



AISSMS
COLLEGE OF ENGINEERING



ज्ञानम् सकलजनहिताय
Approved by AICTE, New Delhi, Recognized by Govt. of Maharashtra,
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC
(Id.No. PU / PN/ Engg. / 093 (1992)
(Accredited by NAAC with grade A+)

Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Study of Raspberry-Pi, Beagle board, Arduino, and different operating systems for Raspberry Pi/Beagle board/Arduino

Date of Performance:

Date of Submission:

Aim: Study of Raspberry-Pi, Beagle board, Arduino, and different operating systems for Raspberry-Pi/Beagle board/Arduino. Understanding the process of OS installation on Raspberry-Pi/Beagle board/Arduino

Introduction:

Internet of Things: - IoT is short for Internet of Things. The Internet of Things refers to the ever-growing network of physical objects that feature an IP address for internet connectivity, and the communication that occurs between these objects and other Internet-enabled devices and systems. The Internet of Things (IoT) refers to the use of intelligently connected devices and systems to leverage data gathered by embedded sensors and actuators in machines and other physical objects. In other words, the IoT (Internet of Things) can be called to any of the physical objects connected with network.

Examples of IoT: -

- 1) Apple Watch and Home Kit.
- 2) Smart Refrigerator.
- 3) Smart Refrigerator.
- 4) Smart cars.
- 5) Google Glass.
- 6) Smart thermostats.

A) Raspberry-Pi: - The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. It does not include peripherals (such as keyboards and mice). The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. The Raspberry Pi is a credit-card-sized computer that costs between \$5 and \$35. It's available anywhere in the world, and can function as a proper desktop computer or be used to build smart devices. A

Raspberry Pi is a general-purpose computer, usually with a Linux operating system, and the ability to run multiple programs. Raspberry Pi is like the brain. Its primary advantage comes in processing higher level processing capability. It's a single board computer.



Fig.1: - Raspberry-Pi

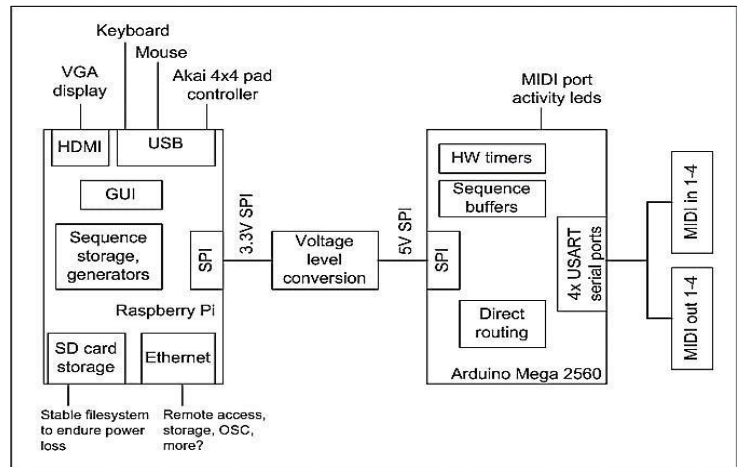


Fig.2: -Raspberry-Pi Architecture

Here are the various components on the Raspberry Pi board:

- **ARM CPU/GPU** -- This is a Broadcom BCM2835 System on a Chip (SoC) that's made up of an ARM central processing unit (CPU) and a Video core 4 graphics processing unit (GPU). The CPU handles all the computations that make a computer work (taking input, doing calculations and producing output), and the GPU handles graphics output.
- **GPIO** -- These are exposed general-purpose input/output connection points that will allow the real hardware hobbyists the opportunity to tinker.
- **RCA** -- An RCA jack allows connection of analog TVs and other similar output devices.
- **Audio out** -- This is a standard 3.55-millimeter jack for connection of audio output devices such as headphones or speakers. There is no audio in.
- **LEDs** -- Light-emitting diodes, for your entire indicator light needs.
- **USB** -- This is a common connection port for peripheral devices of all types (including your mouse and keyboard). Model A has one, and Model B has two. You can use a USB hub to expand the number of ports or plug your mouse into your keyboard if it has its own USB port.
- **HDMI** -- This connector allows you to hook up a high-definition television or other compatible device using an HDMI cable.
- **Power** -- This is a 5v Micro USB power connector into which you can plug your compatible power supply.
- **SD card slot** -- This is a full-sized SD card slot. An SD card with an operating system (OS) installed is required for booting the device. They are available for purchase from the manufacturers, but you can also download an OS and save it to the card yourself if you have a Linux machine and the wherewithal.
- **Ethernet** -- This connector allows for wired network access and is only available on the Model B.

B) Beagle Board: - The **Beagle Board** is a low-power open-source single-board computer produced by Texas Instruments in association with Digi-Key and Newark element14. The Beagle Board was also designed with open-source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip. The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open-source hardware and software capabilities. It is also sold to the public under the Creative Commons share-alike license. The board was designed using Cadence OrCAD for schematics and Cadence Allegro for PCB manufacturing; no simulation software was used. **Beagle Bone Black** is a low-cost, open source, community-supported development platform for ARM® Cortex™-A8 processor developers and hobbyists. Boot Linux in under 10-seconds and get started on Sitara™ AM335x ARM Cortex-A8 processor development in less than 5 minutes with just a single USB cable.

Fig.3: -Beagle Board Black

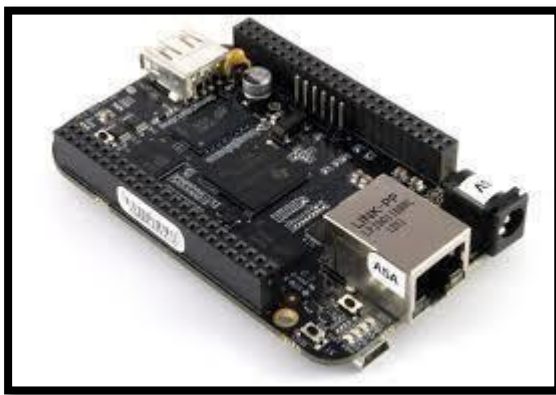
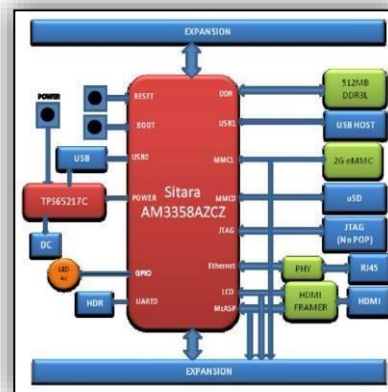


Fig.4 : - Beagle Board Black architecture



Here are the various components on the Beagle board:

Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- 3D graphics accelerator
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

Connectivity

- USB client for power & communications
- USB host
- Ethernet
- HDMI
- 2x 46 pin headers

Software Compatibility

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ Bone Script library

C) Arduino: - **Arduino** is an open-source hardware and software company, project and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical and digital world. Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler tool chains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project. Arduino is open-source hardware. The hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino website. Layout and production files for some versions of the hardware are also available

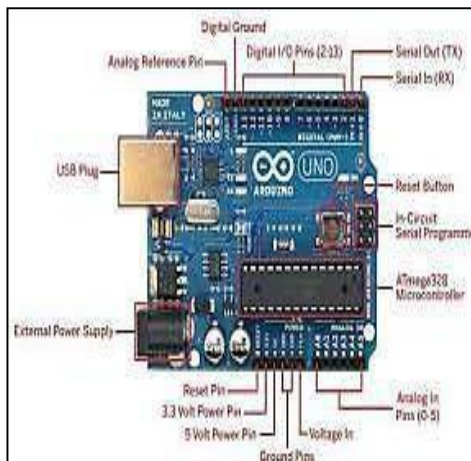


Fig.5: - Arduino Board

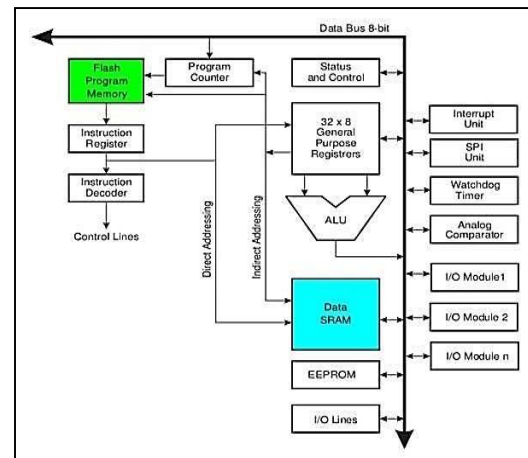


Fig.6: - Arduino Board Architecture

Here are the various components on the Arduino board:

Microcontrollers

ATmega328P (used on most recent boards)

ATmega168 (used on most Arduino Diecimila and early Duemilanove) ATmega8 (used on some older board)

Digital Pins

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the `pinMode()`, `digitalRead()`, and `digitalWrite()` commands. Each pin has an internal pull-up resistor which can be turned on and off using `digitalWrite()` (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input. The maximum current per pin is 40 mA.

Analog Pins

In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the `analogRead()` function. Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.

Power Pins

VIN (sometimes labelled "9V"). The input voltage to the Arduino board when it's 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 powerpack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the Lilypad has no VIN pin and accepts only a regulated input.

Other Pins

- **AREF.** Reference voltage for the analog inputs. Used with `analogReference()`.
- **Reset.** (Diecimila-only) Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- Analog Reference pin (orange)
- Digital Ground (light green)
- Digital Pins 2-13 (green)
- Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital i/o (**digitalRead** and **digitalWrite**) if you are also using serial communication (e.g., **Serial.begin**).
- Reset Button - S1 (dark blue)
- In-circuit Serial Programmer (blue-green)
- Analog In Pins 0-5 (light blue)
- Power and Ground Pins (power: orange, grounds: light orange)
- External Power Supply In (9-12VDC) - X1 (pink)
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board)

Conclusion:



Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Study of different operating systems for Raspberry-Pi/Beagle board/Arduino. Understanding the process of OS installation on Raspberry-Pi/Beagle board/Arduino.

Date of Performance:

Date of Submission:

Aim: To study operating systems for platforms such as Raspberry-Pi/Beagle board/Arduino

Introduction:

Theory:

- 1) **Raspberry-Pi:** - The Pi can run the official Raspbian OS, Ubuntu Mate, Snappy Ubuntu Core, the Kodi- based media centers OSMC and LibreElec, the non-Linux based Risc OS (one for fans of 1990s Acorn computers). It can also run Windows 10 IoT Core, which is very different to the desktop version of Windows, as mentioned below.
- **OS which installs on Raspberry-Pi:** Raspbian, Ubuntu MATE, Snappy Ubuntu, Pidora, Linutop, SARPi, Arch Linux ARM, Gentoo Linux, etc.

➤ How to install Raspbian on Raspberry-Pi: Step 1:

Download Raspbian

Step 2: Unzip the file. The Raspbian disc image is compressed, so you'll need to unzip it. The file uses the ZIP64 format, so depending on how current your built-in utilities are, you need to use certain programs to unzip it.

Step 3: Write the disc image to your microSD card. Next, pop your microSD card into your computer and write the disc image to it. The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what you're using. Each of these program will have you select the destination (make sure you've picked your microSD card!) and the disc image (the unzipped Raspbian file). Choose, double-check, and then hit the button to write.

Step 4: Put the microSD card in your Pi and boot up. Once the disc image has been written to the microSD card, you're ready to go! Put that sucker into your Raspberry Pi, plug in the peripherals and power source, and enjoy. The current edition to Raspbian will boot directly to the desktop. Your default credentials are username **pi** and password **raspberry**.

- 2) **BeagleBone Black:** - The **BeagleBone Black** includes a 2GB or 4GB on-board eMMC flash memory chip. It comes with the Debian distribution factory pre-installed. You can flash new operating systems including Angstrom, Ubuntu, Android, and others.

Os which installs on BeagleBone Black: Angstrom, Android, Debian, Fedora, Buildroot, Gentoo, Nerves Erlang/OTP, Sabayon, Ubuntu, Yocto, MINIX 3

➤ **How to install Debian on BeagleBone Black:Step 1:**

Download Debian img.xz file.

Step 2: Unzip the file.

Step 3: Insert your MicroSD (uSD) card into the proper slot. Most uSD cards come with a full-sized SD card that is really just an adapter. If this is what you have then insert the uSD into the adapter, then into your card reader.

Step 4: Now open Win32 Disk imager, click the blue folder icon, navigate to the debian img location, and double click the file. Now click Write and let the process complete. Depending on your processor and available RAM it should be done in around 5 minutes.

Step 5: Alright, once that's done, you'll get a notification pop-up. Now we're ready to get going. Remove the SD adapter from the card slot, remove the uSD card from the adapter. With the USB cable disconnected insert the uSD into the BBB.

Step 6: Now, this next part is pretty straight forward. Plug the USB cable in and wait some more. If everything is going right you will notice that the four (4) leds just above the USB cable are doing the KIT impression. This could take up to 45 minutes, I just did it again in around 5 minutes. Your mileage willvary. Go back and surf reddit some more.

Step 7: If you are not seeing the leds swing back and forth you will need to unplug the USB cable, press and hold down the user button above the uSD card slot (next to the 2 little 10 pin ICs) then plug in the USB cable. Release the button and wait. You should see the LEDs swinging back and forth after a few seconds. Once this happens it's waiting time. When all 4 LEDs next to the USB slot stay lit at the same time the flash process has been completed.

Step 8: Remove the uSD card and reboot your BBB. You can reboot the BBB by removing and reconnecting the USB cable, or hitting the reset button above the USB cable near the edge of the board.

Step 9: Now using putty, or your SSH flavor of choice, connect to the BBB using the IP address 192.168.7.2. You'll be prompted for a username. Type root and press Enter. By default, there is no root password. I recommend changing this ASAP if you plan on putting your BBB on the network. To do this type password, hit enter, then enter your desired password. You will be prompted to enter it again to verify.

- 3) **Arduino:** - The Arduino itself has no real operating system. You develop code for the Arduino using the Arduino IDE which you can download from Arduino - Home. Versions are available for **Windows, Mac** and **Linux**. The Arduino is a constrained microcontroller.

Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, **used** to write and upload computer code to the physical board. You are literally writing the "firmware" when you write the code and upload it. It's both good and its bad.

Conclusion:



Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Simple program digital read/write using LED and Switch -Analog read/write using sensor

Date of Performance:

Date of Submission:

Aim: Interfacing LED with Arduino

Hardware Requirements: Arduino Board, LED, Push Button

Software Requirements: Arduino IDE

Introduction:

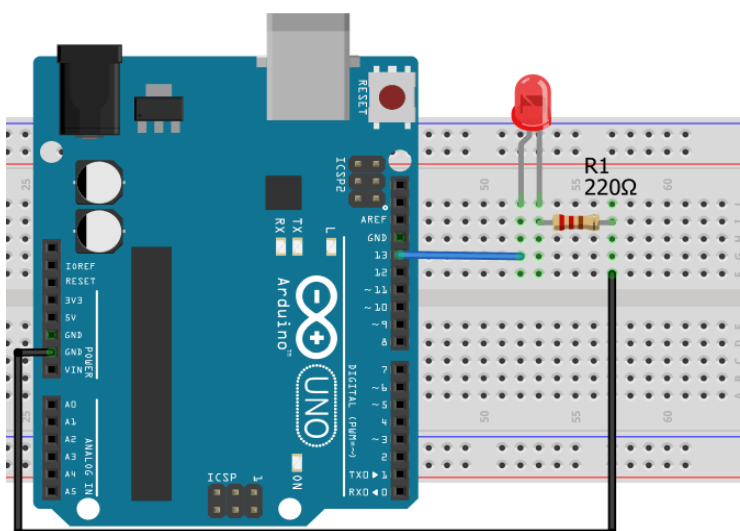


Fig 1 Interfacing LED with Arduino

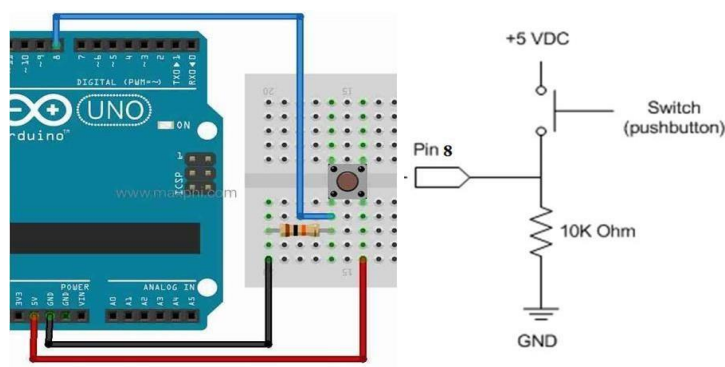


Fig 2 Interfacing LED with Push Button to Aurduino

Procedure:

1. Make the connection as per diagram
2. Open Arduino IDE
3. Select the Board and the Port
4. Write a code in the editor window
5. Compile and upload the code

Code:**A) LED with Arduino:**

```
#define LED_BUILTIN=4;

void setup()
{
    pinMode(LED_BUILTIN, OUTPUT); // initialize digital pin LED_BUILTIN as an output.
}

void loop()
{
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000);                      // wait for a second
    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
    delay(1000);                      // wait for a second
}
```

B) LED with Push Button to Arduino:

```
const int buttonPin = 8;    // Digital Pin 8 define as a push Button
const int ledPin = 10; // Digital Pin 13 define as LED Pin
int buttonState = 0; // variable for reading the pushbutton status

void setup()
{
    pinMode(ledPin , OUTPUT);
    pinMode(buttonPin , INPUT);
    Serial.begin(9600);
```

```

}

void loop()
{
  buttonState = digitalRead(buttonPin);

  if (buttonState == HIGH)
  {
    digitalWrite(ledPin , HIGH);

    Serial.println("Button Pressed");
  }

  else
  {
    digitalWrite(ledPin , LOW);

    Serial.println("Button Not Pressed");
  }
}

```

Results:

A) LED with Arduino:

```

LED | Arduino 1.8.16
File Edit Sketch Tools Help

LED

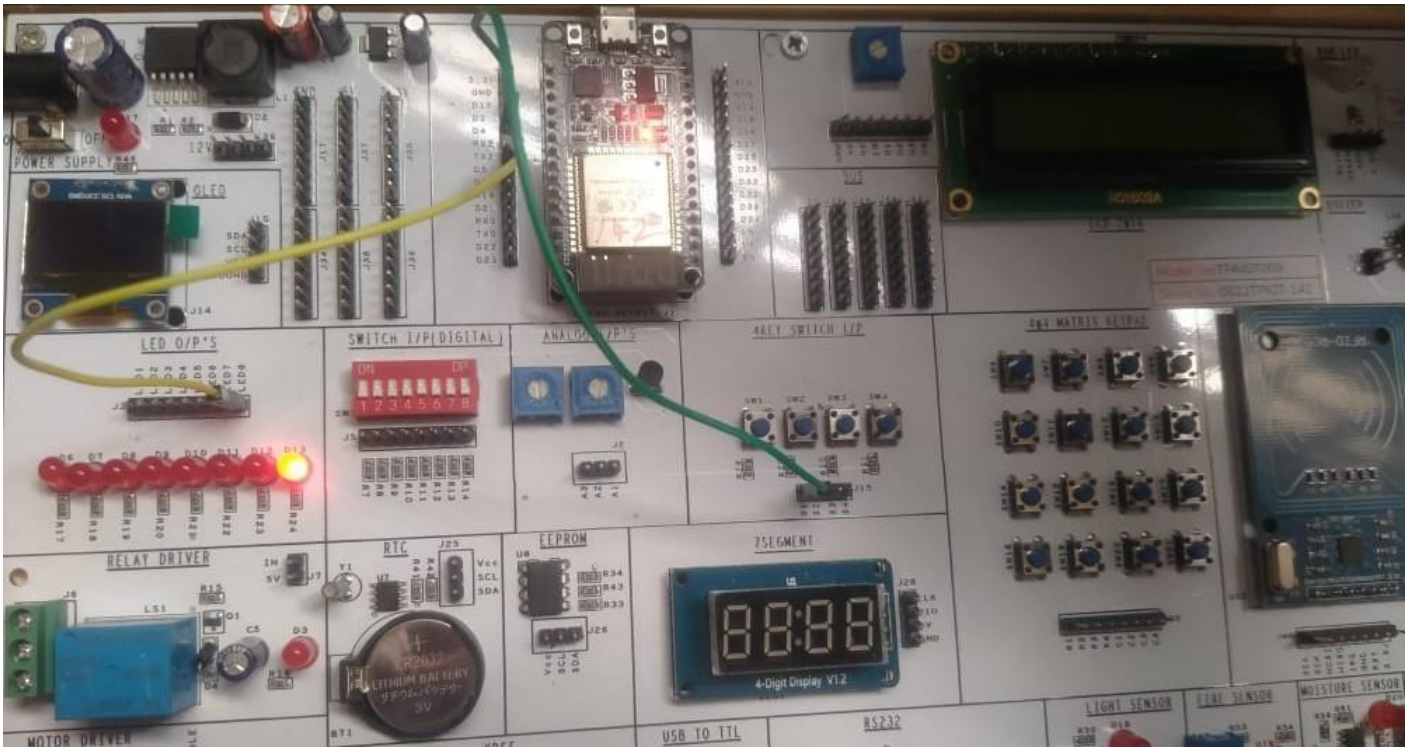
int LED_PIN=4;
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_PIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_PIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);                // wait for a second
  digitalWrite(LED_PIN, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);                // wait for a second
}

Done uploading.
Sketch uses 936 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

11 Arduino Uno on COM4

```

Conclusion:



Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Interfacing sensors and actuators with Arduino.

Date of Performance:

Date of Submission:

Aim: To interface Ultrasonic sensor and DHT11 - temperature humidity sensor with Arduino or ESP866 or ESP 32

Hardware Requirements: Arduino Board, Ultrasonic Sensor

Software Requirements: Arduino IDE

Theory:

A) DHT11 Sensor

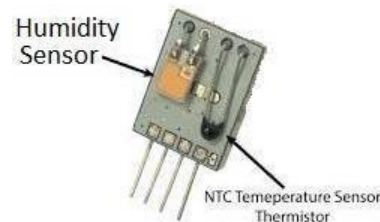
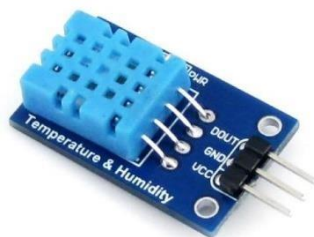


Fig. 1 DHT11 Sensor

The DHT11 is a basic, ultra- low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and gives out a digital signal on the data pin (no analog input pins needed).

Specifications

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50°C temperature readings $\pm 2^\circ\text{C}$ accuracy
- Body size 15.5mm x 12mm x 5.5mm
- 4 pins with 0.1" spacing

The DHT11 measures *relative humidity*. The relative humidity is the amount of water vapor in air vs. the saturation point of water vapor in the air. At the saturation point, water vapor starts

to condense and accumulate on surfaces forming dew. The saturation point changes with air temperature. Cold air can hold less water vapor before it becomes saturated, and hot air can hold more water vapor before it becomes saturated.

The formula to calculate relative humidity is:

$$RH = \left(\frac{\rho_w}{\rho_s} \right) \times 100\%$$

RH : Relative Humidity

ρ_w : Density of water vapor

ρ_s : Density of water vapor at saturation

The relative humidity is expressed as a percentage. At 100% RH, condensation occurs, and at 0% RH, the air is completely dry.

No:	Pin Name	Description
Four pin Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
Three pin sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

B) Ultrasonic Sensor



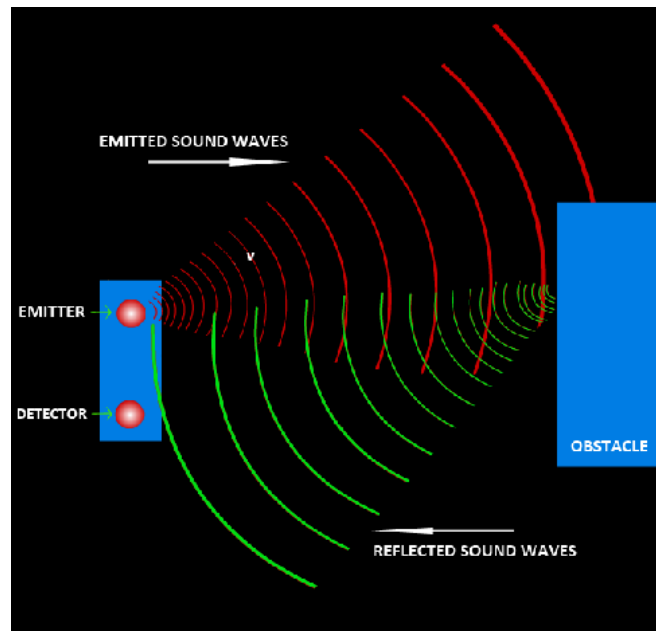
Fig. 2 DHT11 Sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e., the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is $D = \frac{1}{2} T \times C$ (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second). For example, if a scientist set up an ultrasonic sensor aimed at a box and it took 0.025 seconds for the sound to bounce back, the distance between the ultrasonic sensor and the box would be:

$$D = 0.5 \times 0.025 \times 343$$

or about 4.2875 meters.



Ultrasonic sensors are also used as level sensors to detect, monitor, and regulate liquid levels in closed containers (such as vats in chemical factories).

Circuit Connection:

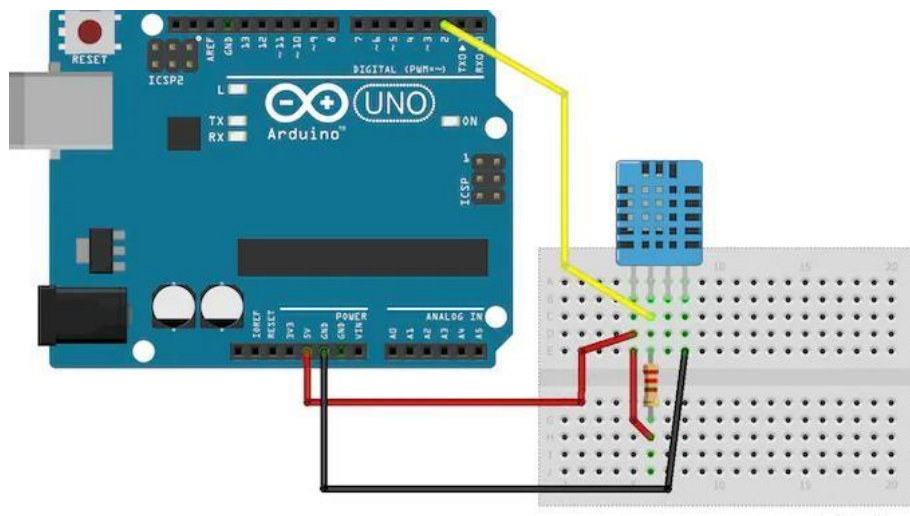


Fig. 3 Interfacing of DHT11 with Aurdunio

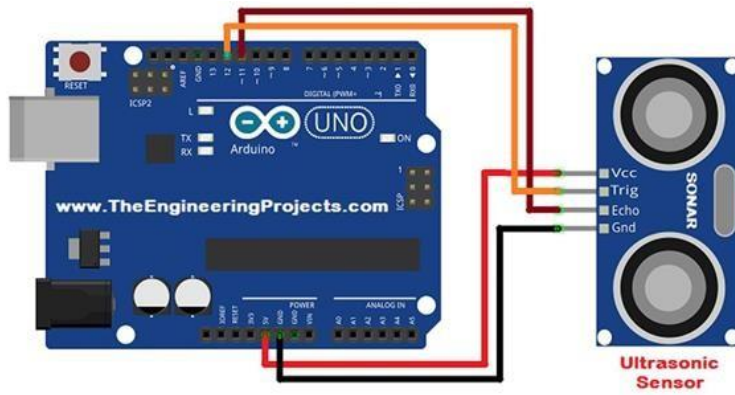


Fig. 4 Interfacing of Ultrasonic with Arduino

Code:

A) Interfacing DHT11 with Arduino:

```
#include "DHT.h"

#define DHTPIN 2    // Digital pin connected to the DHT sensor

#define DHTTYPE DHT11  // DHT 11

DHT dht11(DHTPIN, DHTTYPE);

void setup()
{
    Serial.begin(9600);
    dht11.begin();
}

void loop()
{
    delay(2000); // Reading temperature or humidity takes about 250 milliseconds!
    float h = dht11.readHumidity(); // Read temperature as Celsius (the default)
    float t = dht11.readTemperature(); // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t))
    {
        Serial.println("Failed to read from DHT sensor!");
    }
}
```

```

    return;
}

Serial.println("Humidity: ");

Serial.println(h);

Serial.println("Temperature: ");

Serial.println(t);
}

```

B) Interfacing Ultrasonic Sensor with Arduino:

```

#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04

#define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04

// defines variables

long duration; // variable for the duration of sound wave travel

int distance; // variable for the distance measurement

void setup()

{

    pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT

    pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

    Serial.begin(9600); // Serial Communication is starting with 9600 of baudrate speed
}

void loop()

{

    // Clears the trigPin condition

    digitalWrite(trigPin, LOW);

    delayMicroseconds(2); // Sets the trigPin HIGH (ACTIVE) for 10 microseconds

    digitalWrite(trigPin, HIGH);

    delayMicroseconds(10);

    digitalWrite(trigPin, LOW);

```

```

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)

// Displays the distance on the Serial Monitor

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

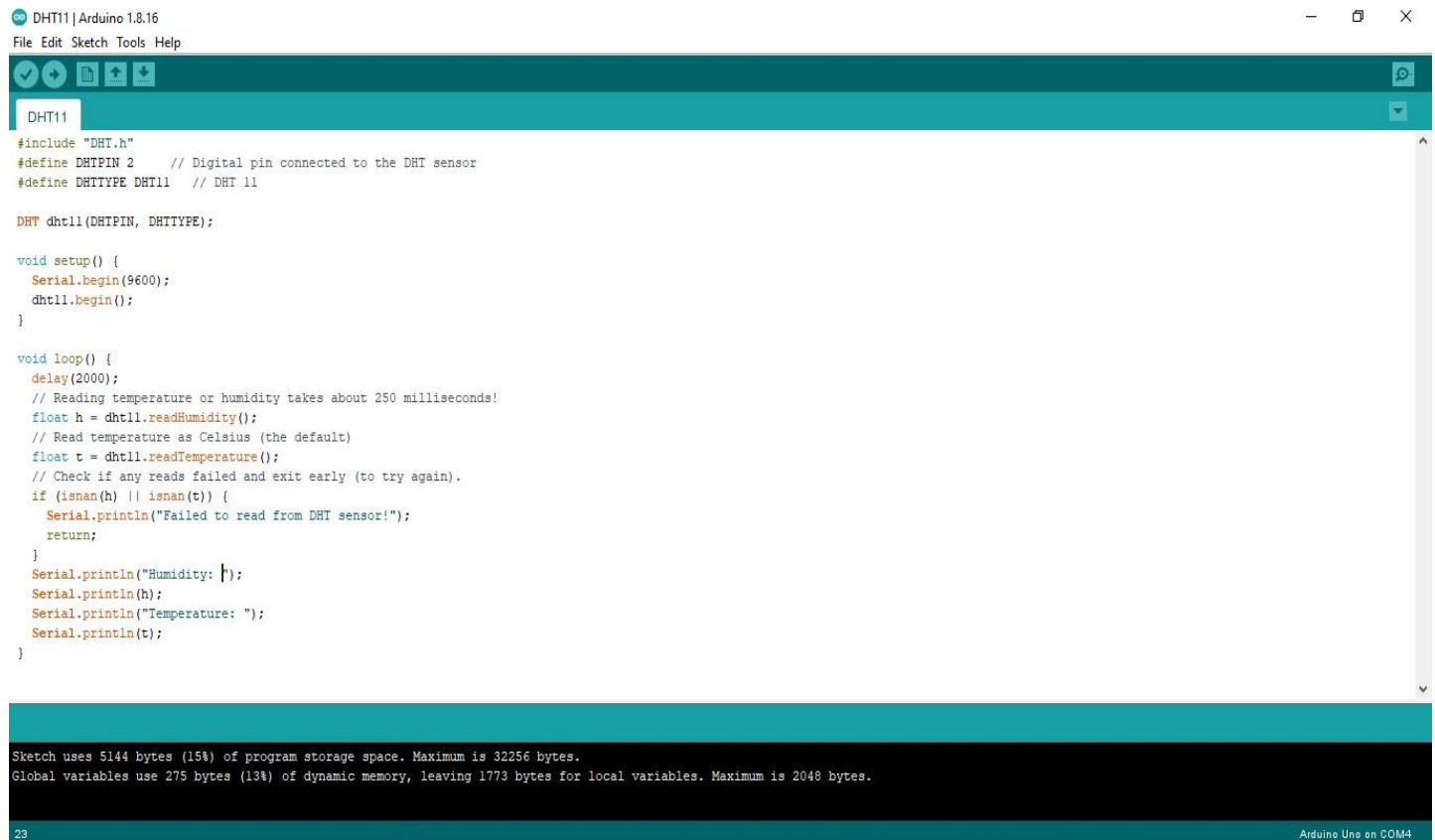
delay(1000);

}

```

Results:

A) Interfacing DHT11 with Arduino:



```

DHT11 | Arduino 1.8.16
File Edit Sketch Tools Help

DHT11

#include "DHT.h"
#define DHTPIN 2 // Digital pin connected to the DHT sensor
#define DHTTYPE DHT11 // DHT 11

DHT dht11(DHTPIN, DHTTYPE);

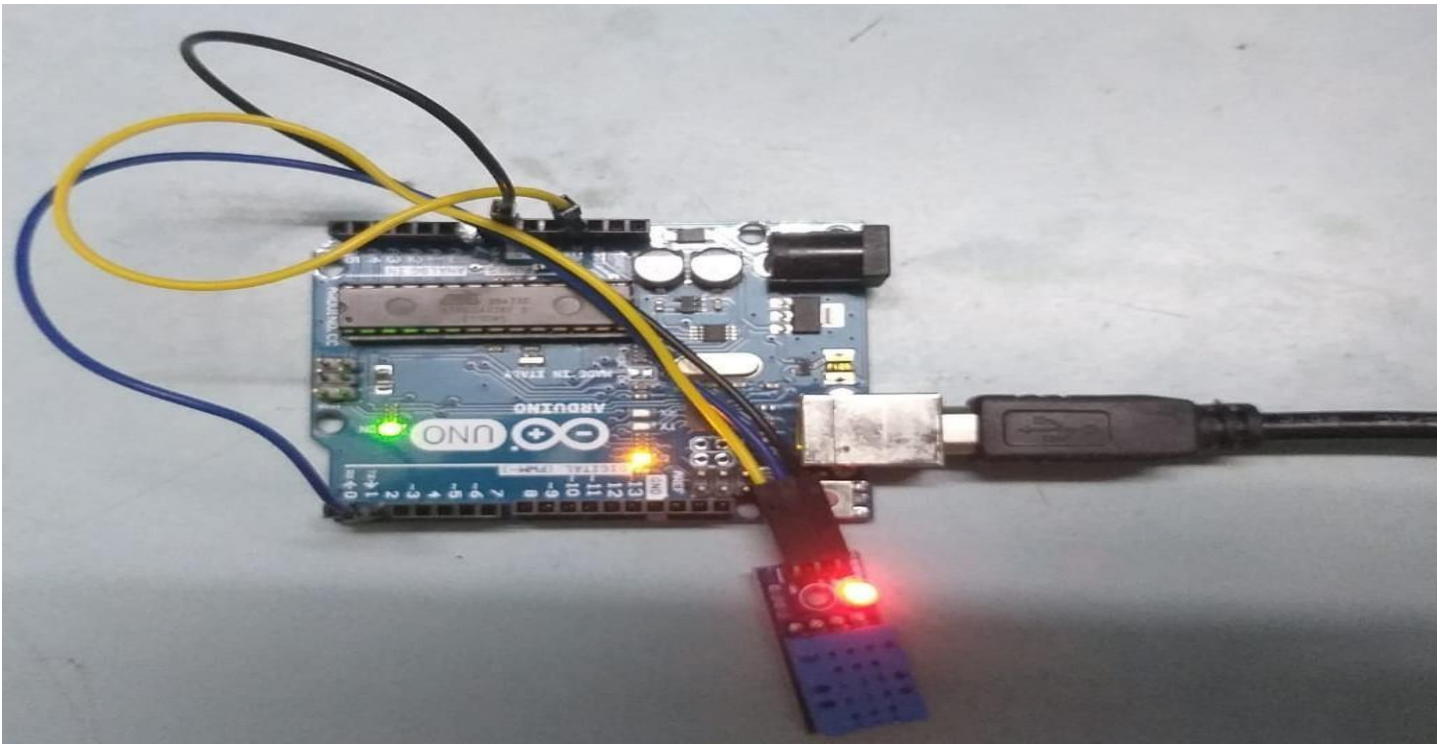
void setup() {
  Serial.begin(9600);
  dht11.begin();
}

void loop() {
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  float h = dht11.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht11.readTemperature();
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.println("Humidity: ");
  Serial.println(h);
  Serial.println("Temperature: ");
  Serial.println(t);
}

Sketch uses 5144 bytes (15%) of program storage space. Maximum is 32256 bytes.
Global variables use 275 bytes (13%) of dynamic memory, leaving 1773 bytes for local variables. Maximum is 2048 bytes.

23 Arduino Uno on COM4

```



B) Interfacing Ultrasonic Sensor with Arduino:

Ultrasonic_Sensor | Arduino 1.8.16

File Edit Sketch Tools Help

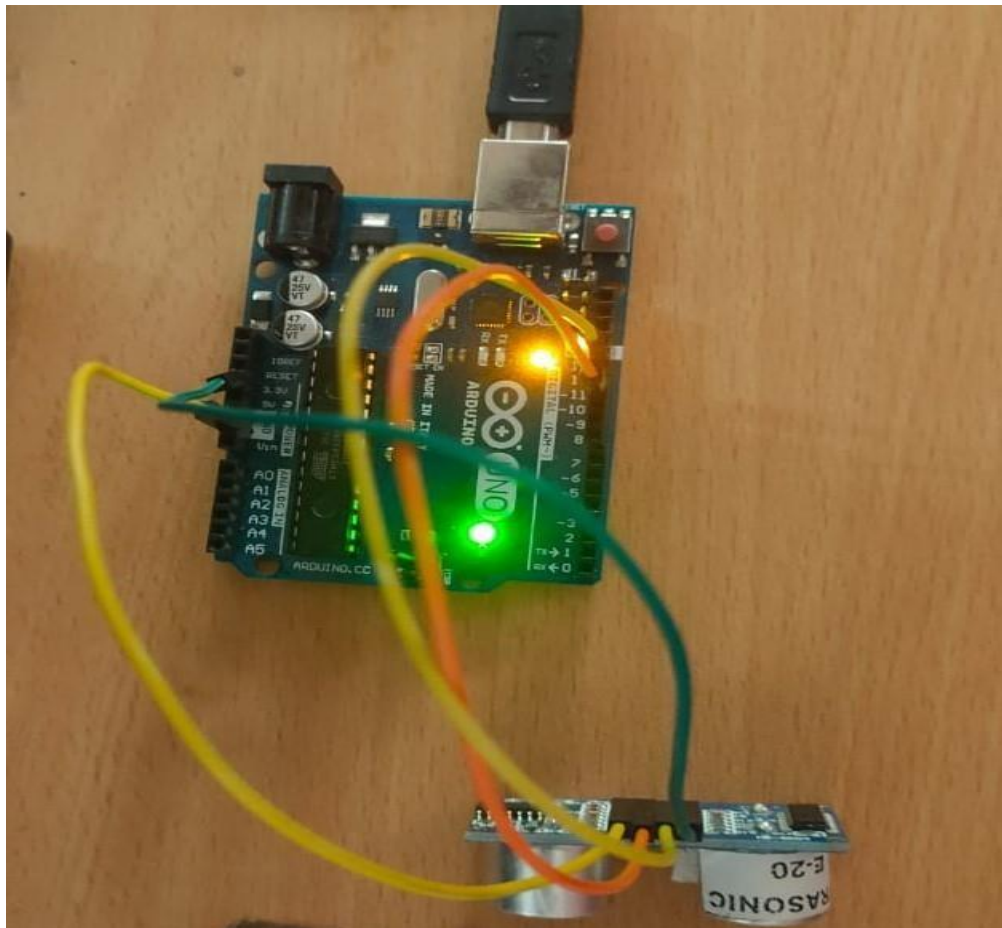
Ultrasonic_Sensor

```
#define echoPin 11 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 12 //attach pin D3 Arduino to pin Trig of HC-SR04
// defines variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
}

void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  delay(1000);
}
```

Sketch uses 3136 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 204 bytes (9%) of dynamic memory, leaving 1844 bytes for local variables. Maximum is 2048 bytes.



Conclusion:



Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: IoT based DC Motor Control with Arduino.

Date of Performance:

Date of Submission:

Aim: To Interface DC Motor Arduino or ESP866 or ESP 32

Hardware Requirements: Arduino Board and DC motor

Software Requirements: Arduino IDE

Theory:

A DC motor (Direct Current motor) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.



Fig. 1 DC Motor

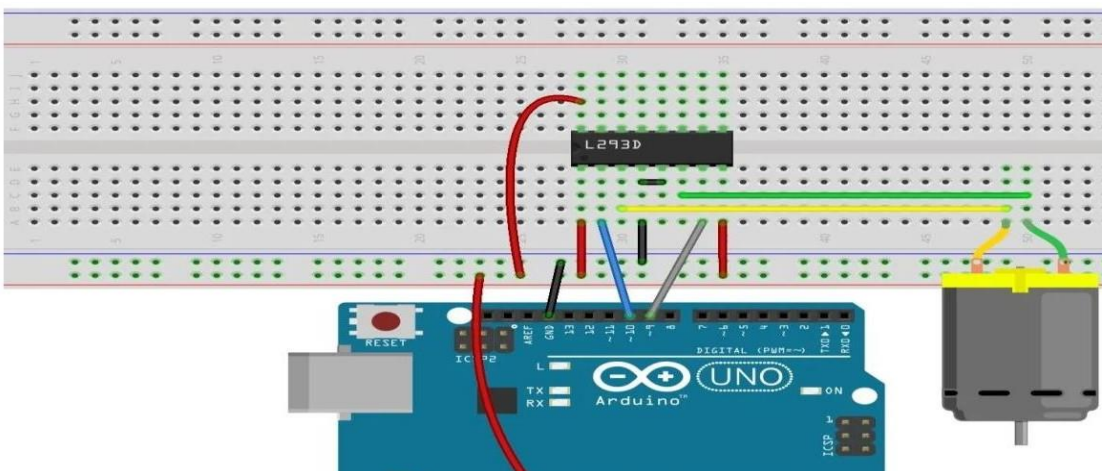


Fig. 2 Interfacing of DC Motor with Arduino

The maximum current of an Arduino I/O port is 20mA but the drive current of a motor is at least 70mA. Therefore, we cannot directly use the I/O port to drive the current; instead, we can use an L293D to drive the motor. L293D is designed to provide bidirectional drive currents of up to 600mA at voltages from 4.5V to 36V. It's used to drive inductive loads such as relays, solenoids, DC and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

Code:

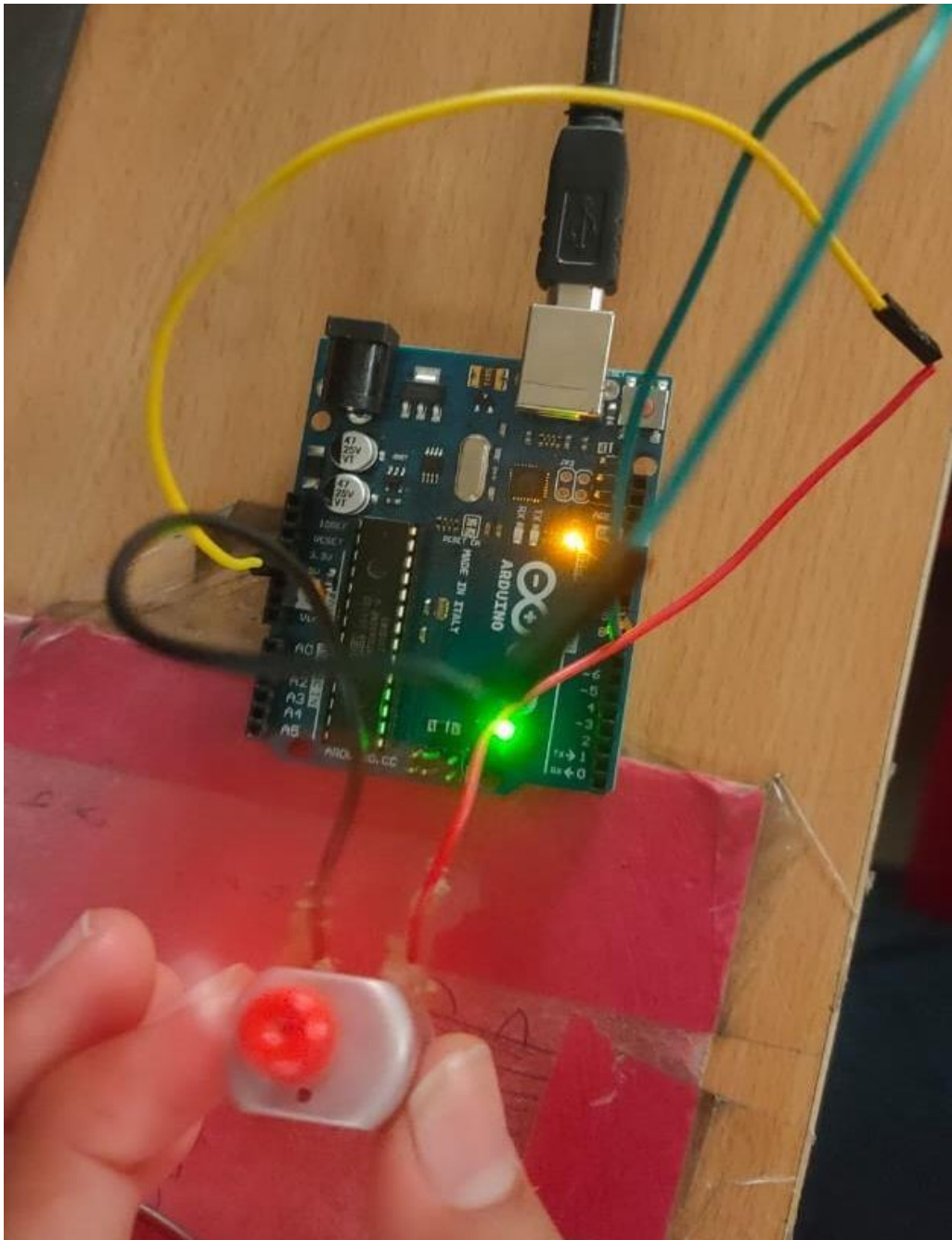
```
int motorPin = 9;

void setup()
{
    pinMode(motorPin, OUTPUT);
    Serial.begin(9600);
}

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

Results:





Conclusion:



Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Introduction to MQTT/ CoAP and sending sensor data to cloud using Raspberry-Pi/Beagle board/Arduino.

Date of Performance:

Date of Submission:

Aim: Interfacing of Potentiometer with Arduino or ESP866 or ESP 32 and store the information on ThingSpeak cloud

Hardware Requirements: Arduino Bord, Potentiometer

Software Requirements: Arduino IDE, ThingSpeak Cloud

Theory:

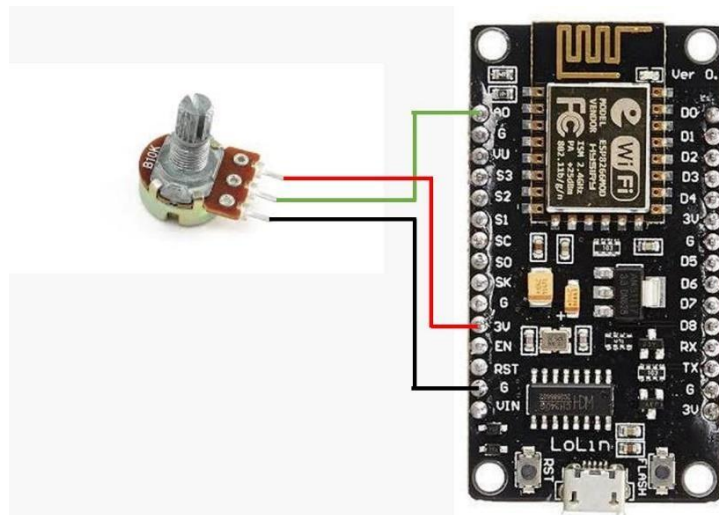


Fig. 1 Interfacing of Potentiometer with ESP8266

Procedure:

- 1, Create a channel on the cloud (ThingSpeak).
- 2, Make the connection as per circuit diagram.
3. Open Arduino IDE & Write the code in editor window.

4. Compile and upload the code.
5. Open ThingSpeak and observe the results

Code:

```
#include <ESP8266WiFi.h>

#include "ThingSpeak.h"

const char* ssid = "network name"; // your network SSID (name)

const char* password = "*****"; // your network password

WiFiClient client;

unsigned long myChannelNumber = 1;

const char * myWriteAPIKey = "SSMJK7SAJCQDNLA7";

unsigned long lastTime = 0; // Timer variables

unsigned long timerDelay = 30000;

float sensorReading; // Variable to hold temperature readings

// Create a sensor object

void setup()

{

    Serial.begin(115200); //Initialize serial

    pinMode(A0,INPUT);

    WiFi.mode(WIFI_STA);

    WiFi.begin(ssid, password);

    delay(5000);

    ThingSpeak.begin(client); // Initialize ThingSpeak

}

void loop()

{

    sensorReading = analogRead(A0); // Get a new temperature reading

    Serial.print("Reading: ");

    Serial.println(sensorReading);
```

```

int x = ThingSpeak.writeField(myChannelNumber, 1, temperatureC, myWriteAPIKey);

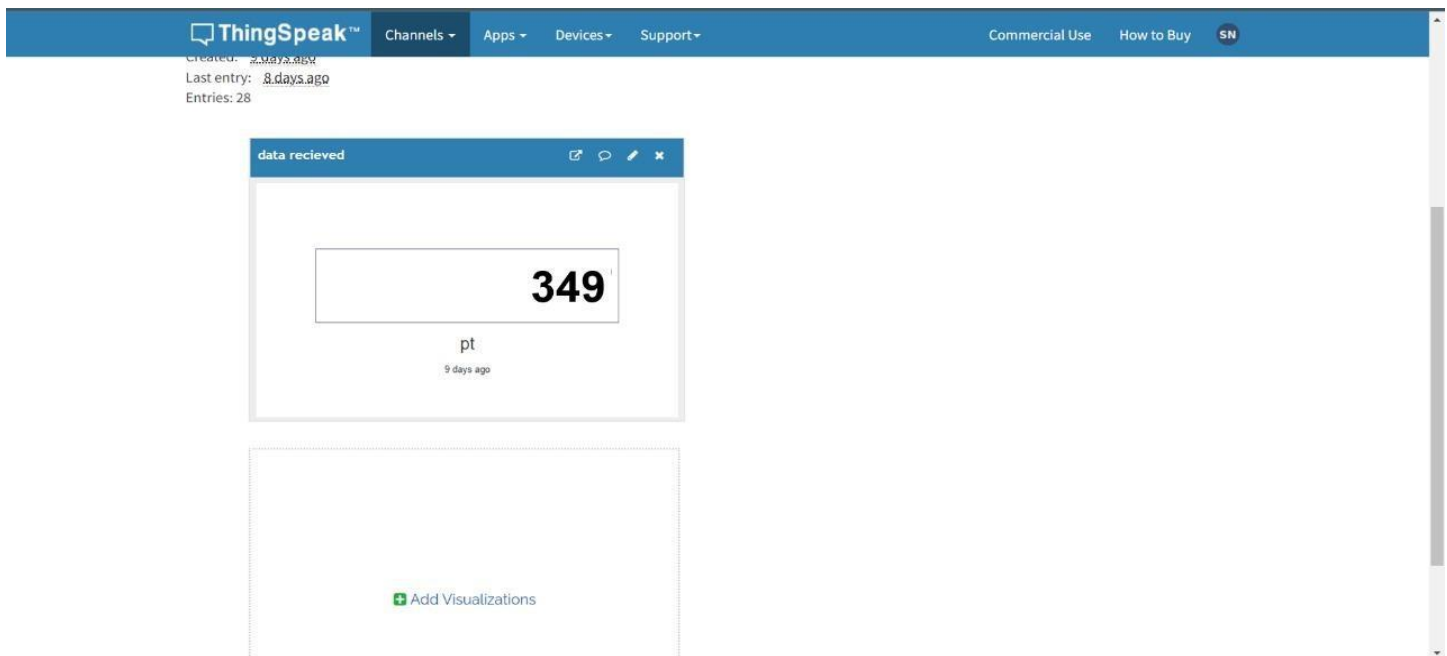
if(x == 200)
{
    Serial.println("Channel update successful.");
}

Else
{
    Serial.println("Problem updating channel. HTTP error code " + String(x));
}

lastTime = millis();
}
}

```

Results:



Conclusion:



Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Get the status of a bulb at a remote place (on the LAN) through web.

Date of Performance:

Date of Submission:

Aim: Get the status of a bulb at a remote place (on the LAN) through web.

Hardware Requirements: Arduino Board, PHPoC Shield 2 for Arduino, 1 PHPoC 4-Port Relay Board for Arduino.

Software Requirements: Arduino IDE

Theory:



Fig. 1 Interfacing of Bulb with Aurdunio

Procedure

1. First test the PHPoC Shield.
2. Connect to the shield by entering "192.168.0.1".
3. upload php code to PHPoC Shield 2 for Arduino, install PHPoC Debugger on a PC.

4. Open PHPoC Debugger. Select connected COM PORT (USB) and press connect button. If USB is successfully connected, "Connect" button will be inactivated and "Disconnect" button will be activated. Drag and drop PHP file to file list. Then hit "Upload" button.
5. To check if php code was installed properly, kill power & disconnect. Then re-power and re-connect PHPoC board to PC. The Debugger should now list the new file and an updated flash memory bar to show how much memory is used.
6. To set up Arduino, upload "remoted_ino.ino" to Arduino using the Arduino Web Editor
7. Stack (from the bottom up) the Arduino, the PHPoC Arduino Shield, and the PHPoC 4-Port Relay Board for Arduino.
8. PHPoC 4-Port Relay Board uses jumpers to select whether the relays are NO (Normally Open) or NC (Normally Closed). If jumpers are not present, the relay will activate but there won't be any connection on the terminal strip. The jumpers (not supplied) determine how the terminals are connected with relay activation. In the "NO" position, when the Light is commanded to be on, a connection will be made between the two terminals marked "0" on the PHPoC 4-Port Relay Board.
9. Wire up LED
Make sure the DIP switches on the PHPoC 4-Port Relay Board are set to 0,0,0,1. (These can be easily bumped during shipment or when stacking the boards.) Relays and Data out terminals are marked 0-3 - each one has two corresponding screw terminals on the terminal strip. Finally, connect the LED you want to turn on and off to a power supply in series with the two screws on the terminal strip corresponding to Relay 0.
10. To test, power up Arduino stack. Go to a computer or smart phone. Find Wi-Fi Network "phpoc_13ba5e". Connect.
11. In web browser go to URL: http://192.168.0.1/remote_led_php.php
12. Webpage should appear, with a button ("Switch") that can toggle remote light switch on and off.

Code:

```
#include <Phpoc.h>
#include <PhpocExpansion.h>
#define ON '1'
#define OFF '0'
byte expansionId = 1;
ExpansionRelayOutput relay(expansionId, 0);
PhpocServer server(80);
void setup()
{
    Serial.begin(9600);
    while(!Serial); // initialize PHPoC [WiFi] Shield:
    Phpoc.begin(PF_LOG_SPI | PF_LOG_NET); //Phpoc.begin();
    server.beginWebSocket("remote_led"); // start WebSocket server
    // print IP address of PHPoC [WiFi] Shield to serial monitor:
    Serial.print("WebSocket server address : ");
    Serial.println(Phpoc.localIP());
```

```

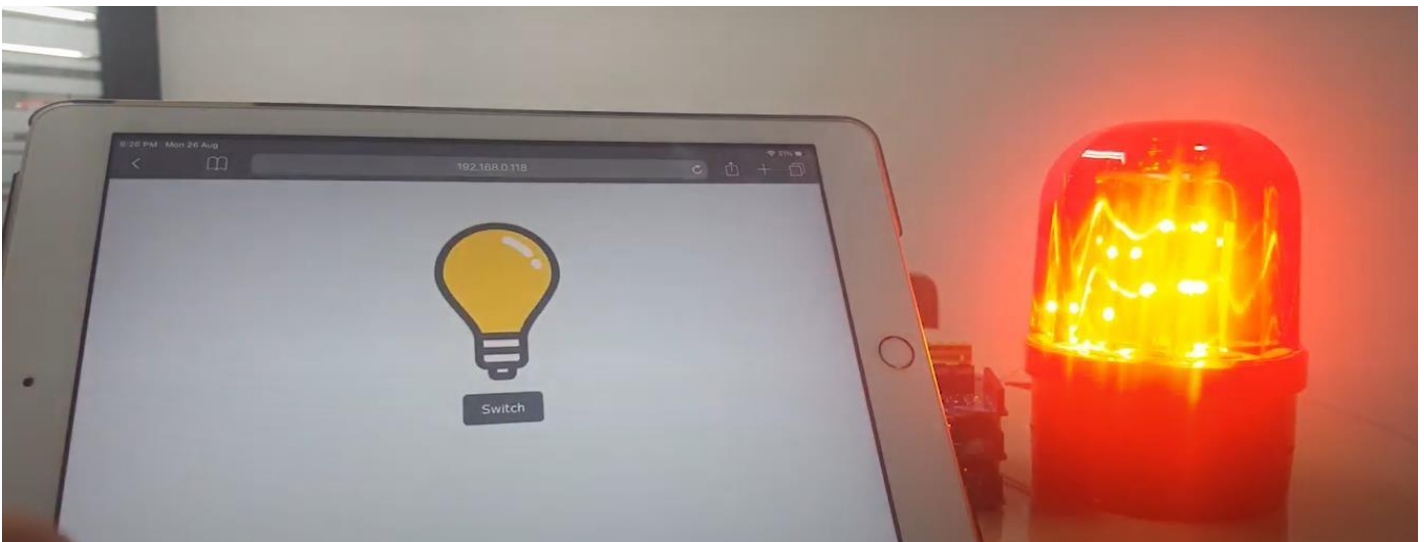
Expansion.begin();
// get name and print it to serial
Serial.println(relay.getName());
}
void loop()
{
  PhpocClient client = server.available(); // wait for a new client:

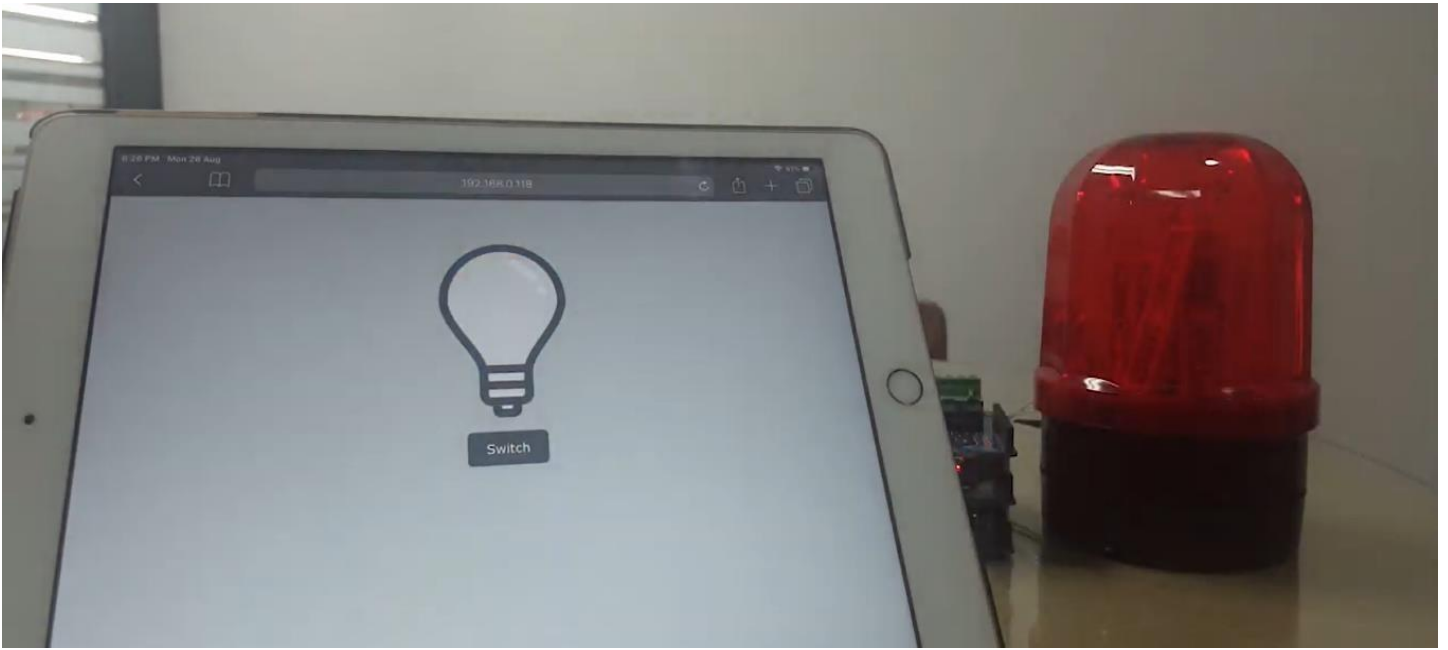
  if (client)
  {
    if (client.available() > 0)
    {
      char thisChar = client.read(); // read a byte incoming from the client:

      switch(thisChar)
      {
        case OFF:
          Serial.println("Turn LED OFF");
          relay.off();
          server.write(OFF);
          break;
        case ON:
          Serial.println("Turn LED ON");
          relay.on();
          server.write(ON);
          break;
      }
    }
  }
}

```

Results:





Conclusion:



AISSMS
COLLEGE OF ENGINEERING



ज्ञानम् सकलजनहिताय
Approved by AICTE, New Delhi, Recognized by Govt. of Maharashtra,
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC
(Id.No. PU / PN/ Engg. / 093 (1992))
(Accredited by NAAC with grade A+)

Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Interfacing Arduino to Bluetooth Module

Date of Performance:

Date of Submission:

Aim: To Interface Arduino to Bluetooth Module

Hardware Requirements: Arduino Board, Bluetooth module HC-05

Software Requirements: Arduino IDE

Theory:

HC-05 Module -

The HC-05 offers many more features such as low power data transmission, high-speed data transfer and many more. It is a full-duplex communication which means both sender and receiver transmit and receive data at a time. Besides, the HC-05 FHSS uses frequency-hopping spread spectrum radio technology. This technique provides many advantages of communication over a noisy channel and prevents the signal from being hacked or jammed.

The HC-05 Bluetooth module uses UART communication to transfer data to the master and uses two pins Rx and Tx. The pinout details of the HC-05 device are as follows:

1. **VCC and GND** - These pins are used to power the module
2. **Rx and Tx** - These pins are used for communication purpose
3. **State** - This pin tells us whether the module is paired or not
4. **EN** - This pin is an active low pin and it defines the mode of operation of HC-05.

HC-05 has two modes of operation those are as follows:

1. Data Mode

By default, HC-05 mode works in data mode. In this mode, HC-05 sends and receives data from other devices.

2. Command Mode

In this mode, HC-05 accepts AT commands from the user and reacts to the commands accordingly.

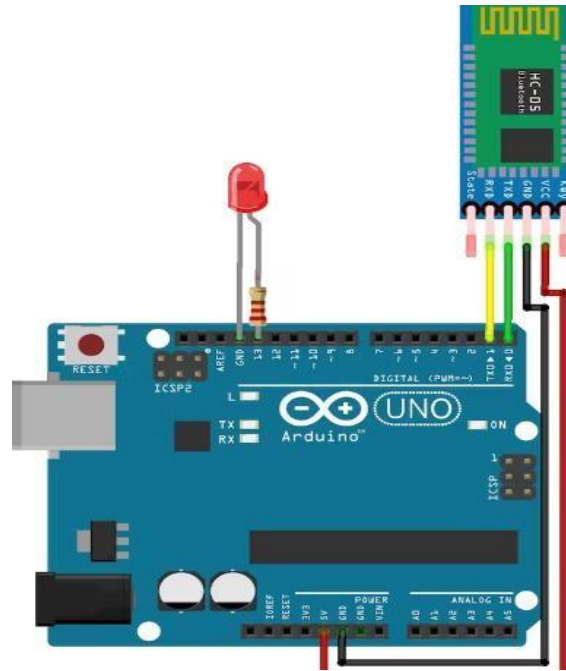


Fig. 1 Interfacing Bluetooth Module with Arduino

Code:

```
char inputByte;

void setup()
{
  Serial.begin(9600);
  pinMode(13,OUTPUT);
}

void loop()
{
  while(Serial.available()>0)
  {
    inputByte= Serial.read();
    Serial.println(inputByte);
    if (inputByte=='Z')
    {
      digitalWrite(13,HIGH);
    }
  }
}
```

```

else if (inputByte=='z')
{
digitalWrite(13,LOW);
}
}
}
}

```

Results:

bluetooth_arduino | Arduino 1.8.16

File Edit Sketch Tools Help

```

bluetooth_arduino
char inputByte;
void setup()
{
  Serial.begin(9600);
  pinMode(13,OUTPUT);
}

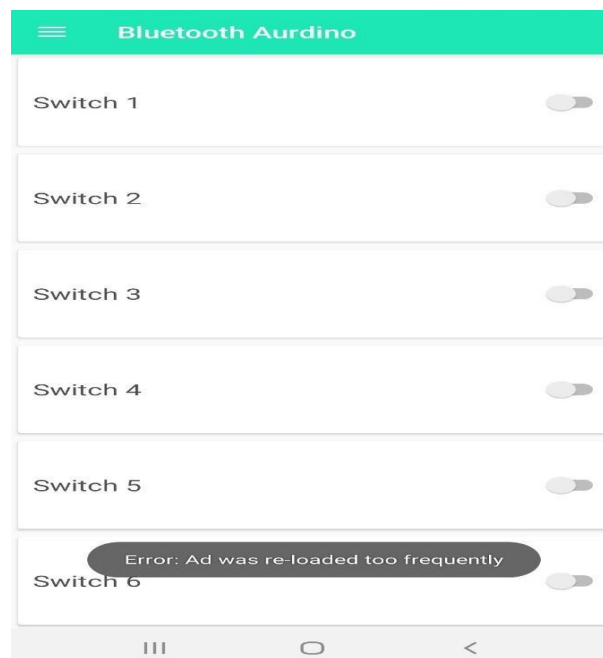
void loop()
{
  while(Serial.available()>0)
  {
    inputByte= Serial.read();
    Serial.println(inputByte);
    if (inputByte=='z')
    {
      digitalWrite(13,HIGH);
    }
    else if (inputByte=='z')
    {
      digitalWrite(13,LOW);
    }
  }
}
}

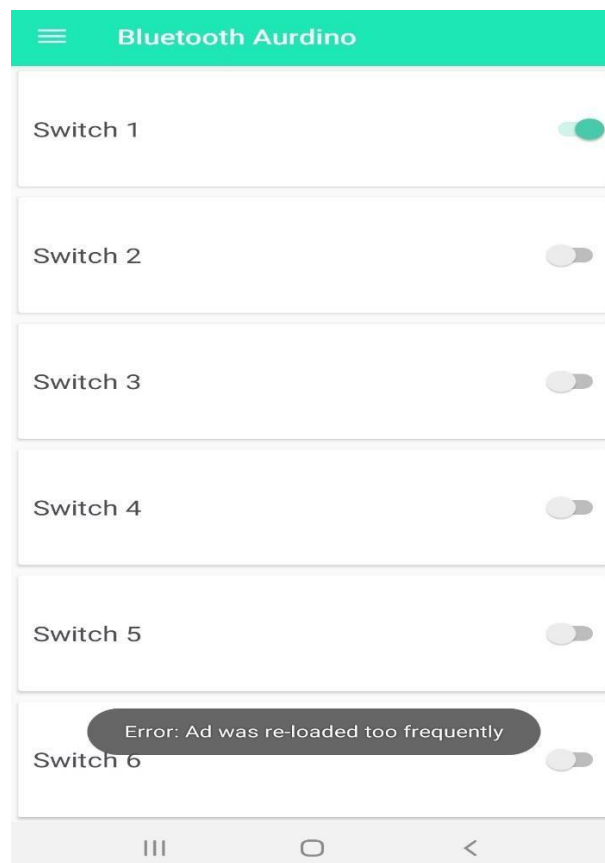
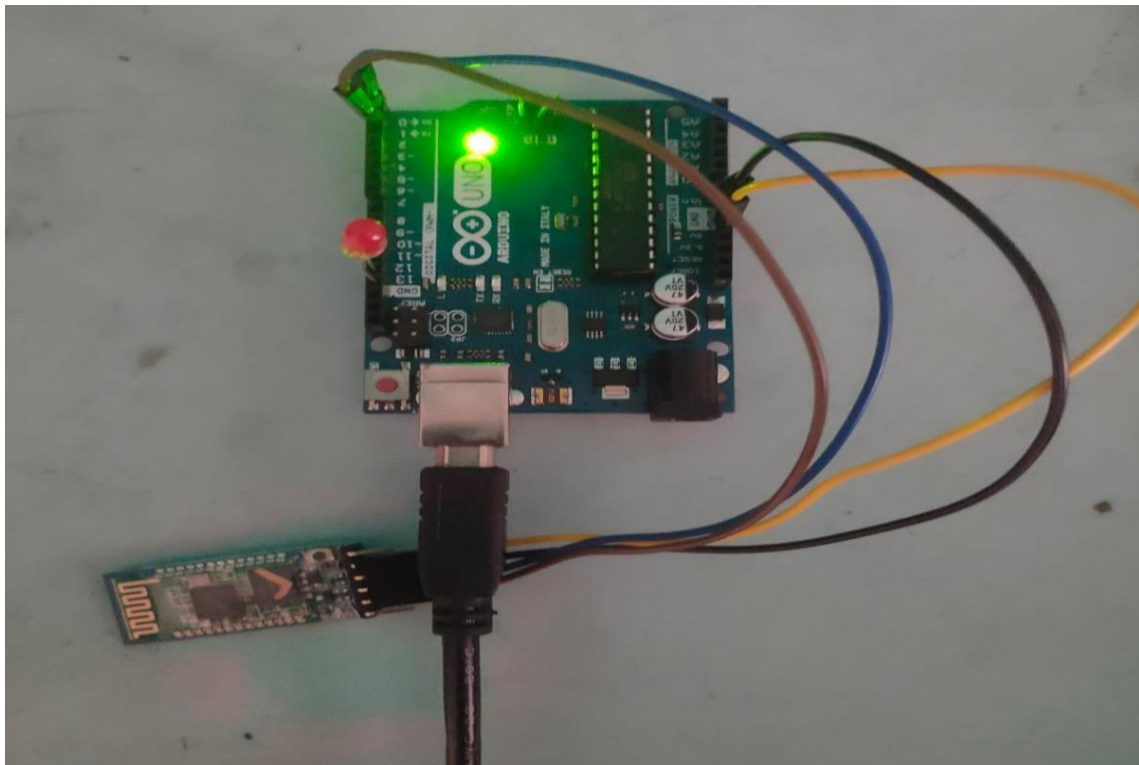
```

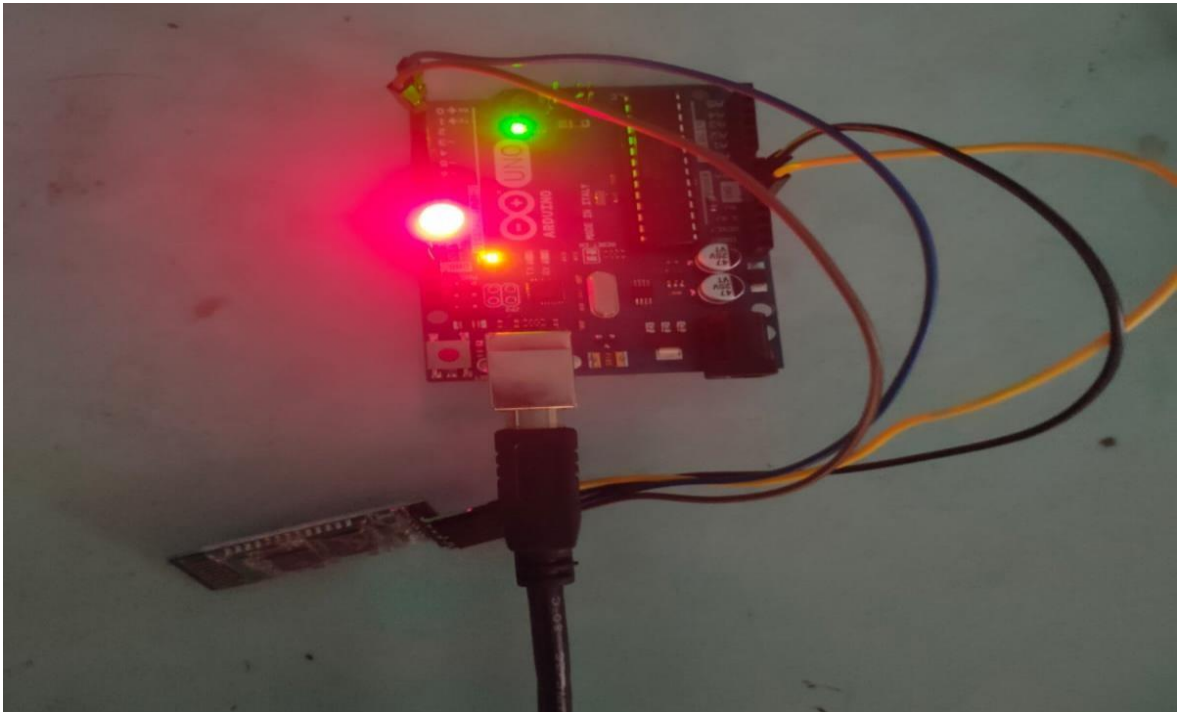
Done Saving.

Sketch uses 1796 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 189 bytes (9%) of dynamic memory, leaving 1859 bytes for local variables. Maximum is 2048 bytes.

23 Arduino Uno en COM4







Conclusion:



AISSMS
COLLEGE OF ENGINEERING



ज्ञानम् सकलजनहिताय
Approved by AICTE, New Delhi, Recognized by Govt. of Maharashtra,
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC
(Id.No. PU / PN/ Engg. / 093 (1992))
(Accredited by NAAC with grade A+)

Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Communicate between Arduino and Raspberry PI using any wireless medium like LoRa Module SX1278

Date of Performance:

Date of Submission:

Aim: To Study Communication between Arduino and Raspberry PI using LoRa Module SX1278

Theory:

Transmitter Section- Interfacing LoRa with Arduino UNO -

On the transmitter side, an Arduino UNO is interfaced with the LoRa module and DHT11 sensor. The interfacing of the Arduino UNO with LoRa and DHT11 is shown below. LoRa is getting popular day by day and there are many LoRaWAN networks all around. It consumes very low power and can communicate over a long-range. Arduino will act as Transmitter/Server and Raspberry Pi as Receiver/Client. A **DHT 11 sensor** is connected to the transmitter side which will send temperature and humidity data to the receiver side. On the receiving side, Raspberry pi will publish these readings on the Cayenne dashboard.

We are going to **send temperature and humidity values from Arduino to Raspberry Pi using the LoRa SX1278 module**. The DHT11 sensor is connected to the transmitting side where Arduino will get temperature and humidity values from DHT11 and then sends it to Raspberry Pi via the LoRa SX1278 module. These humidity and temperature values will be uploaded to the Cayenne IoT platform which can be monitored from anywhere in the world using the internet.

The LoRa module consists of 16 pins, out of these six pins are GPIO pins, and four are Ground pins. This LoRa module operates at 3.3V, and so the 3.3V pin on LoRa is connected to the 3.3v pin on the Arduino UNO board.

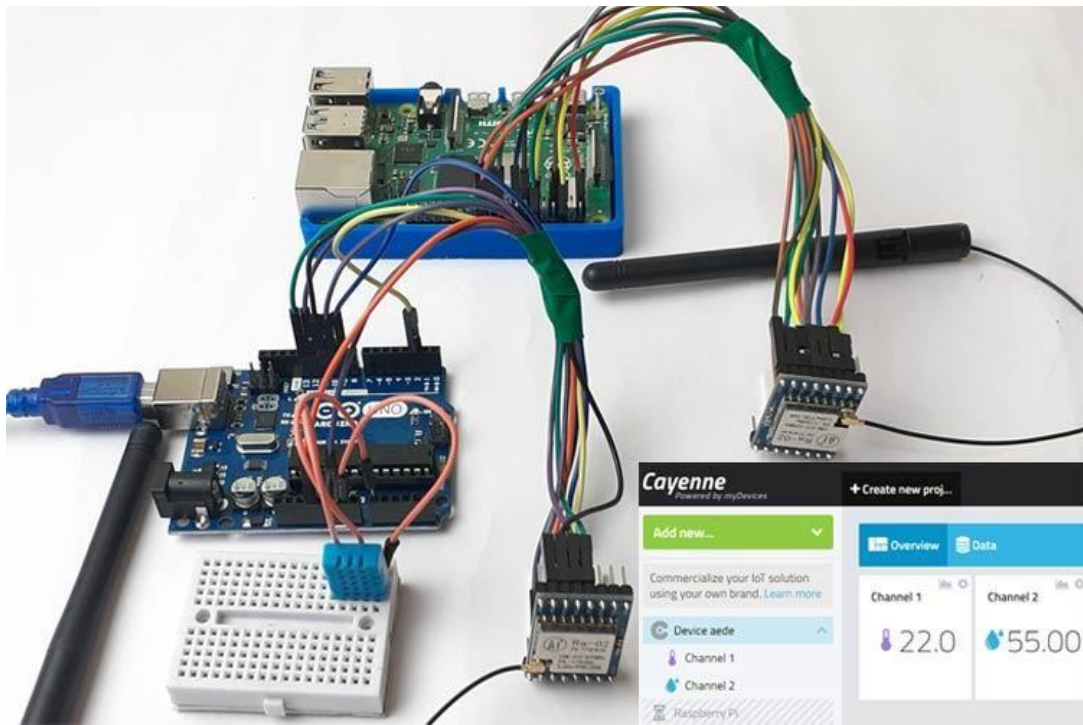


Fig. 1 Interfacing Arduino with LoRaWAN

Receiver Section- Interfacing Raspberry Pi with LoRa Module

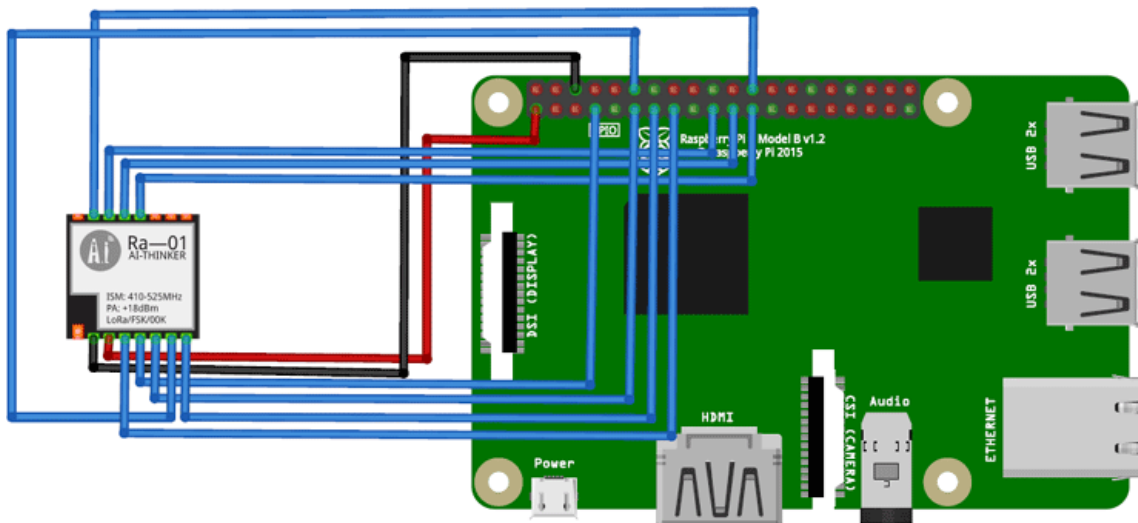


Fig. 2 Interfacing Raspberry Pi with LoRaWAN

On the receiver side, a Raspberry Pi is used to get the data from the LoRa Receiver and send it to the Cayenne platform.

Cayenne Setup for LoRa Communication

Cayenne is an IoT platform that allows you to control the microcontrollers. It is also used to upload any sensor data to the Cayenne cloud. Here temperature and humidity data from the DHT11 sensor will be uploaded to Cayenne.

This is how an **Arduino can wirelessly communicate with Raspberry Pi** using the LoRa Module and further Raspberry pi can upload the sensor data to any cloud so that it can be monitored from anywhere.

Conclusion:



AISSMS
COLLEGE OF ENGINEERING



ज्ञानम् सकलजनहिताय
Approved by AICTE, New Delhi, Recognized by Govt. of Maharashtra,
Affiliated to Savitribai Phule Pune University and recognized 2(f) and 12(B) by UGC
(Id.No. PU / PN/ Engg. / 093 (1992)
(Accredited by NAAC with grade A+)

Department of Electronics and Telecommunication Engineering

Modernized IoT

Experiment No:

Title: Smart Home System (Environment Monitoring) using IoT.

Date of Performance:

Date of Submission:

Aim: To design and implement Smart Home System to monitor parameters like Temperature, Humidity, Air Quality inside Home's Environment.

Introduction: Today environment monitoring becomes important for humans to ensure a safe and wealthy life. Monitoring requirements are extremely different depending on the environment, leading to specially appointed usage that needs adaptability. This system describes an implementation of Wireless Sensor Network that can be adjusted to various applications and it also inserts the adaptability required to be conveyed and updated without necessity of arranging complex infrastructures. The system is based on small autonomous wireless sensor nodes, small wireless receivers connected to the Internet, and a cloud architecture which provides data storage and delivery to remote clients. It permits supervisors on-site not only to monitor the current situation by using their smart-phones but also to monitor remote sites through the Internet. The proposed system is useful in predicting environmental conditions inside the home like Temperature, Humidity and Air Quality Index.

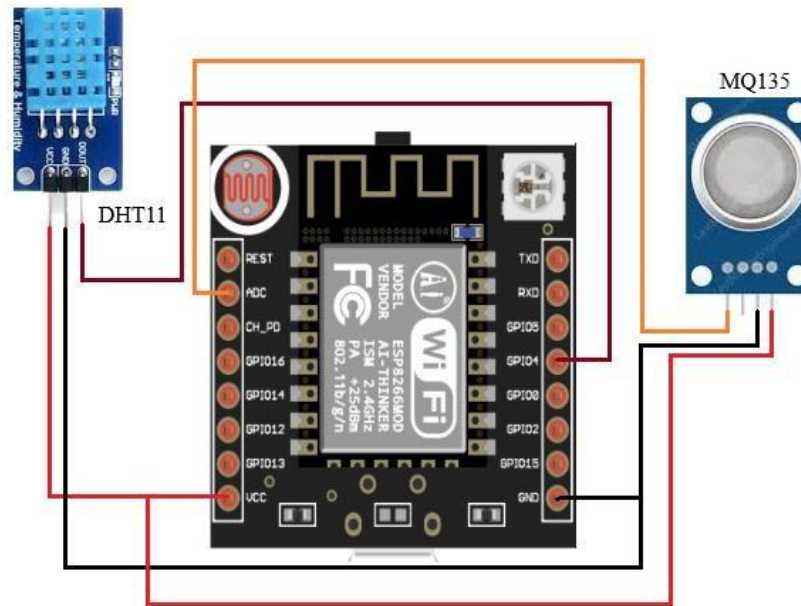
Required Hardware Components:

- ESP8266 Module
- Micro USB Cable
- DHT 11
- MQ - 135
- Bread Board
- Jumper Wires

Required Software & App:

- Arduino IDE
- Blynk Cloud Platform

Circuit Diagram:



Code:

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPLlnhcWrC1"
#define BLYNK_DEVICE_NAME "Air Quality Monitoring"
#define BLYNK_AUTH_TOKEN "-lbY72NiQwNGLqVgHQT-mjWFpEmIu4-i"
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#define DHTPIN 0
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
char auth[] = "-lbY72NiQwNGLqVgHQT-mjWFpEmIu4-i";
char ssid[] = "Piyush DC";
char pass[] = "buof0132";
void setup()
{
  Blynk.begin(auth, ssid, pass);
  dht.begin();
}
void loop()
{
  Blynk.run();
}
```

```

float h = dht.readHumidity();
float t = dht.readTemperature();
float q = analogRead(A0);
Blynk.virtualWrite(V5, t);
Blynk.virtualWrite(V6, h);
Blynk.virtualWrite(V0, q);
}

```

Results:

Air_Quality_Monitoring | Arduino 1.8.16

File Edit Sketch Tools Help

Air_Quality_Monitoring

```

#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPLnhoWrc1"
#define BLYNK_DEVICE_NAME "Air Quality Monitoring"
#define BLYNK_AUTH_TOKEN "-lbY72NiQwNGLqVgHQI-mjWFpEmIu4-i"

#include <DHT.h> //Declare DHT Library
#include <ESP8266WiFi.h> //Declare ESP8266 Board Library
#include <BlynkSimpleEsp8266.h>
#define DHTPIN 0 //Pin of DHT 22
#define DHTTYPE DHT11 //Type of DHT

DHT dht(DHTPIN, DHTTYPE); //Declare dht

int val = 0; //Declare val variable
char auth[] = "-lbY72NiQwNGLqVgHQI-mjWFpEmIu4-i"; //Put your blink token
char ssid[] = "Piyush DC"; //Put your SSID of your connection
char pass[] = "buof0132"; //Put your Password of your connection

void setup()
{
  Blynk.begin(auth, ssid, pass); //Connect the blynk to esp8266 board
  dht.begin(); //Start DHT sensor
}

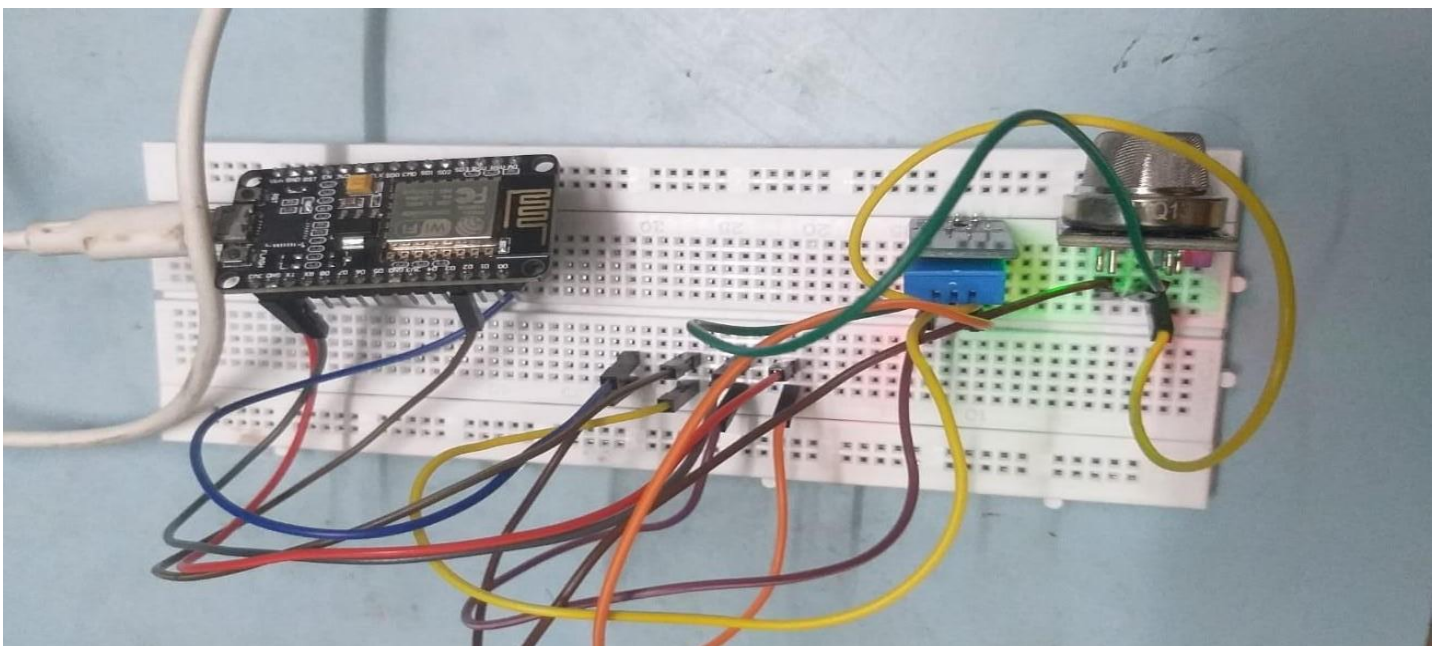
void loop() {
  Blynk.run(); //Run the Blynk
  float h = dht.readHumidity(); //Read humidity and put it in h variable
  float t = dht.readTemperature(); //Read temperature and put it in t variable
  float q = analogRead(A0); //Read air quality and put it into q variable
}

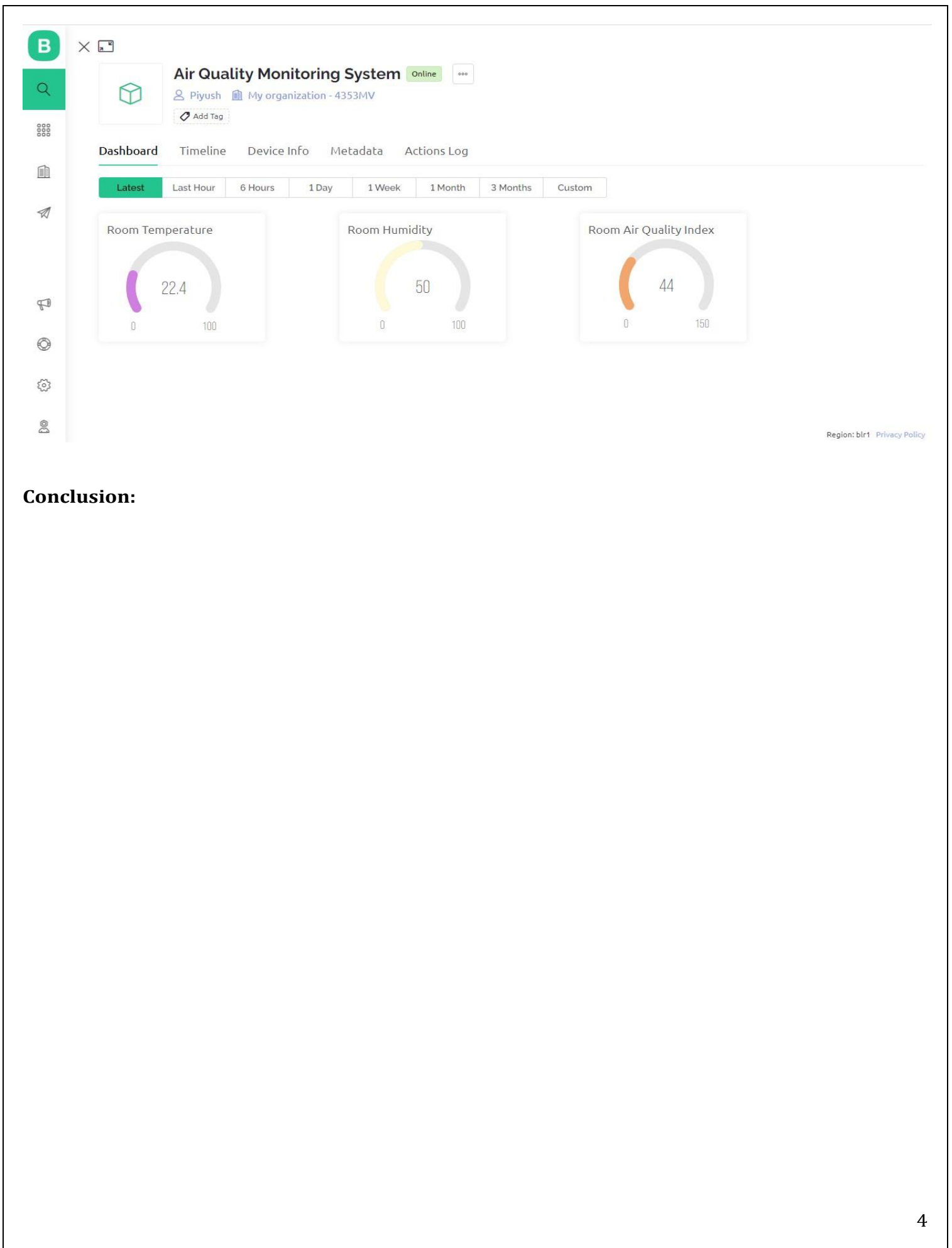
```

Done Saving.

Leaving...

Hard resetting via RTS pin...





Conclusion: