GITAM
(DEEMED TO BE UNIVERSITY)
VISAKHAPATNAM • HYDERABAD • BENGALURU

Internship report on Interfacing sensors with serial communication from Arduino to NodeMCU

Submitted By
Gudipati Rohith
GITAM University, Bengaluru
E-mail: rohithgudipati2002@gmail.com

Under the guidance of
Narendra Reddy T
Scientist - D

During the period
15th May 2023 – 5th July 2023

केन्द्रीय विनिर्माणकारी प्रौद्योगिकी संस्थान
CENTRAL MANUFACTURING TECHNOLOGY INSTITUTE
(An Autonomous R&D Institute
Under Ministry of Heavy Industries, Govt. of India)

CENTRAL MANUFACTURING TECHNOLOGY INSTITUE
TUMKUR ROAD, BENGALURU-560022, INDIA

INTERNSHIP COMPLETION LETTER

04-07-23

Subject: Request for the Internship completion certificate

Respected Sir/Mam,

I G Rohith (322010405026) third-year Electronics and Communication Engineering student of GITAM University Bengaluru I'm thankful to CMTI for having provided me with the opportunity to carry out my internship on the topic Internship report on 'Interfacing sensors with serial communication from Arduino to NodeMCU' under the guidance of Mr. Narendra Reddy T, Scientist-D, Additive and Special Manufacturing Process (D-SMPM).

I am grateful for all the experience and knowledge I have received during my internship from

15.05.2023 to 04.07.2023.

I hereby request you kindly issue the internship completion certificate.

Thanking you,

Yours Sincerely,

G Rohith

Intern

D – SMPM

Centre Head (C – SMPM)

Mr. Narendra Reddy T

(Scientist – D)

ACKNOWLEDGEMENT

I would like to thank Central Manufacturing Technology Institute (CMTI) for giving me the opportunity to do an internship within the organization. I also would like to all the people that I worked with at Central Manufacturing Technology Institute (CMTI) for their patience and openness they created an enjoyable working environment.

I convey my sincere gratitude and special thanks to Mr. Narendra Reddy T (Scientist-D) (SMPM, CMTI) for guiding me through the internship and to all who guided me in the execution and completion of this internship successfully. Thank you for speaking so favorably of my work ethic. It encourages me to continue to perform at a high standard.

And also acknowledge my deepest gratitude and special thanks to Mr. Vinay Kumar P Section In-charge & Staff AEAMT for his continuous support and for providing me an opportunity to work in CMTI as an Intern.

Rohith G

TABLE OF CONTENTS

About CMTI

Central Manufacturing Technology Institute, CMTI, is a Research & Development organization focusing on providing Technology Solutions to the manufacturing sector and assisting technological growth in the country. CMTI plays a key role in applied research, design, and development (ROAD), technology forecasting, assimilation, and dissemination of manufacturing technology to Indian industries.

CMTI is a registered Government of India Society, an autonomous institution under the administrative control of the Ministry of Heavy Industries, Government of India, governed by a Governing Council which has representation from the machine tool manufacturing and user industries, the Union Government and the Government of Karnataka, the Governing Council evolves Policies and monitors policy deployment. A Research Advisory Board (RAB), Technical Committee with representatives from industries and academia, assists the institute- on matters relating to technology advancement.

Drishti is an online platform developed by CMTI that provides a synergy between industries and young innovators to solve complex challenges. CMTI, under the SAMARTH Udyog Bharat 4.0 Platform of the Department of Heavy Industry (DHI) is setting up a Smart Manufacturing Demo & Development Cell (SMDDC) as a Common Engineering Facility Centre (CEFC) to develop, propagate and support the process of adoption of smart manufacturing practices, by the rapidly growing Indian manufacturing industry. Skilling and Re-Skilling in Smart Manufacturing are one of the local activities of SMDDC.

VISION:

- Achieve Technological Leadership and Excellence in the Quality of Products and Services.
- Establish a Dynamic, flexible, and result-oriented organizational structure.
- Achieve organizational excellence through a transparent, professional management system.

Abstract

This project provides a comprehensive guide on interfacing various sensors with Arduino and NodeMCU using serial communication. The document outlines the process of connecting each sensor to Arduino, reading their data, and transmitting it to NodeMCU for further processing. Step-by-step instructions, along with relevant code snippets, are provided for each sensor. By following this guide, readers will gain the knowledge and skills to successfully interface sensors with Arduino and NodeMCU, enabling real-time data acquisition for various applications.

1. Introduction

1.1 Interfacing different sensors using nodemcu and storing its data in cloud

In the realm of **IoT** (Internet of Things) and sensor-based applications, the ability to interface and collect data from various sensors is crucial. Interfacing sensors with microcontrollers, such as Arduino, and establishing communication with other devices, like NodeMCU, allows us to harness the power of sensor data for monitoring, control, and analysis.

Each sensor offers unique capabilities and finds applications in a wide range of fields, such as weather monitoring, home automation, agriculture, and security systems. By understanding how to interface these sensors and transmit data via serial communication, we can unlock their potential for real-time monitoring and decision-making.

Throughout this project, we will provide step-by-step instructions on connecting each sensor to Arduino, reading their data, and transmitting it to NodeMCU for further processing. This integration allows for centralized data analysis, cloud connectivity, and integration with web or mobile applications.

By following the instructions and utilizing the provided code snippets, readers will gain the necessary knowledge and skills to interface with these sensors effectively. The resulting sensor data can be used for various applications, such as environment monitoring, smart homes, industrial automation, and healthcare systems.

1.1Hardware requirements

1. Arduino
2. Nodemcu
3. PCB Board
4. DHT11 sensor
5. Flame Sensor
6. LDR Module
7. Ultrasonic sensor
8. PIR sensor
9. Soil moisture sensor
10. Rain sensor
11. Heartbeat sensor
12. Connecting wires

DHT11 (Temperature and Humidity Sensor):

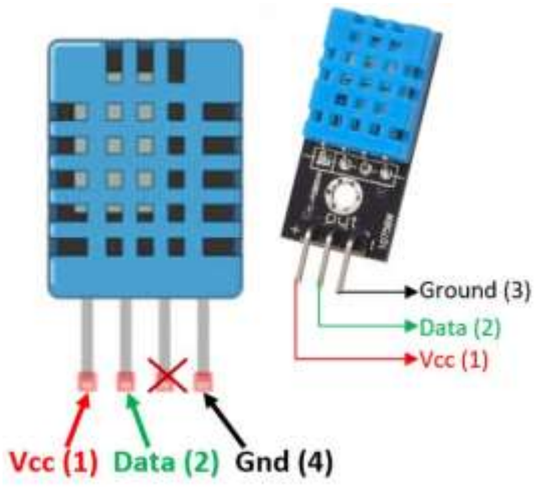
The DHT11 sensor is a popular sensor used to measure temperature and humidity in the surrounding environment. It provides digital outputs for temperature and humidity readings and is relatively low-cost and easy to use.

Working principle:

It operates based on the principle that the humidity sensing component's electrical resistance changes with humidity variations, while the thermistor's resistance changes with temperature. The sensor can determine the corresponding temperature and humidity values by measuring these resistances.

DHT11 Specifications

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%



Applications

- Measure temperature and humidity
- Local Weather station
- Automatic climate control
- Environment monitoring

Flame Sensor:

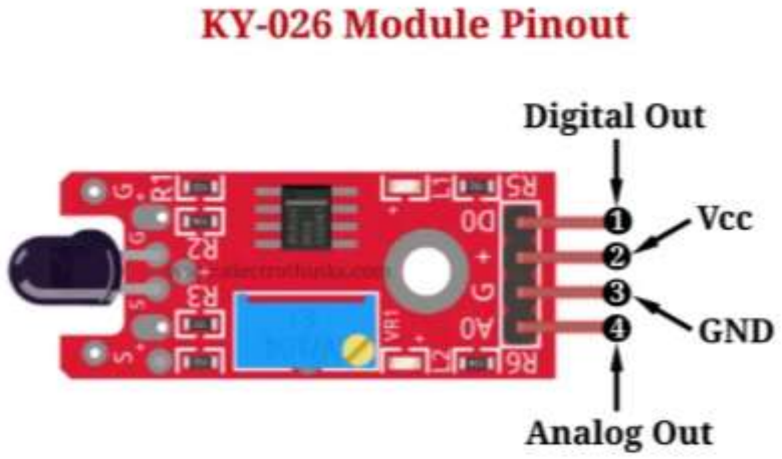
A flame sensor is designed to detect the presence of fire or flames. It typically consists of an infrared (IR) receiver that detects the infrared radiation emitted by flames. Flame sensors are commonly used in fire detection and safety systems.

Working principle:

This sensor/detector can be built with an electronic circuit using a receiver like electromagnetic radiation. This sensor uses the infrared flame flash method, which allows the sensor to work through a coating of oil, dust, water vapor, otherwise ice.

Flame Sensor Specifications:

- Operating Voltage - 3.3V – 5V
- Operating Current - 15 mA
- Comparator chip - LM393
- Sensor type - YG1006 Photo Transistor
- Sensitivity - Adjustable via potentiometer
- Output type - Digital output / Digital and Analog output
- LED lights indicators - Power (red) and Output (green)
- Detection angle – (0 – 60 degrees)
- Operating temperature - (-25℃ ~ 85℃)
- PCB Size - 3cm X 1.6cm



Applications

- Fire detection
- Use in Firefighting robot
- Fire alarm

LDR (Light Dependent Resistor):

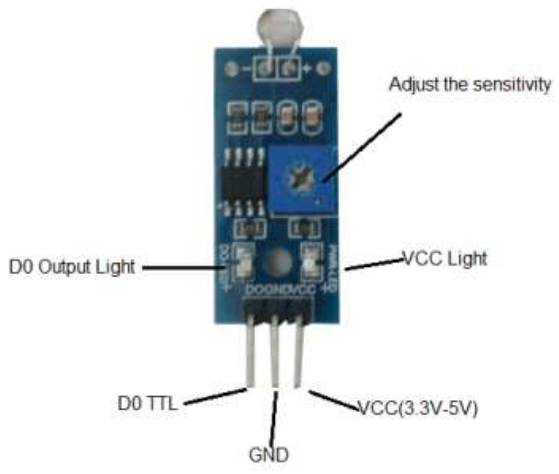
An LDR, also known as a photoresistor, is a light-sensitive resistor that changes its resistance based on the intensity of light falling on it.

Working principle:

LDRs are light-sensitive resistors whose resistance changes in response to the intensity of light falling on them. They contain a photosensitive semiconductor material that exhibits a higher conductivity when exposed to more light and lower conductivity in the absence of light. This change in resistance is utilized to measure the light level in the Surroundings.

LDR Specifications:

- Operating voltage - 5V or 3.3V DC
- Comparator chip - LM393
- Module Pins - 3 pins
- Output type - Digital outputs (D0)
- Sensitivity - Adjustable
- Indicator LED - Output and power LED indicator
- PCB size - 3cm * 1.6cm
- Fixed Hole Diameter - 3mm



Applications

- Detecting the darkness and light.
- Automatic light on/off system
- It is used to measure light intensity.

Ultrasonic Sensor:

Ultrasonic sensors use sound waves at ultrasonic frequencies to measure distances. They emit high-frequency sound pulses and measure the time it takes for the sound waves to bounce back after hitting an object.

Working principle:

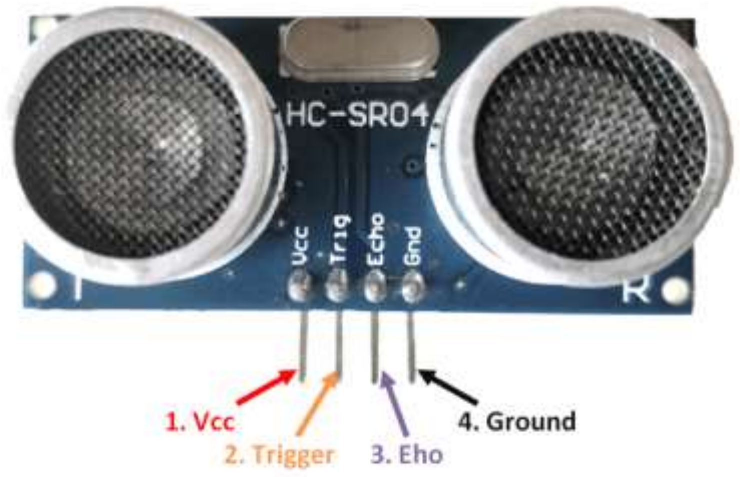
Ultrasonic sensors emit high-frequency sound waves, usually in the ultrasonic range (>20 kHz). These sound waves travel through the air and bounce off objects in their path. The sensor then measures the time it takes for the sound waves to return to the sensor. By knowing the speed of sound, the sensor can calculate the distance to the object based on the time of flight.

The distance can be calculated using the formula:

Distance = (Speed of Sound × Time) / 2

Ultra-Sonic Sensor Features

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz



Applications

- Used to measure the distance.
- Obstacle detection.
- Used in robotics to avoid and detect obstacles
- Map the objects surrounding the sensor by rotating it.

PIR (Passive Infrared) Sensor:

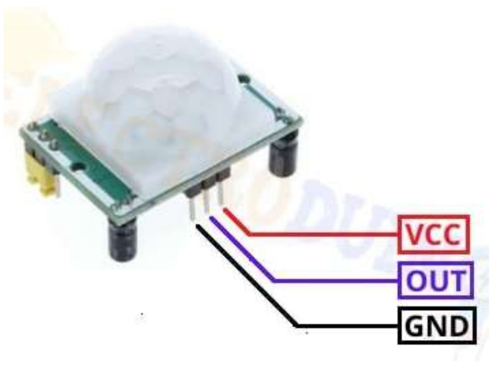
PIR sensors detect infrared radiation emitted by objects in their field of view. They are commonly used for motion detection, as they can sense changes in the infrared energy pattern caused by moving objects.

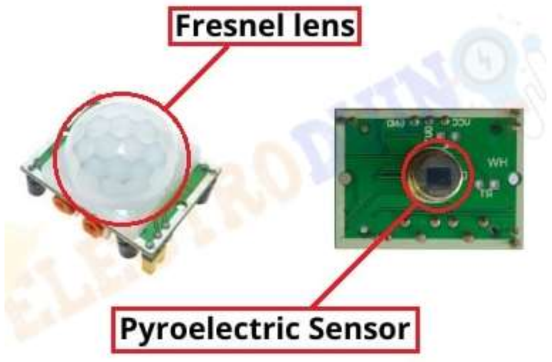
Working principle:

They consist of a pyroelectric sensor, which generates an electrical signal when it detects changes in the infrared energy pattern in its surroundings. When a person or object moves within the sensor's range, it detects the variations in the infrared radiation and triggers a response.

PIR Sensor Features

- Working input voltage - 4.5V to 12V (+5V recommended)
- Output voltage - High: 3.3V TTL, Low: 0V
- Detection angle - Approximately 120 degrees
- Range - Adjustable, up to 7 meters
- Operating modes - Repeatable(H) and Non- Repeatable(H)
- Low power consumption - 65 mA
- Operating Temperature – (-20 – +80 Degrees C).
- PCB Dimensions - 33x25mm, 14mm High not including the Lens.
- Lens Dimensions - 11mm high, 23mmDiameter.
- Weight - 6g





Applications

- Detect Infrared (IR) radiation from objects
- Motion detection
- Automatic lighting system.
- Security system.

Heartbeat Sensor:

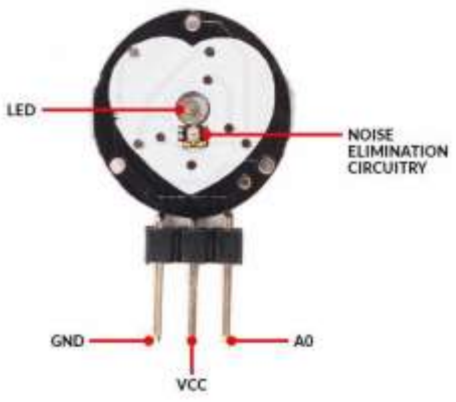
Heartbeat sensors, also known as pulse sensors, detect and measure the heart rate of a person. These sensors usually utilize optical methods to measure blood flow or changes in blood volume. Heartbeat sensors find applications in healthcare monitoring and fitness tracking devices.

Working principle:

Heartbeat sensors typically employ an optical method to measure blood flow or changes in blood volume. They emit light, often using LEDs, into a fingertip or earlobe and measure the light that is transmitted or reflected back. The changes in light intensity correspond to the variations in blood volume, allowing the sensor to detect the heartbeats.

Heartbeat Sensor Features:

- Detection Method - Optical (Light-based)
- Output - Digital or Analog
- Operating Voltage - 3.3V - 5V
- Detection Range - 30 - 200 bpm (beats per minute)
- Sensing Method - Photoplethysmography (PPG)
- Signal Processing - Filtering and Amplification
- Interface - I2C, SPI, Analog Output
- Accuracy - Typically within ±5 bpm
- Dimensions - Varies based on the specific sensor
- Power Consumption - Low



Applications

- Healthcare Monitoring
- Sports and Fitness
- Stress Management
- Biometric Authentication
- Research and Development
- Virtual Reality and Gaming
- Sleep Monitoring

Soil Moisture Sensor:

Soil moisture sensors measure the moisture content or water level in soil. They help monitor soil conditions for efficient irrigation and agriculture applications.

Working principle:

Soil moisture sensors utilize probes or electrodes inserted into the soil to measure the water content in the surrounding soil. They work on the principle that moisture affects the electrical conductivity of the soil. By measuring the electrical resistance or capacitance between the probes, the sensor can determine the moisture level in the soil.

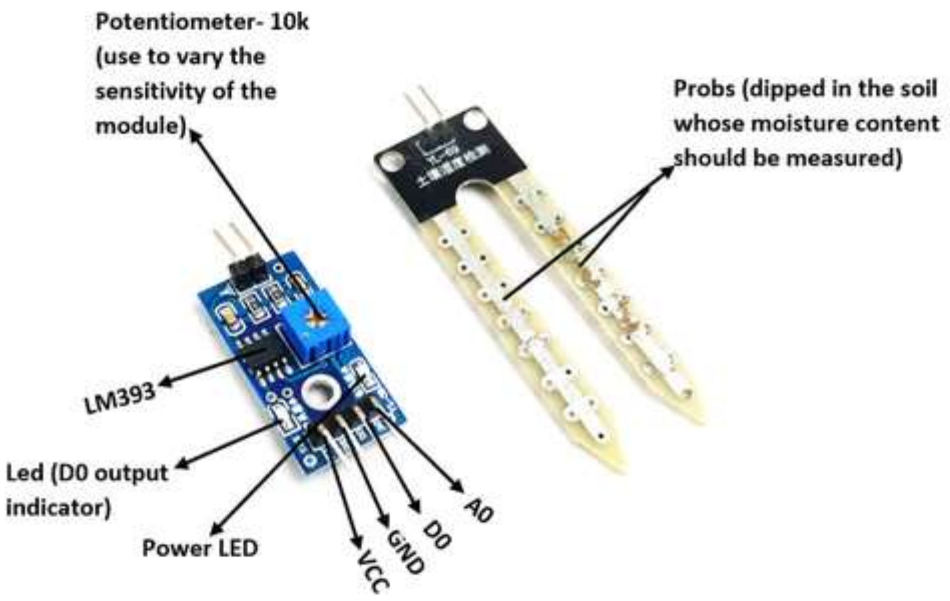
Soil Moisture Features:

The required voltage for working is 5V.

The required current for working is <20mA.

The type of interface is analog.

The required working temperature of this sensor is 10°C~30°C.


Potentiometer- 10k (use to vary the sensitivity of the module)
Probs (dipped in the soil whose moisture content should be measured)
LM393
Led (D0 output indicator)
Power LED
VCC
GND
D0
A0

Applications:

- Agriculture
- Landscape irrigation
- Research
- Simple sensors for gardeners

Rain Sensor:

Rain sensors detect rainfall or the presence of water. They can be used to trigger actions such as closing windows, activating sprinkler systems, or alerting for potential flooding.

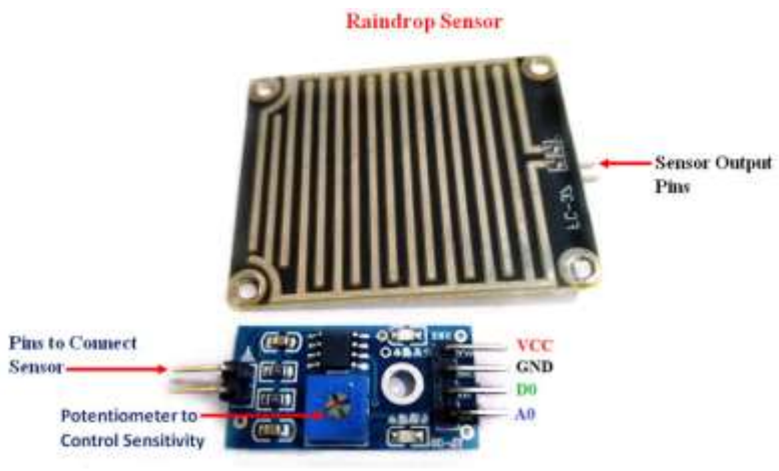
Working principle:

Rain sensors typically utilize a moisture-sensitive surface, such as a printed circuit board (PCB), to detect the presence of water. When raindrops or water come in contact with the sensor's surface, it changes the electrical conductivity, which is detected by the sensor. The change in conductivity indicates the occurrence of rainfall.

Rain Sensor Features:

- Material - Double-sided material with good quality
- Anti-Conductivity - Provides protection against conductivity and oxidation
- Sensor Area - 5cm x 4cm
- Sensor Construction Nickel plated on one side for enhanced conductivity
- Sensitivity Adjustment - Potentiometer for adjusting the sensitivity
- Voltage - 5V
- PCB Size - 3.2cm x 1.4cm
- Comparator - LM393 comparator with wide voltage range

- Output - Clean waveform output
- Driving Capacity - Above 15mA



Applications:

- Automated irrigation systems in agriculture.
- Weather monitoring and forecasting.
- Smart home automation for automatic window closing during rainfall.
- Flood detection and early warning systems.
- Control of outdoor lighting and electronic signage based on rainfall.

NodeMCU:

NodeMCU is an open-source development board based on the ESP8266 microcontroller. It combines the capabilities of Wi-Fi connectivity with a powerful microcontroller, making it ideal for Internet of Things (IoT) projects. NodeMCU boards are designed to provide a simple and cost-effective solution for prototyping and building IoT applications.

Key features of NodeMCU:

Microcontroller: NodeMCU boards are based on the ESP8266 microcontroller, which has built-in Wi-Fi capabilities.

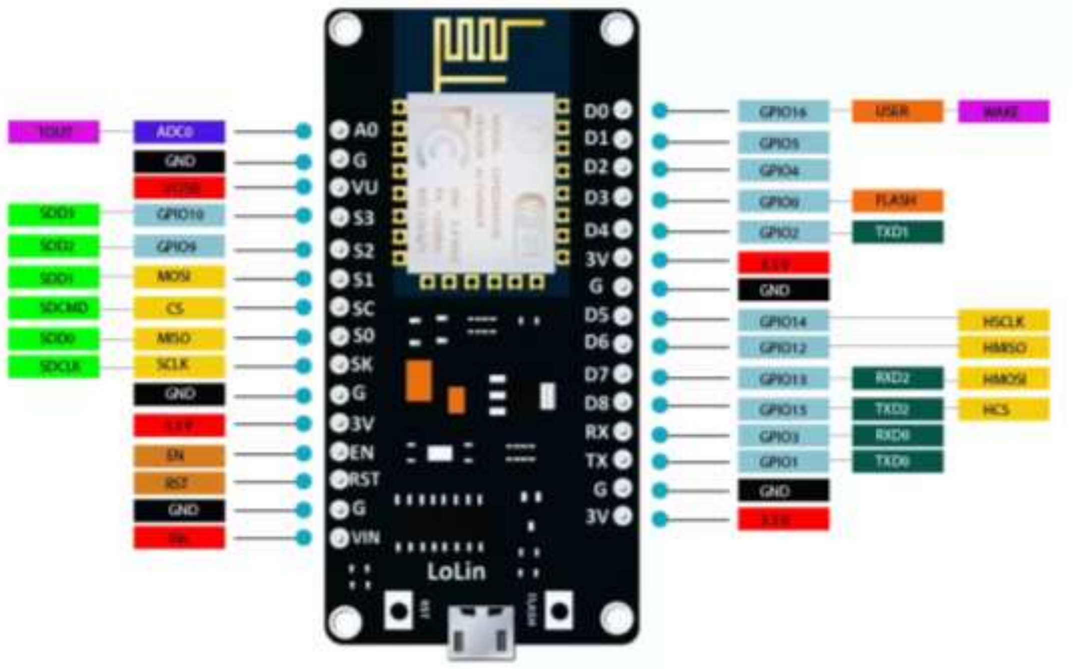
Wi-Fi Connectivity: NodeMCU offers integrated Wi-Fi connectivity, allowing devices to connect to the internet or communicate with other networked devices.

Programming Language: NodeMCU can be programmed using Lua scripting language or the Arduino IDE (Integrated Development Environment).

GPIO Pins: NodeMCU boards have a number of General Purpose Input/Output (GPIO) pins that can be used to interface with sensors, actuators, and other electronic components.

Community Support: NodeMCU has a large and active community, providing resources, libraries, and examples for easy development and troubleshooting.

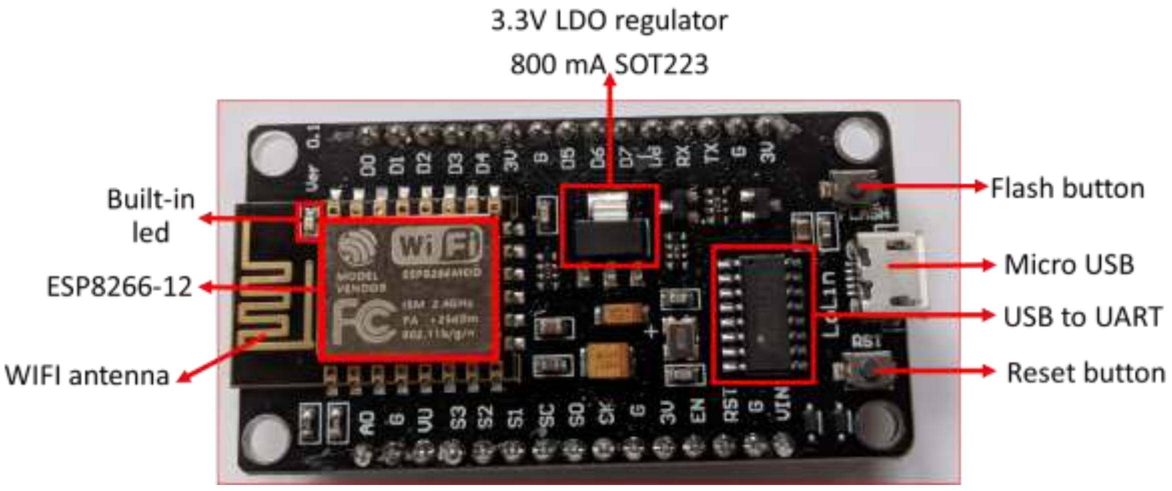
<u>Pinout diagram:</u>



ESP8266 architecture

An ESP8266 is a microcontroller:

- ✓ Low-power, highly-integrated Wi-Fi solution
- ✓ A minimum of 7 external components
- ✓ Wide temperature range: -40°C to +125°C
- ✓ ESP8285 — 8 Mbit flash embedded
- ✓ Different kinds of ESP8266 can be found on the market so your ESP8266 board may differ slightly from the one shown below:
- ✓ ESP8266 microcontroller
- ✓ In the picture above, the ESP8266-12 block is where the processor, memory, and WIFI unit are located. The rest ensures communication with external sensors, USB ports, voltage regulators, etc.
- ✓ the CPU (Central Processing Unit)
- ✓ the memory SRAM (Static Random Access Memory)



The ESP8266 uses a 32-bit processor with 16-bit instructions. It is Harvard architecture which mostly means that instruction memory and data memory are completely separate.

The ESP8266 has on die program Read-Only Memory (ROM) which includes some library code and a first-stage boot loader. All the rest of the code must be stored in external Serial flash memory (provides only serial access to the data - rather than addressing individual bytes, the user reads or writes large contiguous groups of bytes in the address space serially).

Depending on your ESP8266, the amount of available flash memory can vary.

As with any other microcontroller, ESP8266 has a set of GPIO pins (General Purpose Input(Output pins) that we can use to “control” external sensors.

Our ESP8266 has 17 GPIO pins but only 11 can be used (among 17 pins, 6 are used for communication with the on-board flash memory chip). It also has an analog input (to convert a voltage level into a digital value that can be stored and processed in the ESP8266).

It also has a WIFI communication to connect your ESP8266 to your WIFI network, connect to the internet, host a web server, let your smartphone connect to it, etc.

Another advantage of an ESP8266 is that it can be programmed as any other microcontroller and especially any Arduino.

Arduino:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of a microcontroller board and a development environment that allows users to write and upload code to the board. Arduino boards are widely used for various applications, from simple hobby projects to complex automation systems.

Key features of Arduino:

Microcontroller Board: Arduino boards are equipped with a microcontroller, such as the AT mega series, which serves as the brain of the system.

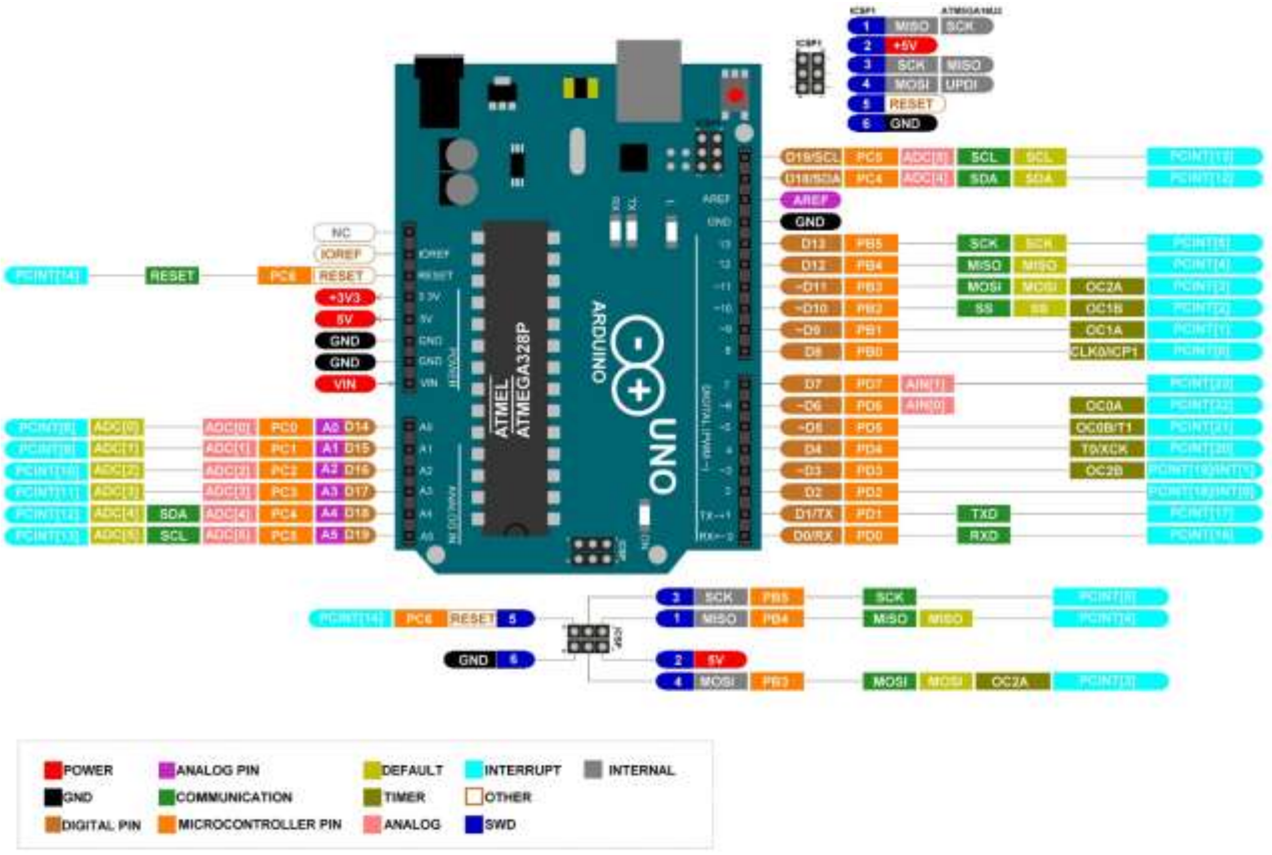
Programming Language: Arduino uses a simplified version of C++ programming language, making it accessible to beginners.

IDE: The Arduino IDE provides a user-friendly interface for writing, compiling, and uploading code to the Arduino board.

Extensibility: Arduino boards can be easily expanded using shields, which are pre-built modules that provide additional functionality such as Wi-Fi, Bluetooth, motor control, or sensor interfaces.

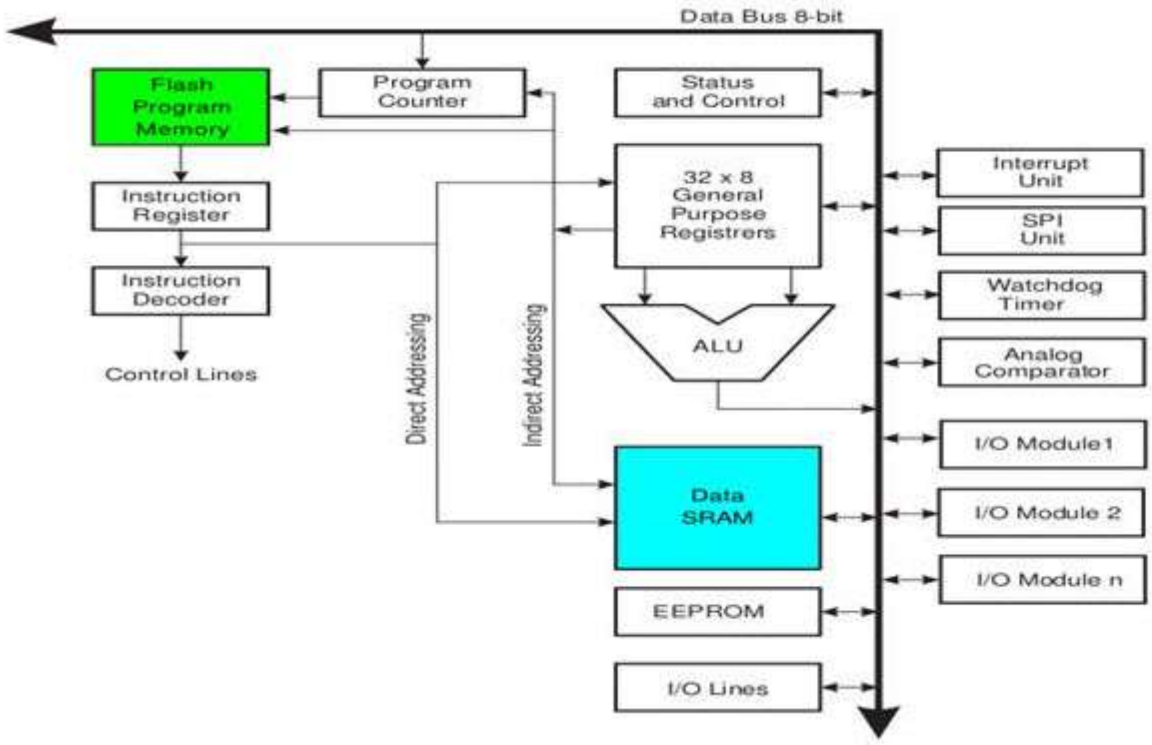
Community and Libraries: Arduino has a large and supportive community, offering extensive libraries and examples that simplify development and troubleshooting.

Pinout diagram:


ICSP1
ATMEGA16U2
1 MISO SCK
2 +5V
3 SCK MISO
4 MOSI UPDI
5 RESET
6 GND
D19/SCL PC5 ADC[5] SCL SCL PCINT[13]
D18/SDA PC4 ADC[4] SDA SDA PCINT[12]
AREF
GND
D13 PB5 SCK SCK PCINT[5]
D12 PB4 MISO MISO PCINT[4]
~D11 PB3 MOSI MOSI OC2A PCINT[3]
~D10 PB2 SS SS OC1B PCINT[2]
~D9 PB1 OC1A PCINT[1]
D8 PB0 CLKO/ICP1 PCINT[0]
D7 PD7 AIN[1] PCINT[23]
~D6 PD6 AIN[0] OC0A PCINT[22]
~D5 PD5 OC0B/T1 PCINT[21]
D4 PD4 T0/XCK PCINT[20]
~D3 PD3 OC2B PCINT[19]/INT[1]
D2 PD2 PCINT[18]/INT[0]
D1/TX PD1 TXD PCINT[17]
D0/RX PD0 RXD PCINT[16]
NC
IOREF
PCINT[14] RESET PC6 RESET
+3V3
5V
GND
GND
VIN
PCINT[8] ADC[0] ADC[0] PC0 A0 D14
PCINT[9] ADC[1] ADC[1] PC1 A1 D15
PCINT[10] ADC[2] ADC[2] PC2 A2 D16
PCINT[11] ADC[3] ADC[3] PC3 A3 D17
PCINT[12] ADC[4] SDA ADC[4] PC4 A4 D18
PCINT[13] ADC[5] SCL ADC[5] PC5 A5 D19
ATMEL ATMEGA328P
ARDUINO
UNO
3 SCK PB5 SCK PCINT[5]
1 MISO PB4 MISO MISO PCINT[4]
PCINT[14] PC6 RESET 5
GND 6
2 5V
4 MOSI PB3 MOSI MOSI OC2A PCINT[3]
POWER
GND
DIGITAL PIN
ANALOG PIN
COMMUNICATION
MICROCONTROLLER PIN
DEFAULT
TIMER
ANALOG
INTERRUPT
OTHER
SWD
INTERNAL

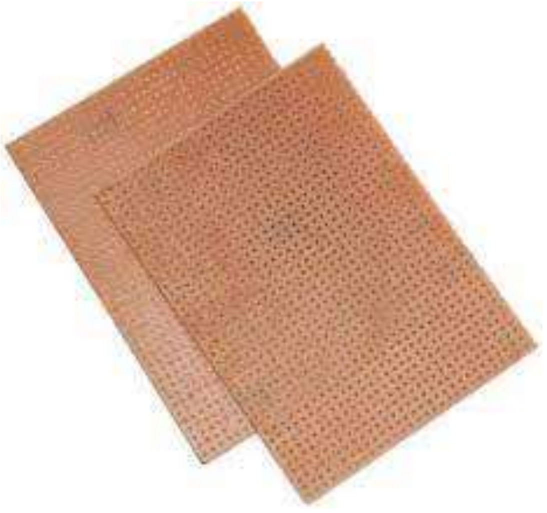
Arduino Architecture

Basically, the processor of the Arduino board uses the Harvard architecture where the program code and program data have separate memory. It consists of two memories such as program memory and data memory. Wherein the data is stored in data memory and the code is stored in the flash program memory. The Atmega328 microcontroller has 32kb of flash memory, 2kb of SRAM 1kb of EPROM and operates with a 16MHz clock speed.



Zero PCB:

Zero PCB is basically a general-purpose printed circuit board (PCB), also known as a perf board or DOT PCB. It is a thin rigid copper sheet with holes pre-drilled at standard intervals across a grid with 2.54mm (0.1-inch) spacing between holes. Each hole is encircled by a round or square copper pad so that component lead can be inserted into the hole and soldered around the pad without short-circuiting the nearby pads and other leads. For connecting the lead of component with another lead, solder these together or join these using a suitable conducting wire.



1.3 Software used

Cloud **- Blynk IoT**

Software – **Arduino IDE**

1. Cloud - Blynk IoT:

Blynk is an IoT platform that allows users to easily build and control connected devices. It provides a cloud-based infrastructure that enables communication between IoT devices and mobile applications. With Blynk, you can create custom user interfaces (UI) using drag-and-drop widgets, which can be connected to various IoT hardware platforms.

Blynk provides a range of features, including real-time data monitoring, push notifications, remote device control, data logging, and integration with popular IoT development boards and protocols. It supports both iOS and Android platforms, making it easy to create mobile apps that interact with your IoT devices.

2. Arduino IDE:

Arduino IDE (Integrated Development Environment) is a software application used for programming Arduino microcontrollers. Arduino is an open-source electronics platform that provides a range of hardware boards and a programming language based on Wiring.

The Arduino IDE allows you to write and upload code to Arduino boards, making it possible to control and interact with various sensors, actuators, and other electronic components. It provides a simplified programming environment with a straightforward syntax and a library of functions for working with Arduino hardware.

The Arduino IDE supports a wide range of Arduino boards, making it accessible for beginners and experienced developers alike. It also has a large community of users and extensive documentation, making it easier to find examples, tutorials, and troubleshooting help.

2. Methodology

Sensor Selection: Choose the sensors you want to interface with Arduino and NodeMCU. In this case, you have selected DHT11, Flame sensor, LDR, Ultrasonic, PIR, heartbeat, soil moisture, and rain sensor.

Hardware Setup: a. Connect Arduino to the sensors: Connect each sensor to the appropriate pins on the Arduino board following the sensor's datasheet or pinout diagram. Ensure that you have the necessary resistors or other components required for sensor connectivity. b. Connect NodeMCU to Arduino: Establish a serial communication link between Arduino and NodeMCU. Connect the TX (transmit) pin of Arduino to the RX (receive) pin of NodeMCU, and the RX pin of Arduino to the TX pin of NodeMCU. Also, connect the ground (GND) pins of both boards together.

Arduino Programming:

a. Install Required Libraries: Depending on the sensors you are using, install the necessary libraries in the Arduino IDE. These libraries provide functions and methods to interact with the sensors.

b. Read Sensor Data: Write Arduino code to read data from each sensor individually. Utilize the sensor libraries to interface with the sensors and retrieve the relevant data.

c. Serial Communication: Use the Serial library in Arduino to establish a serial communication channel with NodeMCU. Send the sensor data through the serial port using the Serial.print() or Serial.write() functions.

NodeMCU Programming:

a. Install Required Libraries: Install the required libraries in the Arduino IDE for NodeMCU. These libraries should include support for serial communication.
b. Serial Communication Setup: Initialize the serial communication on NodeMCU using the appropriate baud rate and other settings that match the configuration on the Arduino side.

c. Receive and Process Data: Implement code on NodeMCU to receive the sensor data sent by Arduino via serial communication. Parse the received data and process it as per your requirements.

Testing and Debugging:

a. Upload the Arduino code to the Arduino board and the NodeMCU code to the NodeMCU board.

b. Open the serial monitor on Arduino IDE to verify that the sensor data is being successfully transmitted from Arduino to NodeMCU.

c. Monitor the output on the NodeMCU side to ensure that the received data is being processed correctly.

Integration and Application:

a. Use the processed sensor data on the NodeMCU board for further application-specific tasks. For example, you can send the data to a server, display it on an LCD, trigger actions based on sensor readings, etc.

Throughout the process, refer to the datasheets and documentation of the sensors, Arduino, and NodeMCU boards, as well as the libraries you are using. Troubleshoot any issues that arise, such as incorrect connections or incompatible library versions.

Remember to follow safety precautions and handle the sensors, boards, and components carefully to prevent damage or injury.

2.1 Algorithm

Initialize Arduino and NodeMCU:

- Connect Arduino and NodeMCU via appropriate communication interface (e.g., USB cable or UART).
- Set up the Arduino and NodeMCU development environments.

Connect the sensors to Arduino:

- Identify the pin connections required for each sensor (consult the datasheets or documentation).
- Connect each sensor to the appropriate digital or analog pins on Arduino.

Read sensor data using Arduino:

- Write code on Arduino to read data from each sensor individually.
- Utilize the corresponding libraries or code examples for each sensor to simplify the process.
- Store the sensor data in variables or data structures for further processing.

Transmit sensor data from Arduino to NodeMCU:

- Set up serial communication on Arduino and NodeMCU.
- Use the Serial library in Arduino to send sensor data via the serial port.
- Write code on NodeMCU to receive the data through the serial port.
- Parse the received data and store it in appropriate variables or data structures on NodeMCU.

Process and utilize the sensor data on NodeMCU:

- Implement the necessary logic on NodeMCU to process the received sensor data.
- Perform any required calculations, comparisons, or transformations on the data.
- Utilize the processed data for your specific application or task.

Repeat steps 3-5 for each sensor:

Follow the same procedure for each sensor, adapting the code and connections accordingly.

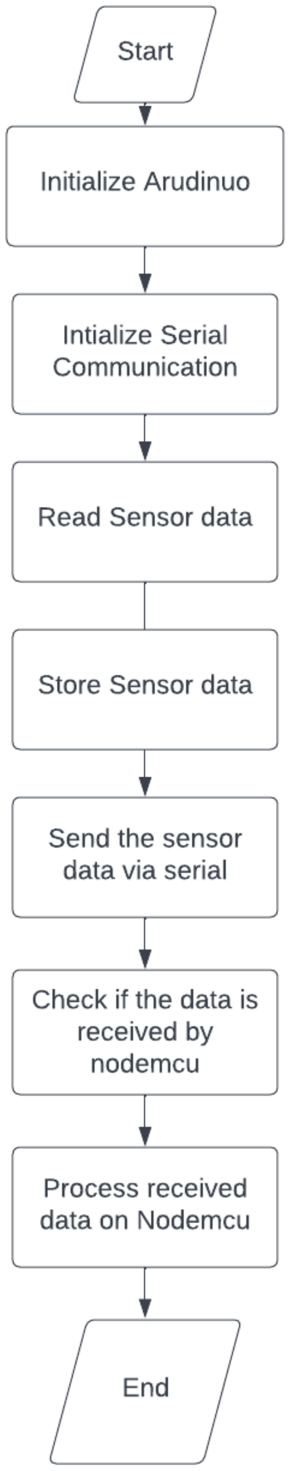
Test and debug:

- Upload the Arduino and NodeMCU code.
- Ensure that the sensors are properly connected and powered.
- Monitor the serial output on NodeMCU to verify the data reception and processing.
- Debug any issues that may arise during testing.

Customize and expand:

- Modify the code or algorithm as per your specific requirements.
- Add additional features, such as data logging, remote access, or real-time visualization, if desired.

2.2 Flowchart


Start
Initialize Arudinuo
Intialize Serial Communication
Read Sensor data
Store Sensor data
Send the sensor data via serial
Check if the data is received by nodemcu
Process received data on Nodemcu
End

3. Implementation

Arduino Side:

- Connect the sensors (DHT11, Flame sensor, LDR, Ultrasonic, PIR, heartbeat, soil moisture, rain sensor) to the Arduino board following their pin configurations.
- Install the required libraries (SoftwareSerial, DHT) in your Arduino IDE.
- Copy and paste the provided Arduino code into a new sketch in your Arduino IDE.
- Modify the code if necessary to match your sensor pin connections.
- Upload the code to the Arduino board.
- Open the Serial Monitor to observe the sensor readings and ensure they are correct.
- Verify that the Arduino board is properly connected to the NodeMCU for serial communication.

NodeMCU Side:

- Connect the NodeMCU board to your computer via USB.
- Install the required libraries (ESP8266WiFi, BlynkSimpleEsp8266) in your Arduino IDE.
- Open a new sketch in your Arduino IDE and copy the provided NodeMCU code into it.
- Modify the code if necessary to match your Wi-Fi credentials and Blynk authentication token.
- Upload the code to the NodeMCU board.
- Verify that the NodeMCU is connected to the Wi-Fi network by checking the Serial Monitor.
- Open the Blynk app on your mobile device and create a new project.
- Add the required widgets (e.g., Gauges, Value Displays) to visualize the sensor data.
- Assign the appropriate virtual pins in the Blynk app for each sensor.
- Run the Blynk app and ensure that it is connected to the Blynk server.

- Check the Blynk app to see if the sensor data is updating correctly.

Testing and Troubleshooting:

- Ensure that the Arduino and NodeMCU are connected and powered on.
- Check the serial communication connections between the Arduino and NodeMCU.
- Verify that the Wi-Fi credentials and Blynk authentication token are correct.
- Double-check the pin connections for each sensor on the Arduino board.
- Monitor the Serial Monitor output on both Arduino and NodeMCU to identify any potential issues.
- Use the Blynk app to monitor the sensor values and check if they are updating correctly.
- Debug any errors or issues that may arise, such as incorrect sensor readings or connectivity problems.

By following these steps, you should be able to successfully interface the sensors with Arduino, transmit the data via serial communication to NodeMCU, and visualize the sensor values using the Blynk app. Remember to test and troubleshoot at each step to ensure a smooth implementation.

3.1CODE

Arduino:

Code:

```
#include <SoftwareSerial.h>
#include <DHT.h>

#define TRIGGER_PIN 3
#define ECHO_PIN 4
#define DHT_PIN 2
#define LDR_PIN A0
#define FLAME_PIN 7
#define PIR_PIN 8
#define SOIL_MOISTURE_PIN A1
#define RAIN_PIN A2
#define HEARTBEAT_PIN A3
#define DHT_TYPE DHT11
SoftwareSerial espSerial(5, 6);
DHT dht(DHT_PIN, DHT_TYPE);
```

```
void setup() {
  Serial.begin(9600);
  espSerial.begin(9600);

  pinMode(TRIGGER_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
  pinMode(FLAME_PIN, INPUT);
  pinMode(PIR_PIN, INPUT);
  pinMode(SOIL_MOISTURE_PIN, INPUT);
  pinMode(RAIN_PIN, INPUT);
  pinMode(HEARTBEAT_PIN, INPUT);
  dht.begin();

  delay(2000);
}

void loop() {
  // Read temperature and humidity from DHT11 sensor
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Read distance from ultrasonic sensor
  long duration, distance;
  // Trigger the ultrasonic sensor
  digitalWrite(TRIGGER_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIGGER_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER_PIN, LOW);

  // Measure the echo duration
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculate the distance in centimeters
  distance = duration * 0.034 / 2;

  // Read value from LDR sensor
  int ldrValue = analogRead(LDR_PIN);
  int mappedLdrValue = map(ldrValue, 0, 1023, 0, 100); // Map the LDR value to
a range of 0 to 100

  // Read value from flame sensor
  int flameValue = digitalRead(FLAME_PIN);

  // Read value from PIR sensor
  int pirValue = digitalRead(PIR_PIN);
  // Read value from soil moisture sensor
  int soilMoistureValue = analogRead(SOIL_MOISTURE_PIN);
  int mappedSoilMoistureValue = map(ldrValue, 0, 1023, 0, 100); // Map the SM
value to a range of 0 to 100

  // Read value from rain sensor
  int rainValue = analogRead(RAIN_PIN);
  int mappedRainValue = map(rainValue, 0, 1023, 0, 100);

  // Read value from heartbeat sensor
  int heartbeatValue = analogRead(HEARTBEAT_PIN);

  // Print the readings to Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  Serial.print("LDR Value: ");
  Serial.println(mappedLdrValue);
  Serial.print("Flame Value: ");
  Serial.println(flameValue);
  Serial.print("PIR Value: ");
  Serial.println(pirValue);
  Serial.print("Soil Moisture: ");
  Serial.println(mappedSoilMoistureValue);
  Serial.print("Rain Value: ");
  Serial.println(mappedRainValue);
  Serial.print("Heartbeat Value: ");
  Serial.println(heartbeatValue);

  // Send the data to NodeMCU via software serial
  String data = String(temperature) + "," + String(humidity) + "," +
String(distance) + "," +  String(mappedLdrValue) + "," + String(flameValue) +
"," + String(pirValue) + "," + String(mappedSoilMoistureValue) + "," +
String(mappedRainValue) + "," + String(heartbeatValue);
  espSerial.println(data);
  Serial.println("Data sent to NodeMCU: " + data);
delay(1000);
}
```

Serial monitor Output:

```
Temperature: 27.20 °C
Humidity: 62.00 %
Distance: 118 cm
LDR Value: 11
Flame Value: 0
PIR Value: 1
Soil Moisture: 11
Rain Value: 97
Heartbeat Value: 50
Data sent to NodeMCU: 27.20,62.00,118,11,1,0,11,97,50
```

Nodemcu:

Code:

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "_//Auth_token//_";
char ssid[] = "__//wifi_name//__";
char password[] = "__//wifi_password//__";

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Blynk.begin(auth, ssid, password);
}

void loop() {
  Blynk.run();

  if (Serial.available()) {
    String receivedData = Serial.readStringUntil('\n');
    // Parse the received data
    int commaIndex1 = receivedData.indexOf(',');
    int commaIndex2 = receivedData.indexOf(',', commaIndex1 + 1);
    int commaIndex3 = receivedData.indexOf(',', commaIndex2 + 1);
    int commaIndex4 = receivedData.indexOf(',', commaIndex3 + 1);
    int commaIndex5 = receivedData.indexOf(',', commaIndex4 + 1);
    int commaIndex6 = receivedData.indexOf(',', commaIndex5 + 1);
    int commaIndex7 = receivedData.indexOf(',', commaIndex6 + 1);
    int commaIndex8 = receivedData.indexOf(',', commaIndex7 + 1);

    float temperature = receivedData.substring(0, commaIndex1).toFloat();
    float humidity = receivedData.substring(commaIndex1 + 1,
commaIndex2).toFloat();
```

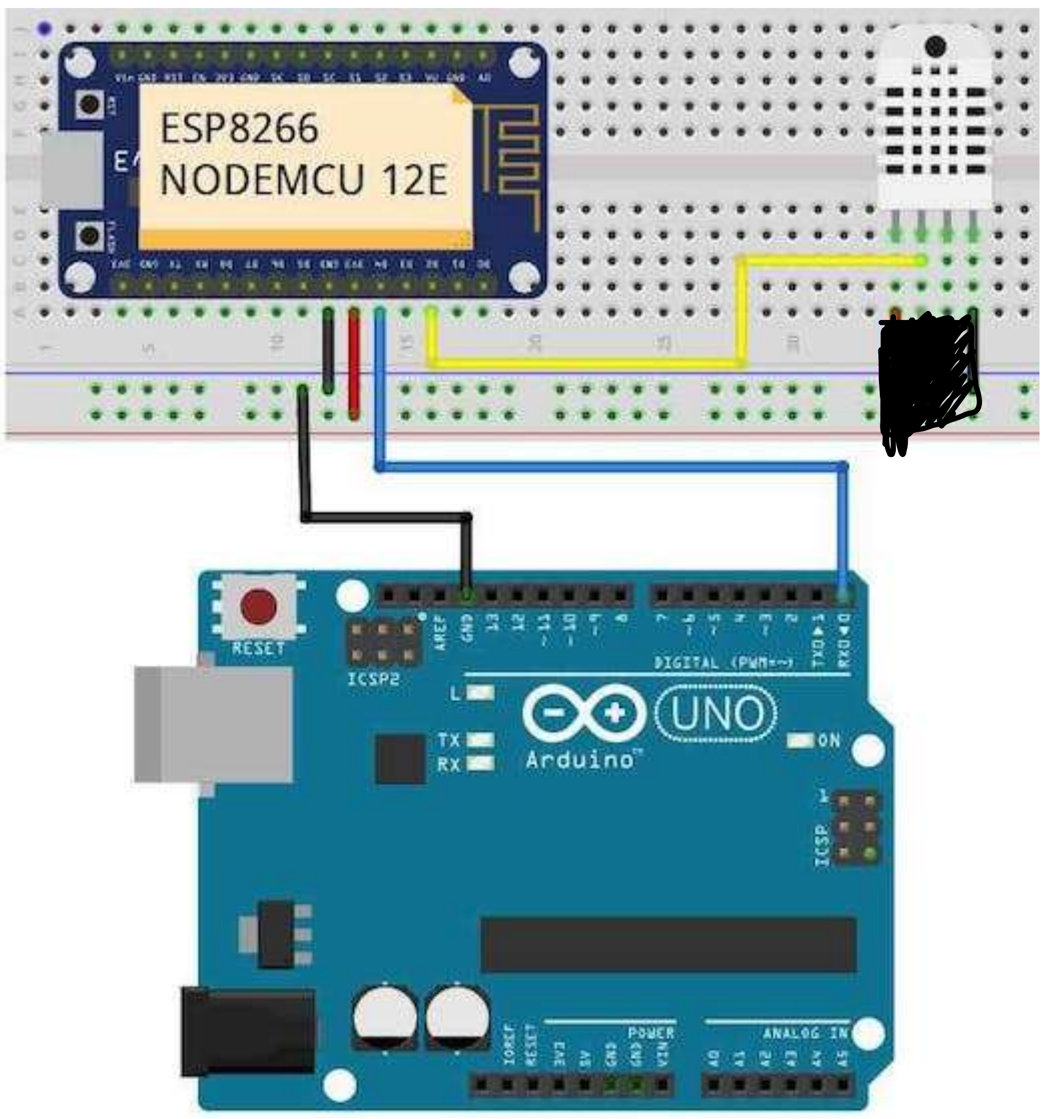
```
    long distance = receivedData.substring(commaIndex2 + 1,
commaIndex3).toInt();
    int ldrValue = receivedData.substring(commaIndex3 + 1,
commaIndex4).toInt();
    int flameValue = receivedData.substring(commaIndex5 + 1,
commaIndex6).toInt();
    int pirValue = receivedData.substring(commaIndex4 + 1,
commaIndex5).toInt();
    int soilMoistureValue = receivedData.substring(commaIndex6 + 1,
commaIndex7).toInt();
    int rainValue = receivedData.substring(commaIndex7 + 1,
commaIndex8).toInt();
    int heartbeatValue = receivedData.substring(commaIndex8 + 1).toInt();

    // Print the readings to Serial Monitor
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" °C");
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.println(" %");
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    Serial.print("LDR Value: ");
    Serial.println(ldrValue);
    Serial.print("Flame Value: ");
    Serial.println(flameValue);
    Serial.print("PIR Value: ");
    Serial.println(pirValue);
    Serial.print("Soil Moisture: ");
    Serial.println(soilMoistureValue);
    Serial.print("Rain Value: ");
    Serial.println(rainValue);
    Serial.print("Heartbeat Value: ");
    Serial.println(heartbeatValue);
    // Update Blynk virtual pins with the sensor values
    Blynk.virtualWrite(V0, temperature);
    Blynk.virtualWrite(V1, humidity);
    Blynk.virtualWrite(V2, distance);
    Blynk.virtualWrite(V3, ldrValue);
    Blynk.virtualWrite(V5, flameValue);
    Blynk.virtualWrite(V8, pirValue);
    Blynk.virtualWrite(V6, soilMoistureValue);
    Blynk.virtualWrite(V4, rainValue);
    Blynk.virtualWrite(V7, heartbeatValue);
  }
}
```

Serial monitor Output:

```
Temperature: 27.20 °C
Humidity: 62.00 %
Distance: 118 cm
LDR Value: 11
Flame Value: 0
PIR Value: 1
Soil Moisture: 11
Rain Value: 97
Heartbeat Value: 50
```

3.2 Circuit diagram :

Example for DHT11 sensor

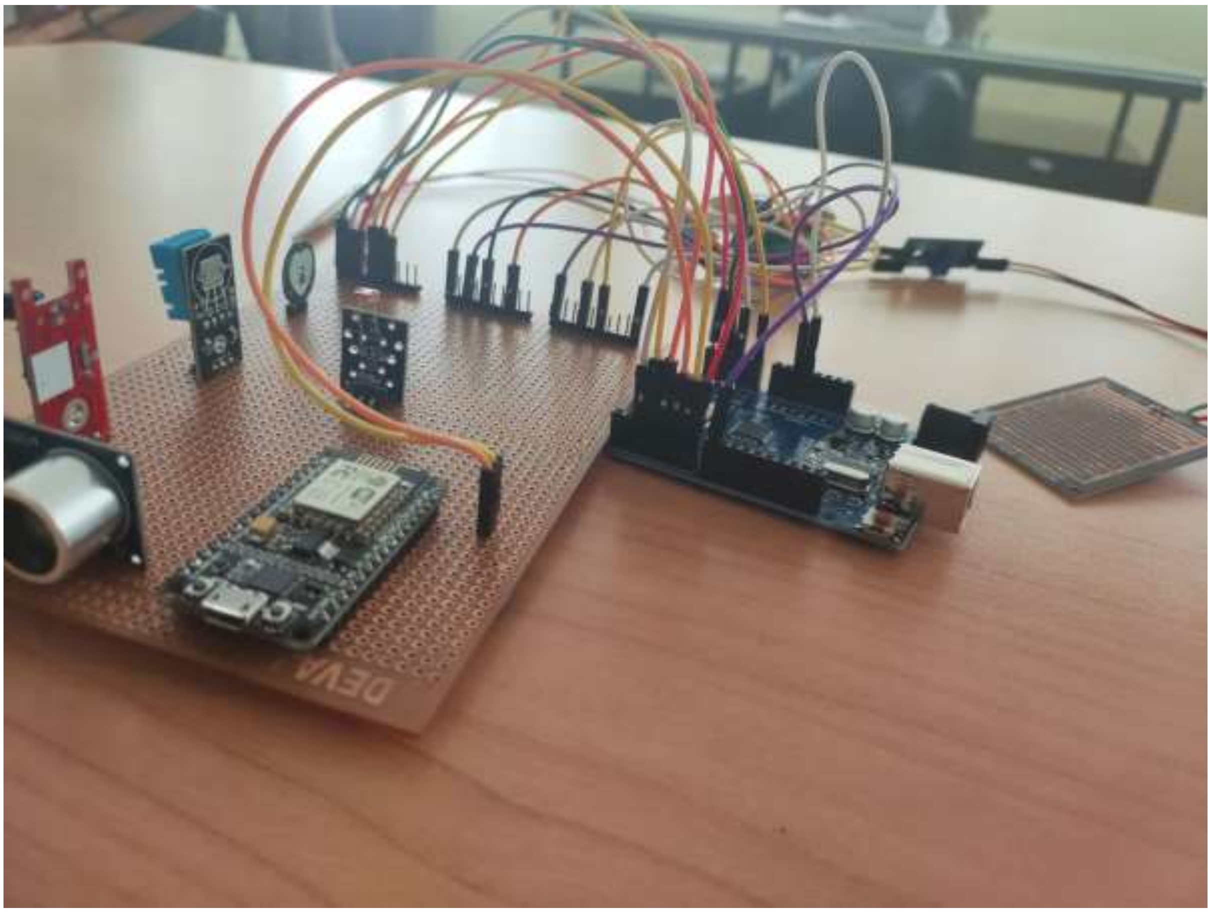


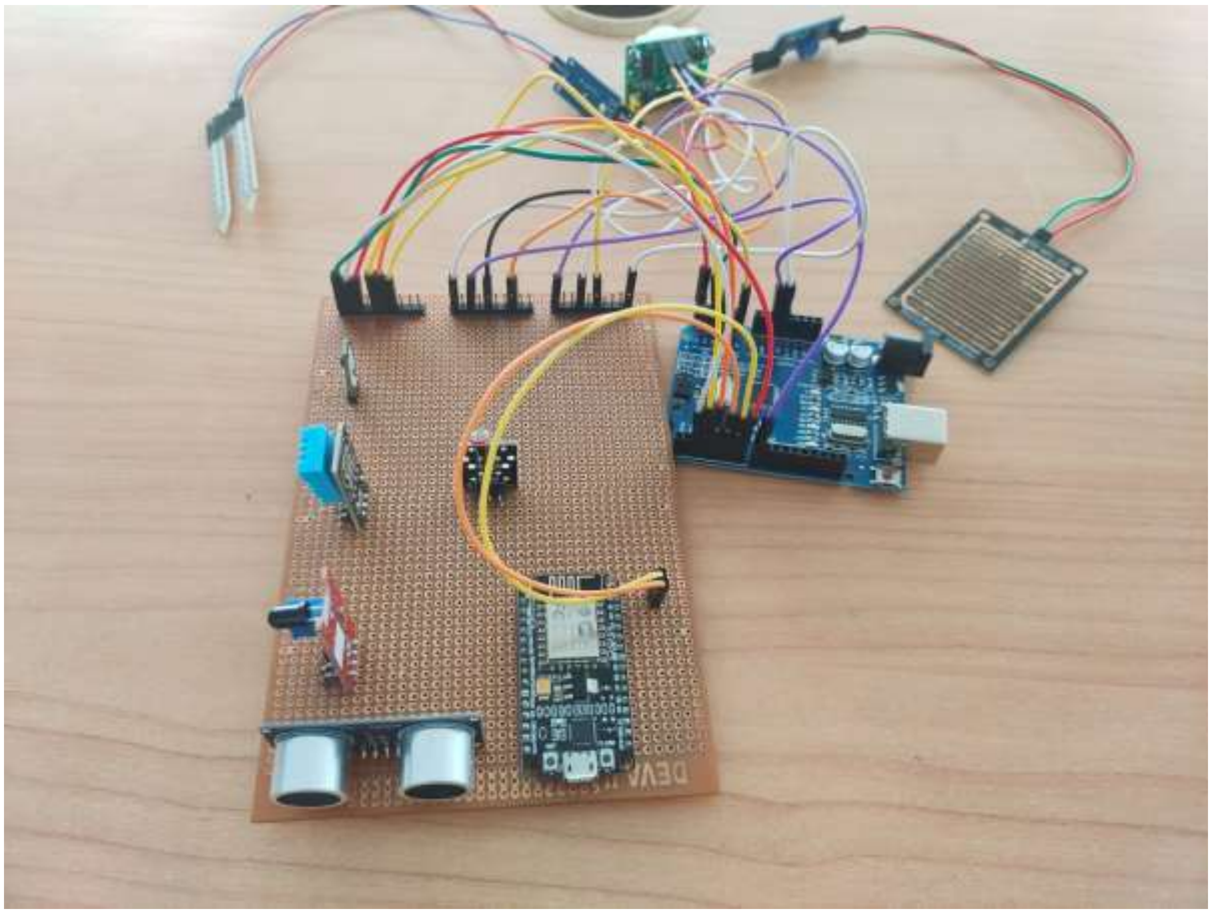
[redacted]

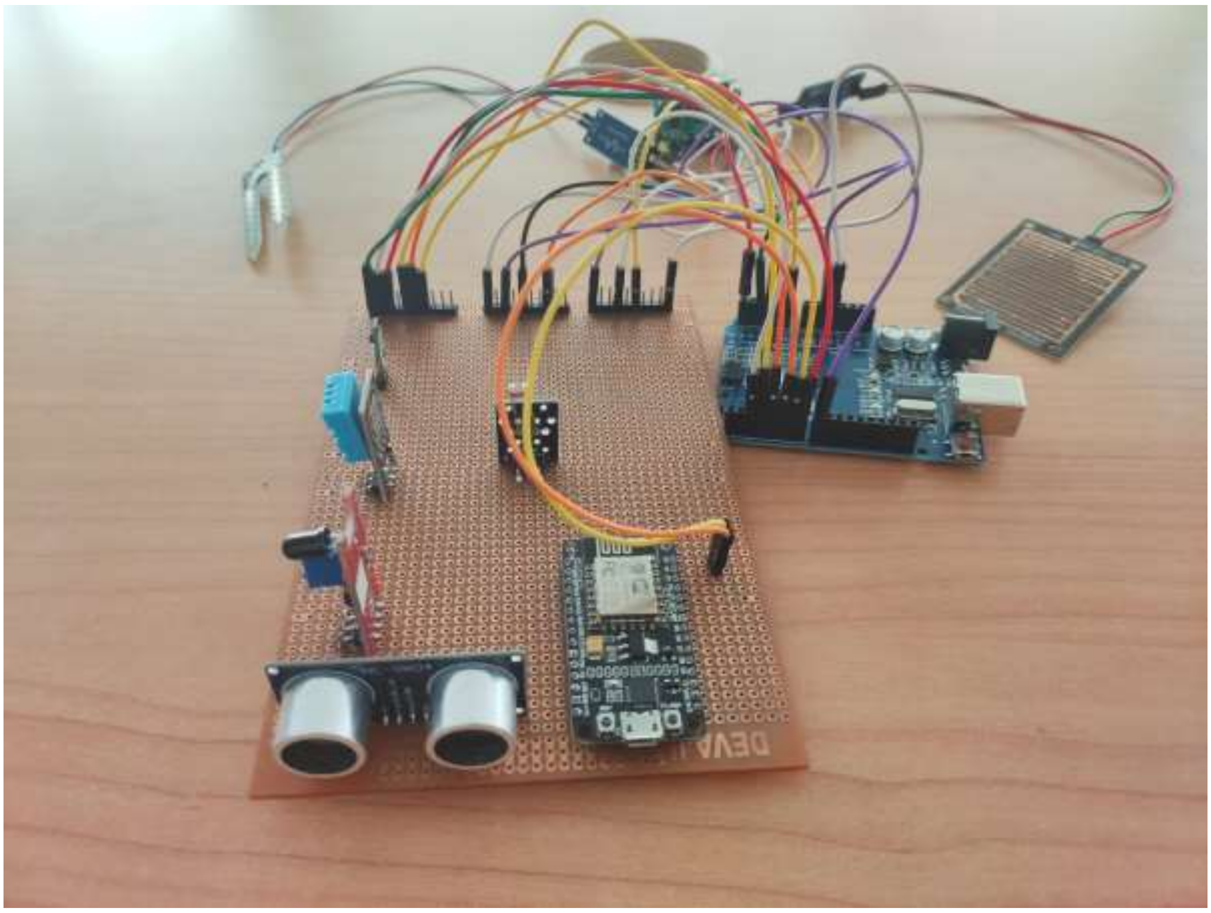
NOTE:

DO it for all sensors, just connect vcc, ground and data pin to Arduino

3.3 MODEL:

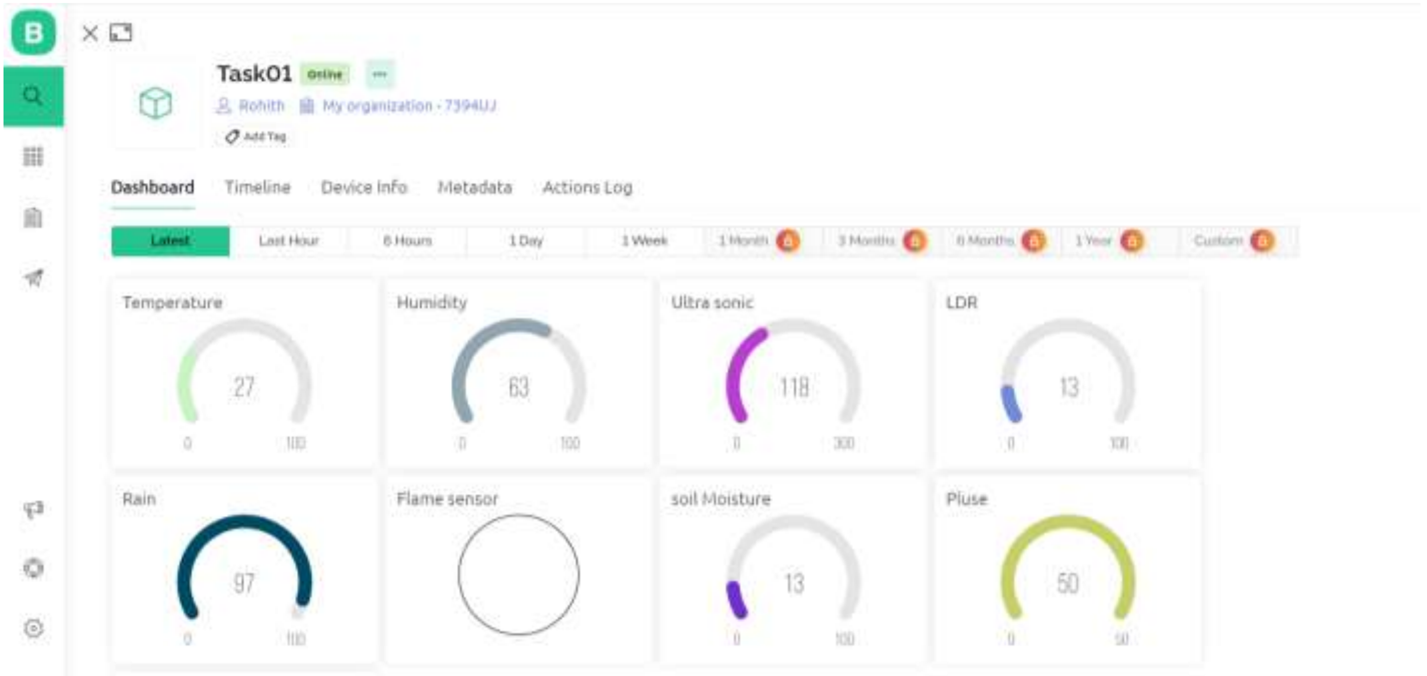




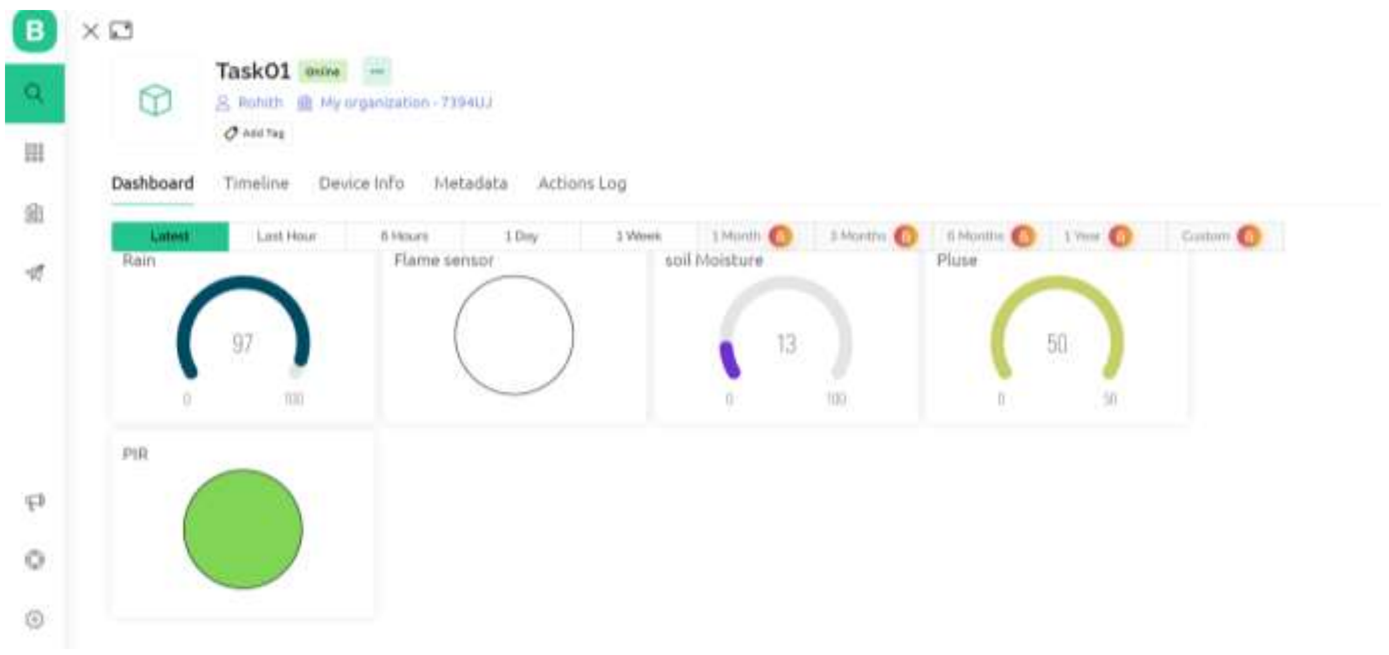


4. Results

Output in Blynk IOT Cloud:



Temperature 27 (0–100)
Humidity 63 (0–100)
Ultra sonic 118 (0–300)
LDR 13 (0–100)
Rain 97 (0–100)
Flame sensor
soil Moisture 13 (0–100)
Pluse 50 (0–50)



4.1 Discussions

Advantages of Serial Communication:

- Explain why serial communication is chosen for interfacing sensors with Arduino and NodeMCU.
- Discuss the benefits of using serial communication for data transmission between the two devices.
- Highlight the simplicity and reliability of serial communication in this context.

Sensor Selection and Applications:

- Discuss the importance of selecting the appropriate sensors for the intended application.
- Explain the specific applications of each sensor in the project (e.g., environmental monitoring, home automation, etc.).
- Consider the advantages and limitations of the chosen sensors and how they impact the overall system.

Hardware Connections:

- Describe the wiring connections between Arduino and each sensor.
- Discuss any special considerations or challenges encountered during the hardware setup.
- Provide insights on how to ensure reliable and secure connections between the components.

Data Acquisition and Processing:

- Explain the process of acquiring data from each sensor using Arduino.
- Discuss any calibration or preprocessing techniques required for accurate sensor readings.
- Address any potential issues or limitations in data acquisition and processing.

Serial Communication Protocol:

- Discuss the choice of serial communication protocol (e.g., UART) and its suitability for the project.
- Explain the configuration and settings used for serial communication between Arduino and NodeMCU.
- Explore any potential bottlenecks or challenges related to serial communication.

Code Implementation:

- Provide a detailed explanation of the code snippets for both Arduino and NodeMCU.
- Discuss any specific libraries or functions used in the code and their significance.
- Highlight any modifications or optimizations that can be made to improve code efficiency or functionality.

Testing and Troubleshooting:

- Discuss the testing process used to verify the sensor readings and data transmission.
- Address common issues and challenges encountered during the testing phase.
- Provide troubleshooting tips and solutions for potential problems that may arise.

Future Improvements and Extensions:

- Discuss possible enhancements or additions to the project, such as integrating more sensors or expanding the functionality.
- Explore other communication protocols or technologies that could be used in future iterations.

- Encourage further research and development in the field of sensor interfacing and communication.

5. Conclusion

In conclusion, this document has provided a comprehensive overview of interfacing multiple sensors (DHT11, Flame sensor, LDR, Ultrasonic, PIR, heartbeat, soil moisture, rain sensor) with Arduino and NodeMCU using serial communication. By following the outlined steps and utilizing the provided code snippets, you can effectively connect these sensors to Arduino, read their data, and transmit it to NodeMCU for further processing.

Throughout this project, we have discussed the individual sensor modules, their connections to Arduino, and the necessary code implementation to retrieve sensor data. Additionally, we have demonstrated the process of serial communication between Arduino and NodeMCU, enabling the seamless transfer of sensor data for subsequent analysis and utilization.

By harnessing the power of this sensor interfacing technique, you can create a wide range of projects that require real-time data acquisition from multiple sensors. Whether it's environmental monitoring, home automation, or health tracking, the ability to interface sensors with Arduino and transmit the data via serial communication to NodeMCU opens up numerous possibilities for innovative applications.

It is important to note that the examples provided in this document serve as a foundation and can be further expanded and customized based on specific project requirements. As technology evolves, new sensor modules may become available, and advancements in microcontrollers can further enhance the capabilities of these systems.

By continuously exploring and experimenting with sensor interfacing techniques, you can stay at the forefront of the Internet of Things (IoT) and embedded systems, enabling you to create intelligent and interconnected devices that improve various aspects of our lives.

6. References

1. Arduino Official Website:
    - Website: https://www.arduino.cc/
    - The official website of Arduino provides comprehensive documentation, tutorials, and examples for working with Arduino boards and sensors.
2. NodeMCU Documentation:
    - Website: https://nodemcu.readthedocs.io/
    - The official NodeMCU documentation offers detailed information about the NodeMCU development board and its functionalities.
3. DHT11 Sensor Library:
    - Library: https://github.com/adafruit/DHT-sensor-library
    - The DHT sensor library by Adafruit provides Arduino code examples and functions specifically for DHT11 and other DHT series sensors.
4. Flame Sensor Tutorial:
    - Tutorial: https://www.instructables.com/Flame-Sensor-Arduino-Interface/
    - This Instructables tutorial demonstrates how to interface a flame sensor with Arduino and includes sample code for flame detection.
5. Interfacing Ultrasonic Sensor:
    - Tutorial: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
    - Random Nerd Tutorials provides a comprehensive guide on interfacing an HC-SR04 ultrasonic sensor with Arduino, including code examples.
6. PIR Sensor Tutorial:
    - Tutorial: https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/