer/Alarm
  - GSM/WiFi module (ESP8266/SIM800L)

- **Implementation Steps:**

  1. Place PIR motion sensors in key areas.
  2. If motion is detected, trigger an alarm or capture an image.
  3. Send notifications via SMS or email.
  4. Integrate with a mobile app for remote monitoring.

6. Smart Parking System

: Detect parking availability and guide users to free spaces.

- **Components:**

  - Ultrasonic Sensors (for detecting empty spots)
  - Raspberry Pi/Arduino
  - LCD/LED Display
  - WiFi/Bluetooth module
  - Mobile App (for real-time updates)

- **Implementation Steps:**

  1. Place ultrasonic sensors in each parking slot.
  2. Detect if a slot is occupied and update a database.
  3. Display available slots on an LCD or mobile app.
  4. Provide navigation assistance using Google Maps API.