

## **1. SOFTWARE INSTALLATION**

<b>EX.N0 : 1</b>	<b>FAMILIARIZATION WITH ARDUINO/RASPBERRY PI AND PERFORM NECESSARY SOFTWARE INSTALLATION.</b>
<b><u>DATE : 06/02/2025</u></b>	

### **AIM:**

To study Arduino UNO hardware and Arduino IDE software.

### **APPARATUS REQUIRED:**

<b>Sign Number</b>	<b>Name of the Equipment</b>	<b>Quantity</b>
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:**

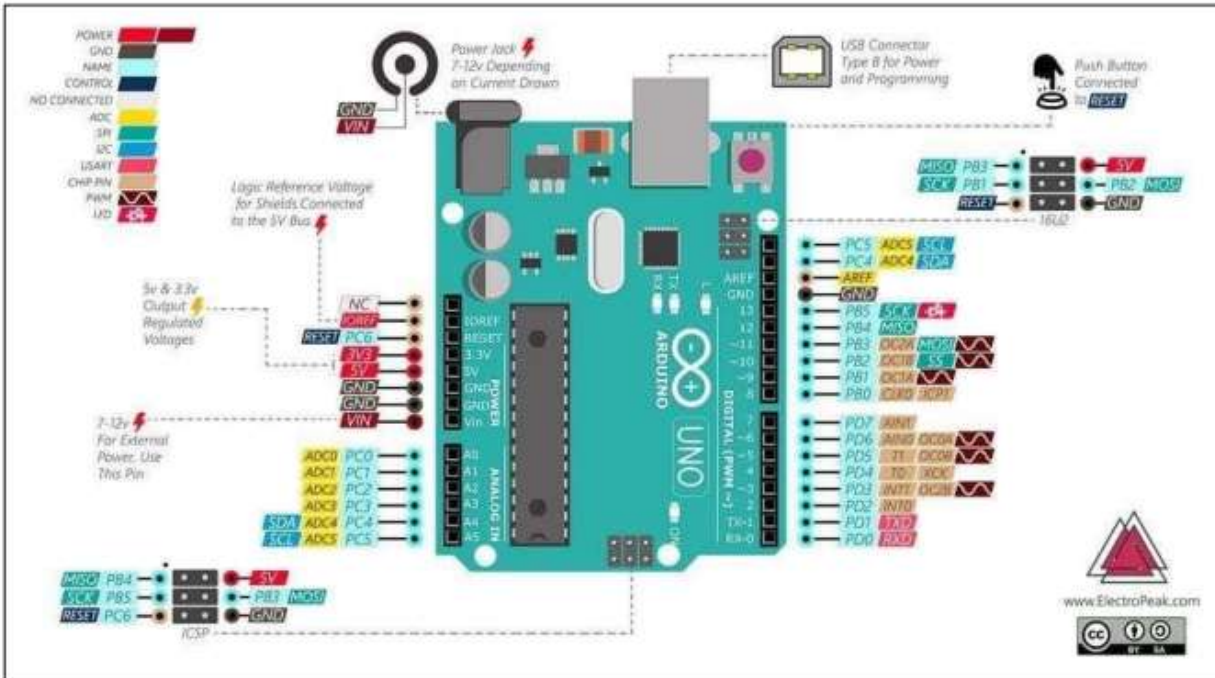
Basically, any product or device that interacts with its user has a microcontroller buried inside. Microcontrollers are "Special Purpose Computers", that differs from the PC on:

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 mill watts.

### **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, it is 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 via the 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, labelled 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 connected to 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 falling edge, 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 SPI communication 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 Arduino compatible 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)

A screenshot of the Arduino IDE (version 1.8.16) window. The interface has a teal header bar with icons for File, Edit, Sketch, Tools, and Help. Below the header is a menu bar. The main area is a code editor with a white background and a teal border. It contains a C++ program for controlling an LED. The code is as follows:

```
1 void setup()
2 {
3   // Initialize digital pin LED_BUILTIN as an output.
4   pinMode(LED_BUILTIN, OUTPUT);
5 }
6
7 // The loop function runs over and over again forever
8 void loop()
9 {
10  digitalWrite(LED_BUILTIN, HIGH); // Turn the LED on (HIGH is the voltage level)
11  delay(1000); // Wait for a second.
12  digitalWrite(LED_BUILTIN, LOW); // Turn the LED off by making the voltage LOW
13  delay(1000);
14 }
```

The bottom of the window shows a black status bar with the text "Arduino IDE 1.8.16" on the left and "Arduino IDE 1.8.16" on the right.

## **RESULT:**

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

## **2. INTERFACING LED**

<b>EX.N0 : 2.a)</b>	<b>INTERFACING LED WITH ARDUINO UNO</b>
<b><u>DATE:12/02/25</u></b>	

### **AIM:**

To interface LED with Arduino UNO.

### **APPARATUS REQUIRED:**

<b>Sign Number</b>	<b>Name of the Equipment</b>	<b>Quantity</b>
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

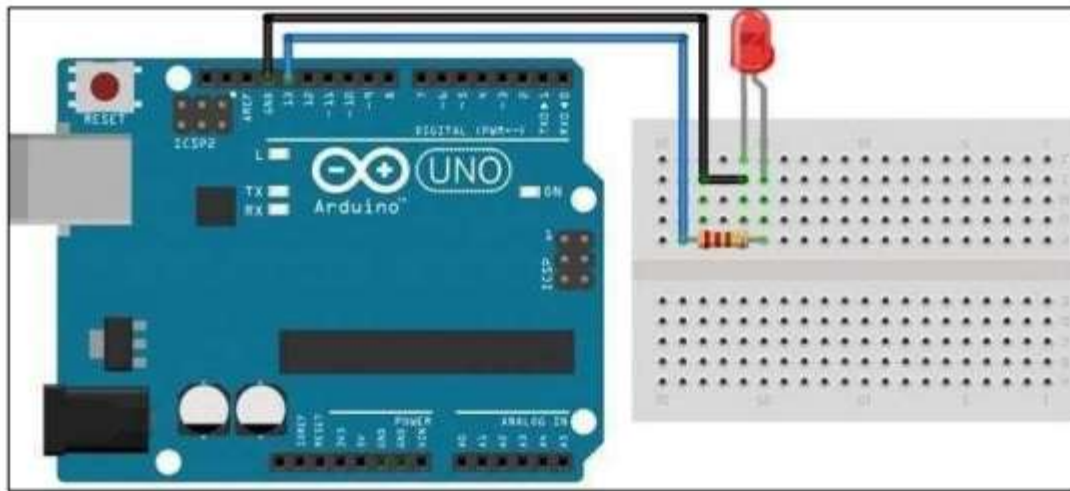
### **THEORY:**

Light Emitting Diode (LED) is a widely used standard source of light in electrical equipment.

It has a wide range 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.



## **2. INTERFACING BUZZER**

<b>EX.N0 : 2.b)</b>	<b>INTERFACING BUZZER WITH ARDUINO UNO</b>
<b><u>DATE : 12/02/2025</u></b>	

### **AIM:**

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

### **APPARATUS REQUIRED:**

<b>Sign Number</b>	<b>Name of the Equipment</b>	<b>Quantity</b>
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

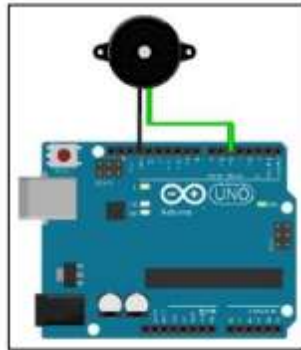
### **THEORY:**

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.

### **3. INTERFACING DHT11**

<b>EX.N0 : 3</b>	<b>INTERFACING DHT11 WITH ARDUINO AND PRINT TEMPERATURE AND HUMIDITY</b>
<b><u>DATE : 24/02/25</u></b>	

#### **AIM:**

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

#### **APPARATUS REQUIRED:**

<b>Sign Number</b>	<b>Name of the Equipment</b>	<b>Quantity</b>
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

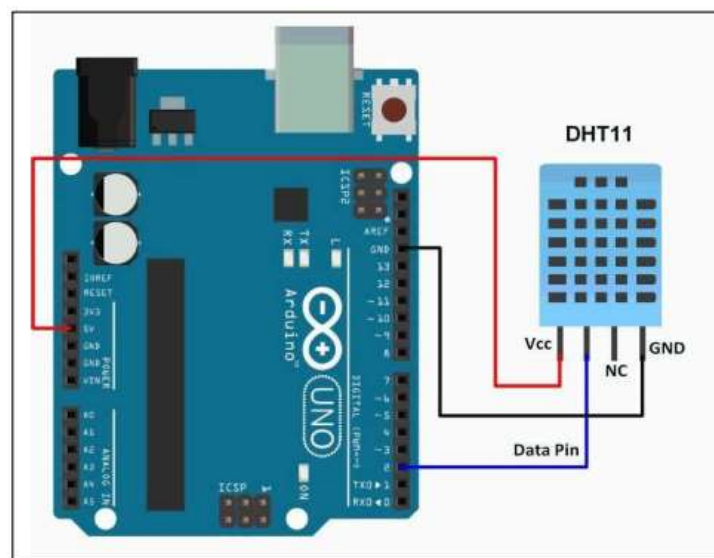
#### **THEORY:**

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:**



### **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.

**CODE:**

```
#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!"));

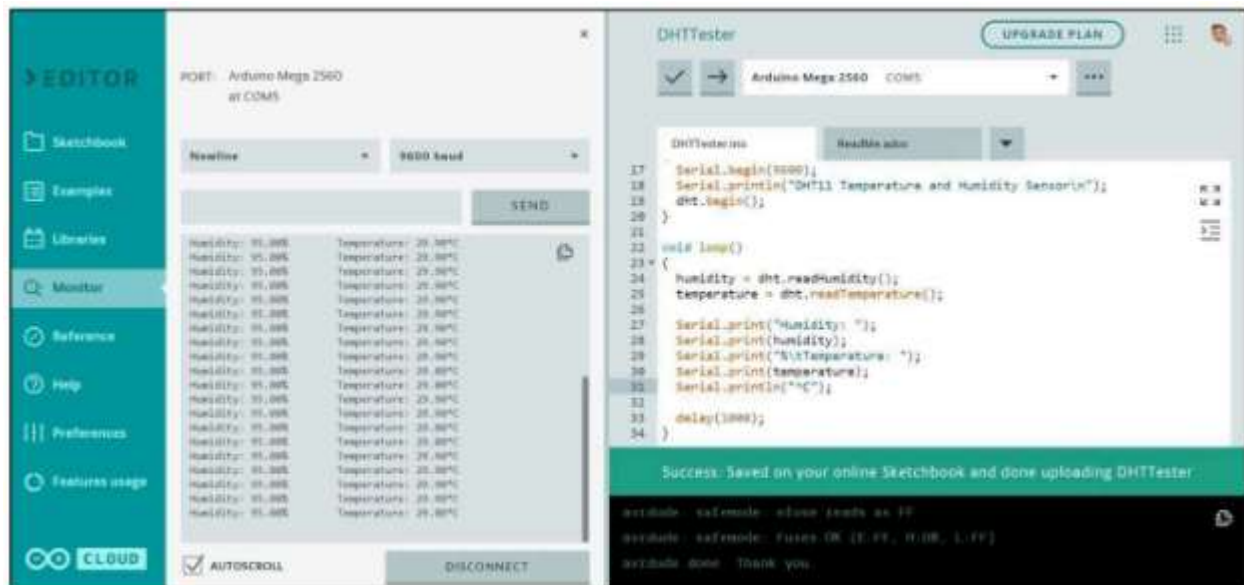
}
```

```
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%



**RESULT:**

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

#### **4. INTERFACING BLUETOOTH/WIFI**

<b>EX.N0 : 4</b>	<b>TO INTERFACE BLUETOOTH/WIFI WITH ARDUINO AND WRITE A PROGRAM TO SEND SENSOR DATA TO SMARTPHONE USING BLUETOOTH/WIFI.</b>
<b><u>DATE : 20/03/25</u></b>	

#### **AIM:**

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

#### **APPARATUS REQUIRED:**

<b>S.no</b>	<b>Name of the Equipment</b>	<b>Specification</b>
1	Arduino Boards	Uno/Nano/Mega
2	Bluetooth Module	HC-05 or HM-10
3	Wi-Fi Module	ESP8266 (NodeMCU)
4	Sensor	Any analog sensor (e.g., temperature, LDR)
5	Jumper Wires	Male-Female, Male-Male
6	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.

#### **4. MINI PROJECTS**

<b>EX.N0 : 5</b>	<b>MINI PROJECTS</b>
<b><u>DATE</u> : 03/04/2025</b>	

- 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.

## **DESCRIPTION**

### **SMART CAR PARKING SYSTEM –**

#### **PROJECT DESCRIPTION**

This project is a simple Smart Car Parking System designed using an Arduino Uno, a servo motor, 2 IR sensors, 2 ultrasonic sensors, and 3 LEDs. The system automates vehicle entry control and slot monitoring in a small parking area with two parking slots.

#### **KEY FEATURES:**

#### **ENTRY GATE CONTROL:**

- An IR sensor before the gate detects an approaching car.
- If at least one parking slot is free, the servo motor opens the gate.
- Once the vehicle crosses the IR sensor after the gate, the gate closes automatically.
- A gate LED blinks while the gate is moving.



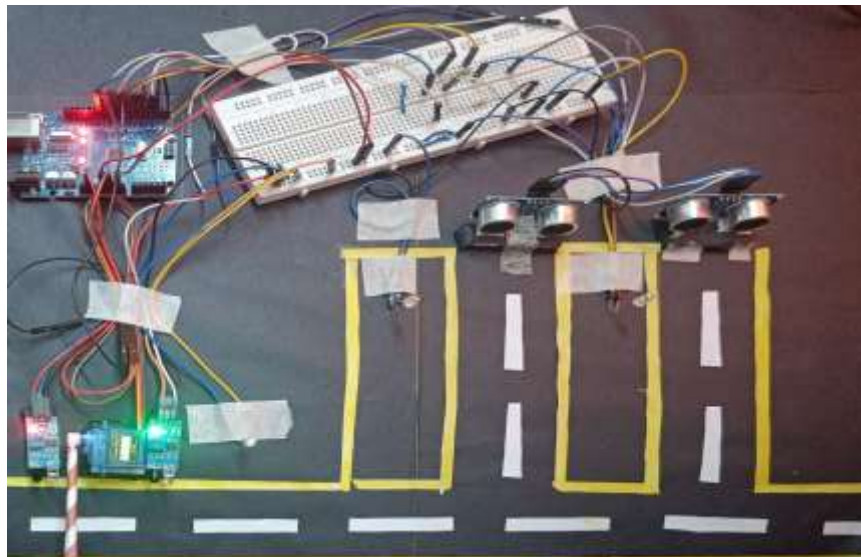
### **SLOT MONITORING:**

- Two ultrasonic sensors constantly measure the distance to detect if each slot is occupied or free.
- LED indicators light up if the slot is occupied.
- If both slots are occupied, new entries are not allowed (gate stays closed).

🔌 Hardware Used:

### **ARDUINO UNO**

- 1x Servo Motor (for gate)
- 2 IR Sensors (entry and exit)
- 2x Ultrasonic Sensors (slot detection)
- 3x LEDs (2 for slots, 1 for gate status)
- Jumper wires, Breadboard, and USB cable



### **RESULT:**

Thus, Mini Projects done successfully