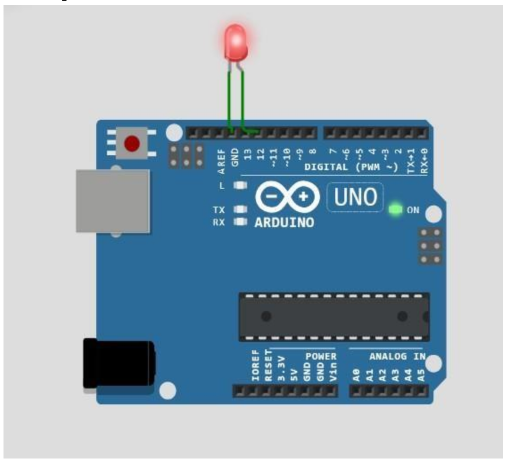
Practical 1

AIM: Study of Arduino board and Interfacing of LED (s) with Arduino.

Code:

```
 int led = 12; void setup()
 {
         // put your setup code here
         pinMode(led, OUTPUT);
 }
 void loop() {
         // put your main code here, to run
         repeatedly: digitalWrite(led,HIGH);
         delay(300);
         digitalWrite(led,LOW);
         delay(100);
}
```

Output:



Practical 2

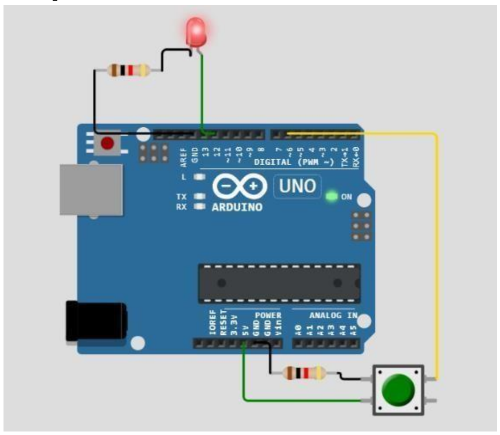
AIM: Study and implementation of Buzzer, Switches, LCD, keypad, LDR, Ultrasonic sensors and PWM interfacing with Arduino.

1) **Buzzers**

Code:

```
int led =12;
int btn=6;
int state=HIGH;
void setup() {
        // put your setup code here, to
        run once: pinMode(led, OUTPUT);
        pinMode(btn, INPUT);
}
void loop() {
// put your main code here, to run repeatedly:
int reading=digitalRead(btn);
if(reading!=state){
        state=!state;
        digitalWrite(led,state);
        } }
```

Output:

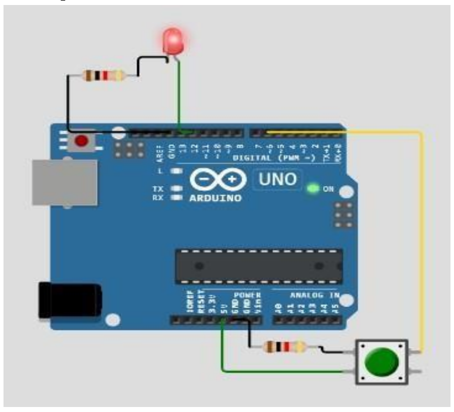


2) Switches

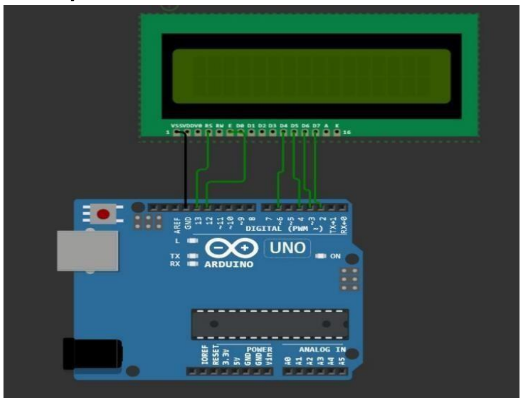
Code:

```
int led=12;
int btn=6;
int reading;
int state=LOW;
int last_state=LOW;
void setup(){
           // put your setup code here,to run once:
          pinMode(led,OUTPUT);
          pinMode(btn, INPUT);
}
void loop() {
// put your main code here, to run repeatedly:
int reading=digitalRead(btn);
if(reading==HIGH){
          if(last_state==HIGH){
          state=!last_state;
          digitalWrite(led, last_state);
   }
}
else{
          last_state=!last_state;
          digitalWrite(led,last_st ate);
   }
 }
```

Output:



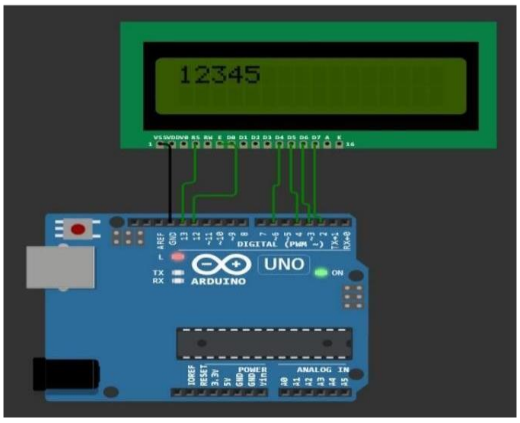
3) LCD



Code:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(13,12,6,4,3,2);
void setup() {
        // put your setup code here, t0 run once:
        lcd.begin(16,2);
}
void loop() {
        // put your main code here, to run repeatedly:
        lcd.setCursor(0,0);
        lcd.print("12345");
}
```

Output:

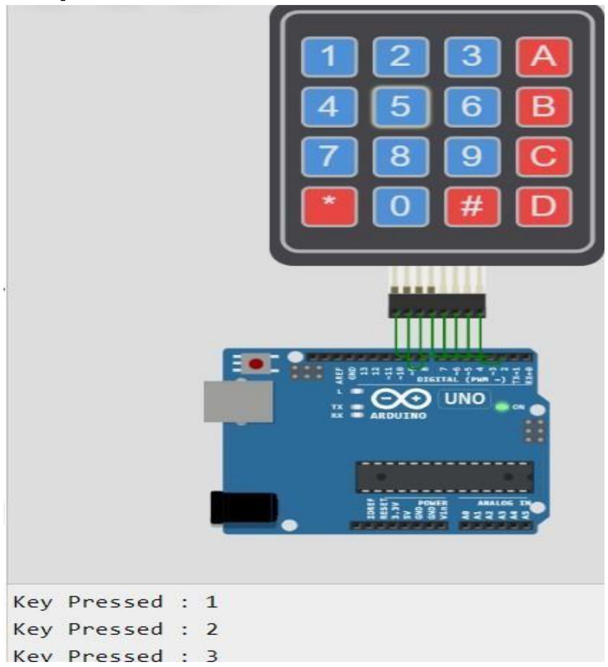


4) **Keypad:**

Code:

```
#include <Keypad.h>
const byte
ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'},
};
byte rowPins[ROWS] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS
);

void setup(){
    serial.begin(9600);
}
void loop(){
    char key = keypad.getKey();
    if (key){
    Serial.print("Key Pressed : ");
    Serial.println(key);
    }
}
```

Output:



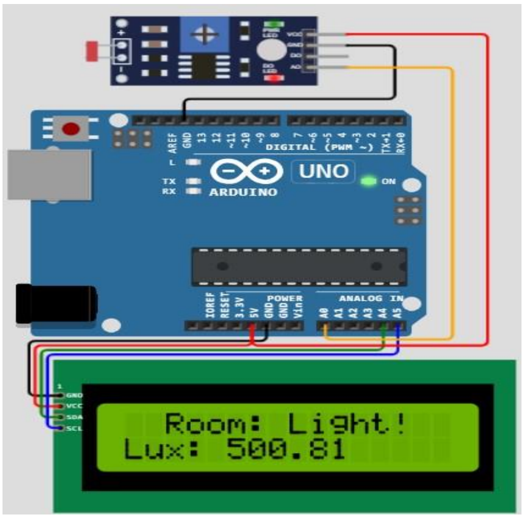
```
Key Pressed : 1
Key Pressed : 2
Key Pressed : 3
```

5) LDR

Code:

```
#include LiquidCrystal_I2C.h>
#define LDR_PIN 2

const float RL10 = 50;
LiquidCrystal_I2C lcd(0x27, 20, 4);

void setup() {
        pinMode(LDR_PIN, INPUT);
        lcd.init();
        lcd.backlig ht();
}
void loop() {
        int analogValue = analogRead(A0);
        float voltage = analogValue / 1024. * 5;
        float resistance = 2000 * voltage / (1 - voltage / 5);
        float lux = pow(RL10 * 1e3 * pow(10, GAMMA) / resistance, (1 /
GAMMA));

        lcd.setCursor(2, 0);
        lcd.print("Room: ");
        if (lux > 50) {
                lcd.print("Light!");
        }
        else {
        lcd.print("Dar k ");
}
lcd.setCursor( 0, 1);
lcd.print("Lux: ");
lcd.print(lux);
lcd.print(" ");
delay(100);
}
```

Output:

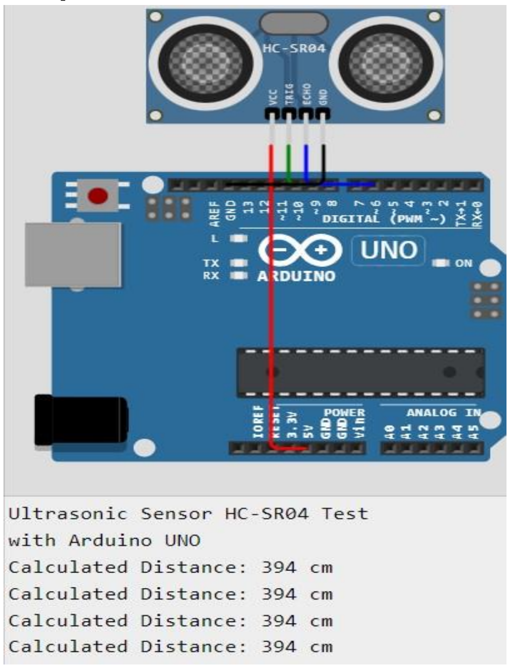


6) Ultrasonic Sensor

Code:

```
#define echoPin 6
#define triggerPin 7

void setup() {
    pinMode(triggerPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
    Serial.println("Ultrasonic Sensor HCSR04 Test");
    Serial.println("with Arduino UNO");
}
void loop() {
    long highPulseDuration;
    int calculatedDistance Cm;
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(5);
    digitalWrite(triggerPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW);
    highPulseDuration = pulseIn(echoPin, HIGH);
    calculatedDistanceCm = highPulseDuration * 0.034 / 2;

    Serial.print("Calculated Distance: ");
    Serial.print(calculatedDista nceCm);
    Serial.println("cm");
}
```

Output:



```
Ultrasonic Sensor HC-SR04 Test
with Arduino UNO
Calculated Distance: 394 cm
Calculated Distance: 394 cm
Calculated Distance: 394 cm
Calculated Distance: 394 cm
```

Practical 3

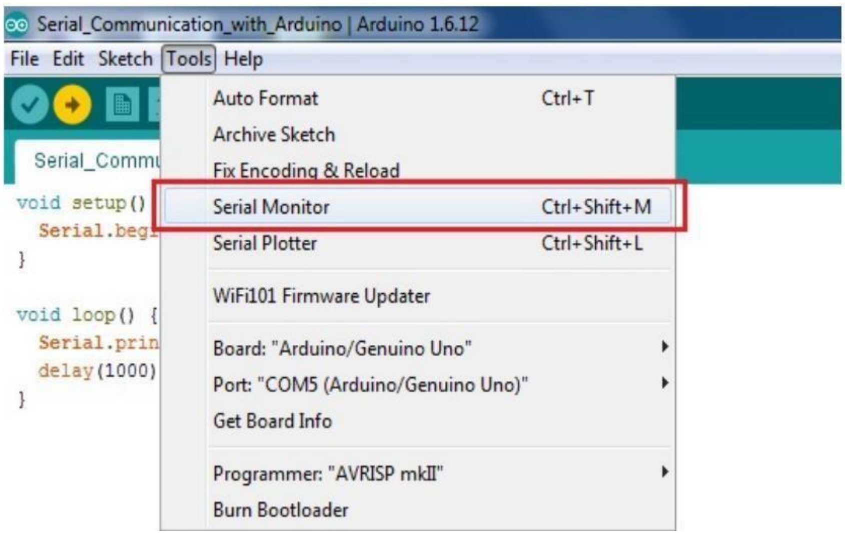
AIM: Study of serial communication and device control using serial communication with Arduino.

Example:

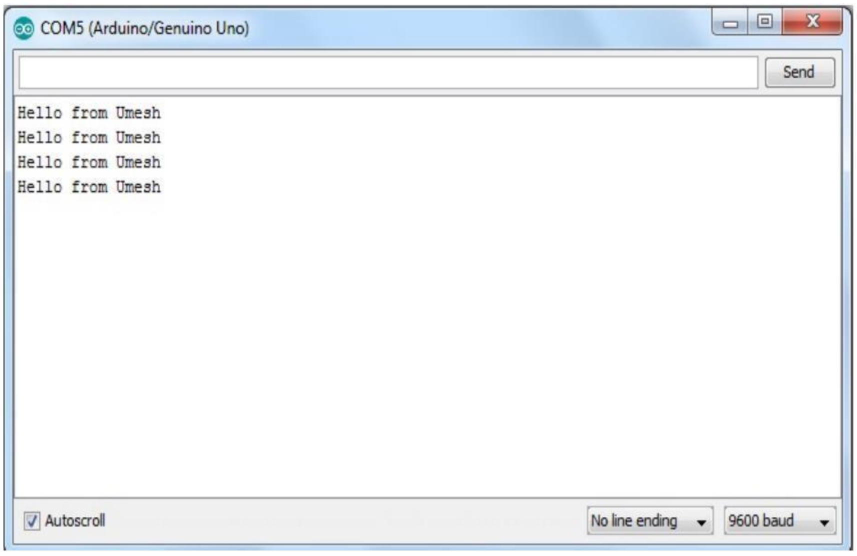
In this example, we will send a string from Arduino to PC. Once we upload the sketch into Arduino. We will see this string will be printing on Serial Monitor Window of Arduino IDE.

```
void setup() {
        Serial.begin(9600);
}
void loop() {
        Serial.println("Hello om Umesh");
        delay(1000);
}
```

Hardware Setup: We only need to connect Arduino Uno to PC over standard USB Cable. Once we have done connection we are ready to upload the sketch to Arduino and open serial monitor window to display data sent from Arduino. As we been transmitting string from Arduino to PC. We will observe that LED connected to TX Pin on Arduino will light up.


Serial_Communication_with_Arduino | Arduino 1.6.12
File Edit Sketch Tools Help
Serial_Commu
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM5 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

```
void setup()
  Serial.beg
}

void loop() {
  Serial.prin
  delay(1000)
}
```



```
Hello from Umesh
Hello from Umesh
Hello from Umesh
Hello from Umesh
```

Autoscroll ✓ · No line ending · 9600 baud

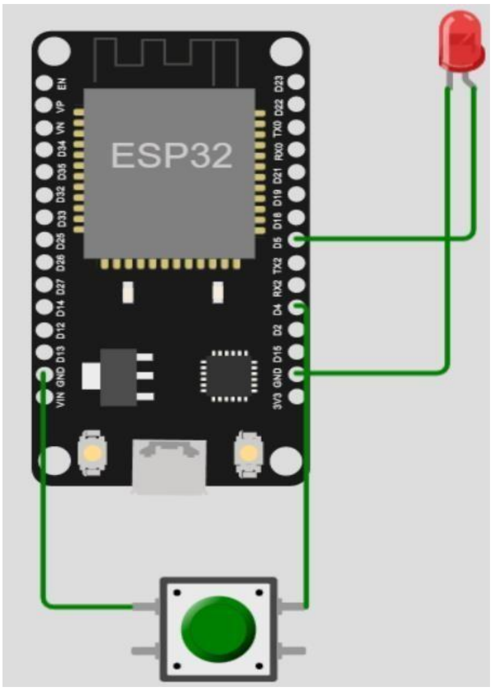
Practical 4

AIM: Study and implementation LED (s) Interfacing with NodeMCU.

NodeMCU

NodeMCU is an open-source LUA based firmware developed for the ESP8266 wifi chip. By exploring functionality with the ESP8266 chip, NodeMCU firmware comes with the ESP8266 Development board/kit i.e. NodeMCU Development board.

NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer to the ESP8266 WiFi Module.
NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board.



Code:

```
int val = 0 ;
void setup(){
        Serial.begin(9600);
        // sensor buart rate
        pinMode(5,HIGH);
        // D5 PIN FOR LED
}

void loop(){
val = digitalRead(4);  // Push Button output pin connected D1
Serial.println(val); // see the value in serial m0nitor in Arduino IDE
delay(100); // for timer
if(val == 1 ){
        digitalWrite(5,HIGH); // LED ON
}
else {
        digitalWrite(5,LOW); // LED OFF
    }
}
```

Practical 5

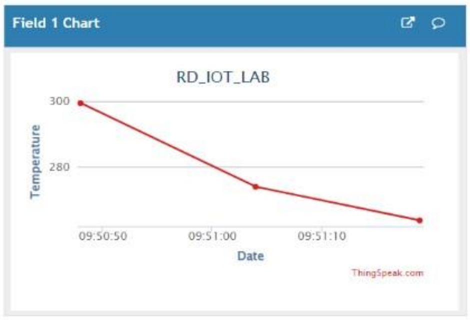
AIM: Implementation of Publishing data on ThingSpeak cloud.

Code:

```
#include <WiFi.h>
#include <ThingSpeak.h> // always include thingspeak header file after other header
files and custom macros

#define SECRET_SSID "Wokwi-GUEST" // replace with your WiFi network name
#define SECRET_PASS "" // replace with your WiFi password

#define SECRET_CH_ID 2015326   // replace 0000000 with your channel number
#define SECRET_WRITE_APIKEY "15TYOGZIQ0EHI2YQ"  // replace XYZ with your
channel write API Key

char ssid[] = SECRET_SSID;   // your network SSID (name)
char pass[] = SECRET_PASS;   // your network password
int keyIndex = 0;            // your network key Index number (needed only for WEP)

WiFiClient  client;
unsigned long myChannelNumber = SECRET_CH_ID;
const char * myWriteAPIKey = SECRET_WRITE_APIKEY;
int temppin=32;
float pinval;
float temp=0.00;
void setup()
{
  pinMode(temppin,INPUT);
  Serial.begin(115200);  //Initialize serial
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
  }
  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize ThingSpeak
}
void loop()
{
  // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED) {
        Serial.print("Attempting to connect to SSID: ");
        Serial.println(SECRET_SSID);
        while(WiFi.status() != WL_CONNECTED) {
        WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this  line if
using open or WEP network
      Serial.print(".");
```

```
          delay(5000);
    }
    Serial.println("\nConnected.");
  }
          pinval=analogRead(temppin);
          temp=pinval*(0.122);
          Serial.println(temp);
          // Write to ThingSpeak. There are up to 8 fields in a channel, allowing you to
store up to 8 different
          // pieces of information in a channel.  Here, we write to field 1.
          int x = ThingSpeak.writeField(myChannelNumber, 1, temp, myWriteAPIKey);
          if(x == 200) {
                    Serial.println("Channel update successful.");
          }
          else
          {
                    Serial.println("Problem updating channel. HTTP error code " + String(x));
          }
  delay(1000); // Wait 7 seconds to update the channel again
}
```

Output:



Practical 6

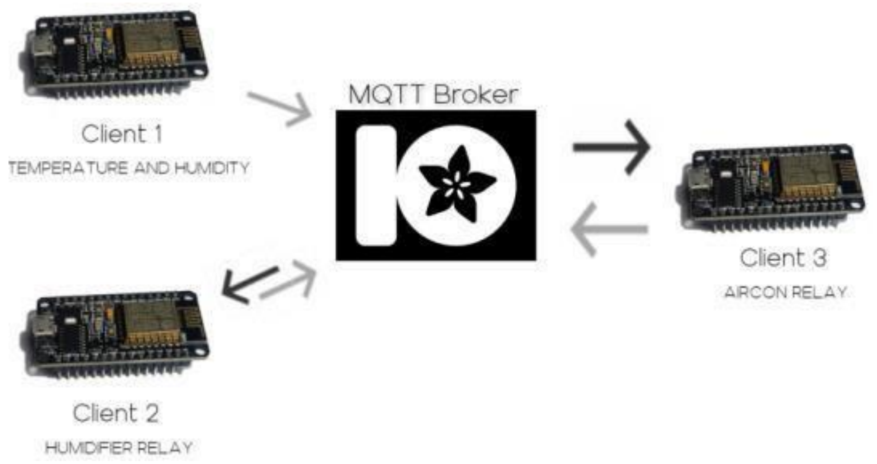
AIM: Controlling Appliances using NodeMCU MQTT over the Internet. (Adafruit Cloud).

MQTT

MQTT, also known as Message Queuing Telemetry Transport, is a protocol that uses a publish and subscribe paradigm. It is simple, lightweight, and it gives you the luxury to stay connected to your devices the entire time.
Moreover, MQTT works on top of any wireless technology. You can use it with TCP/IP, Zigbee, Bluetooth, and many more. You name it.
Before we proceed, you need to learn these terms first:

- **Broker** – The server that distributes the data to the interested clients.
- **Client** – A device that sends or receives information via the broker.
- **Topic** – A feed of data that a broker manages. Clients can publish, subscribe, or do both to a topic.
- **Publish** – A client sends information to a topic.
- **Subscribe** – A client tells the broker which topics it is interested in.
- **QoS** – QoS stands for Quality of Service. It represents the level of connection between a client and a broker.
    - 0, also known as at most once, is the lowest QOS. The broker does not acknowledge the delivery and the client does not retransmit.
    - 1, also known as at least once, the client continuously transmits the data until an acknowledgment from the broker is received.
    - 2, also known as exactly once, is the highest QOS. It is the safest but also the slowest level. It uses a 4-way handshake method.

How MQTT Works



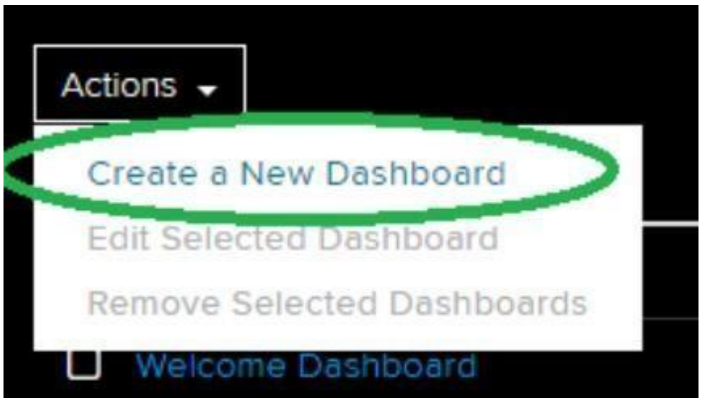
To demonstrate, suppose we have 3 clients and Adafruit IO as a broker. Client 1 is connected to a temperature and humidity sensor, Client 2 is connected to a Humidifier, and Client 3 is connected to an Air Conditioner.

In this case, the topics are Room Temperature and Humidity and Aircon Relay. Client 1 publishes data into Room Temperature and Humidity. Since Client 2 and

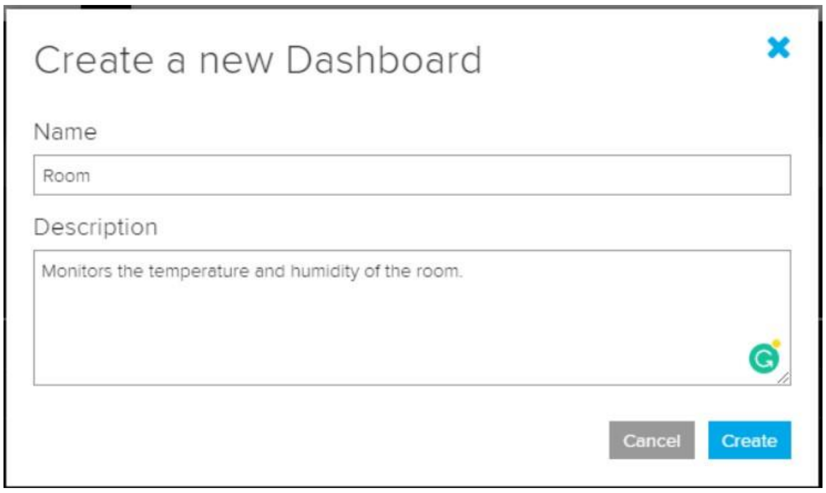
Client 3 are subscribed to the topic, the broker distributes the data to them. Client 2 opens the humidifier when the topic tells that the air is dry while Client 3 opens the Air Conditioner when the topic tells that it is warm. Moreover, Client 3 is subscribed to the topic of Aircon Relay as well. When the topic tells that the aircon is on, Client 2 opens the humidifier.
A broker just acts as a tollway for data. It doesn't send commands to the clients. It only directs the data the client is interested in.
Meanwhile, clients aren't really connected to anything other than the broker. Clients can go to deep sleep or completely shut down to conserve power without affecting the connection of the MQTT system.
Now that we understand the theory, let's proceed by setting up an MQTT broker on Adafruit IO.

Setting up MQTT on Adafruit IO

First, go to https://io.adafruit.com/ and create an account. Once logged in, go to the home page. Under Actions, select Create a New Dashboard.



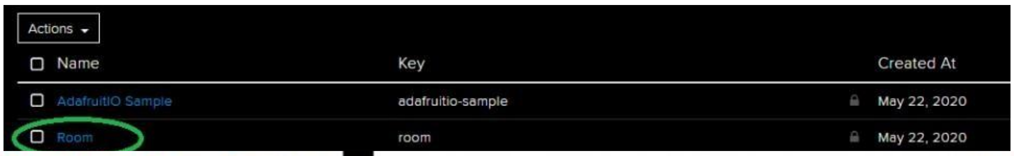
Then, click create after giving the dashboard a name and a short description.



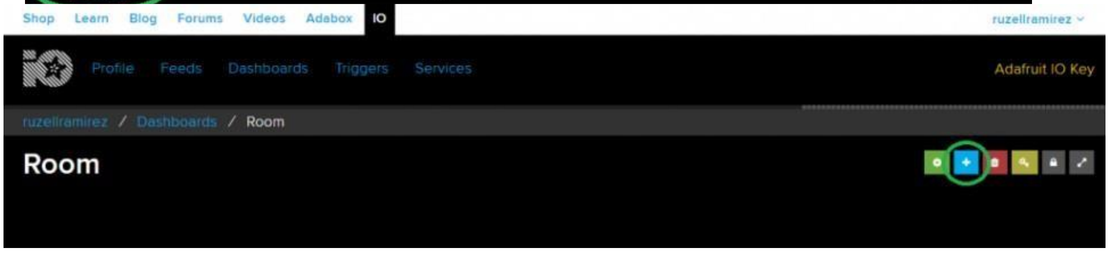
Name: Room

Description: Monitors the temperature and humidity of the room.

Next, go to your new Dashboard.

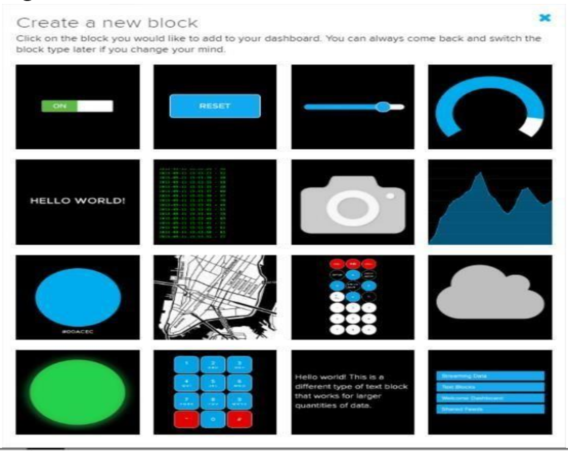


Name	Key	Created At
AdafruitIO Sample	adafruitio-sample	May 22, 2020
Room	room	May 22, 2020



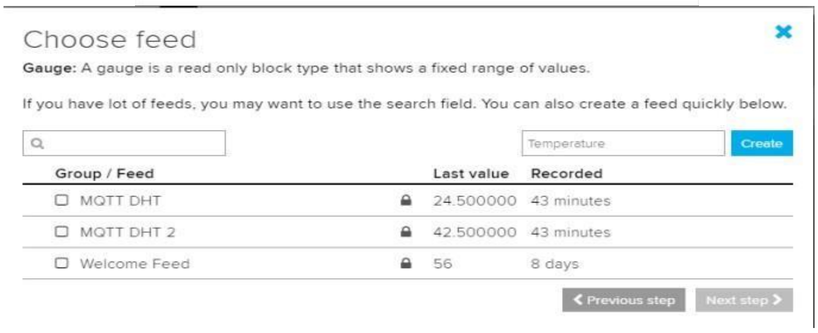
As you can see, the dashboard is still empty. You should populate it first by creating a new block. In order to do that, press the blue plus button on the right side of the dashboard.

Blocks are switches, buttons, levers, gauges, and more visualization techniques that represent and react to your data. For our sample project, we need 2 gauges to display the room's temperature and humidity and a switch to toggle an LED. After selecting a block, Adafruit IO prompts you to choose a feed. Feeds can also be called MQTT topics. We need 3 feeds to hold the temperature and humidity values of the DHT sensor, and the status of the LED. Create a feed using the create button on the right.


Create a new block

Click on the block you would like to add to your dashboard. You can always come back and switch the block type later if you change your mind.

ON
RESET
HELLO WORLD!
Hello world! This is a different type of text block that works for larger quantities of data.

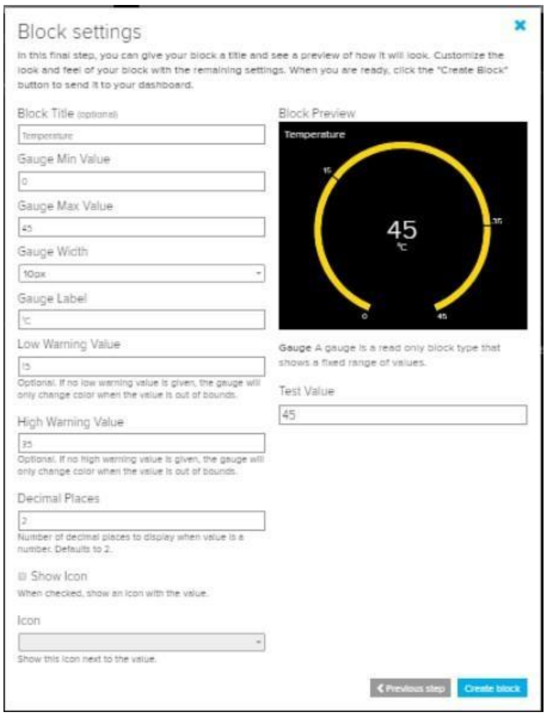

Choose feed

Gauge: A gauge is a read only block type that shows a fixed range of values.

If you have lot of feeds, you may want to use the search field. You can also create a feed quickly below.

Group / Feed	Last value	Recorded
MQTT DHT	24.500000	43 minutes
MQTT DHT 2	42.500000	43 minutes
Welcome Feed	56	8 days

Next, configure the block settings as shown below.



Block settings

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional): Temperature

Gauge Min Value: 0

Gauge Max Value: 45

Gauge Width: 10px

Gauge Label: °C

Low Warning Value: 15

Optional. If no low warning value is given, the gauge will only change color when the value is out of bounds.

High Warning Value: 35

Optional. If no high warning value is given, the gauge will only change color when the value is out of bounds.

Decimal Places: 2

Number of decimal places to display when value is a number. Defaults to 2.

Show Icon

When checked, show an icon with the value.

Icon

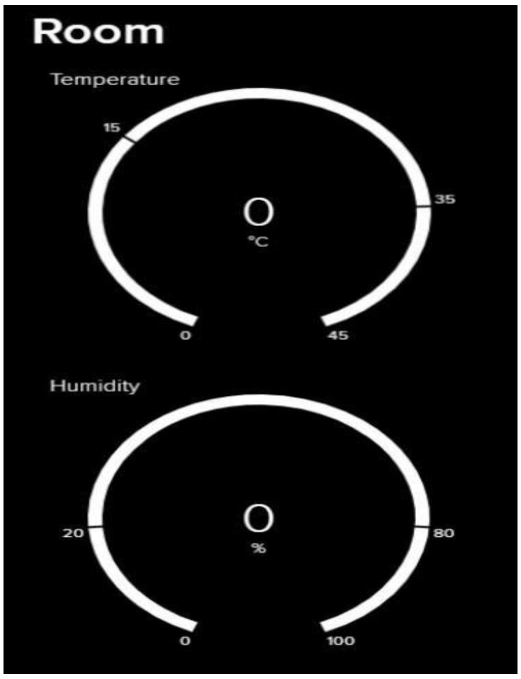
Show this icon next to the value.

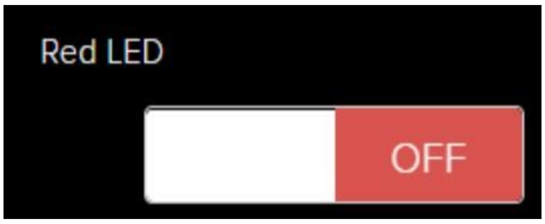
Block Preview

Gauge A gauge is a read only block type that shows a fixed range of values.

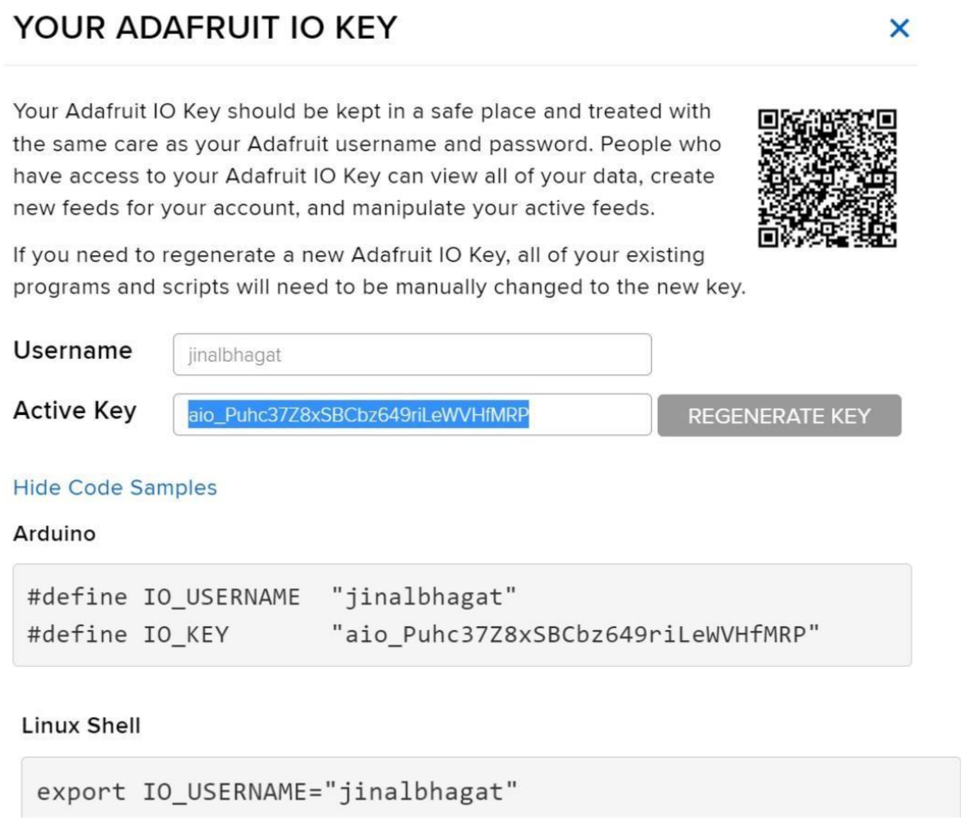
Test Value: 45

Do the same with Humidity and the LED switch so that you can monitor simultaneously.





To fill the blocks with data, you need your personal username and key. To obtain both information, go back to the homepage and press Adafruit IO key. A window will appear that displays your username and key. Grab a pen and paper and write them down.



YOUR ADAFRUIT IO KEY

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username: jinalbhagat

Active Key: aio_Puhc37Z8xSBCbz649riLeWVHfMRP

Arduino

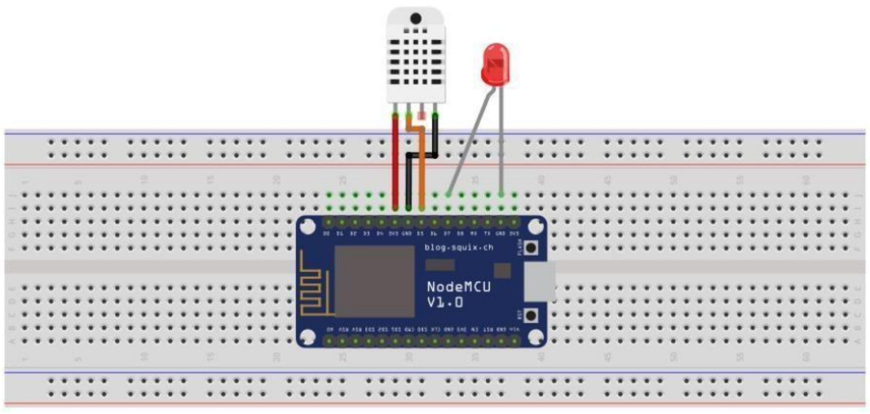
```
#define IO_USERNAME  "jinalbhagat"
#define IO_KEY       "aio_Puhc37Z8xSBCbz649riLeWVHfMRP"
```

Linux Shell

```
export IO_USERNAME="jinalbhagat"
```

Preparing the Hardware

Connect the NodeMCU, DHT22 sensor, and LED as shown below.

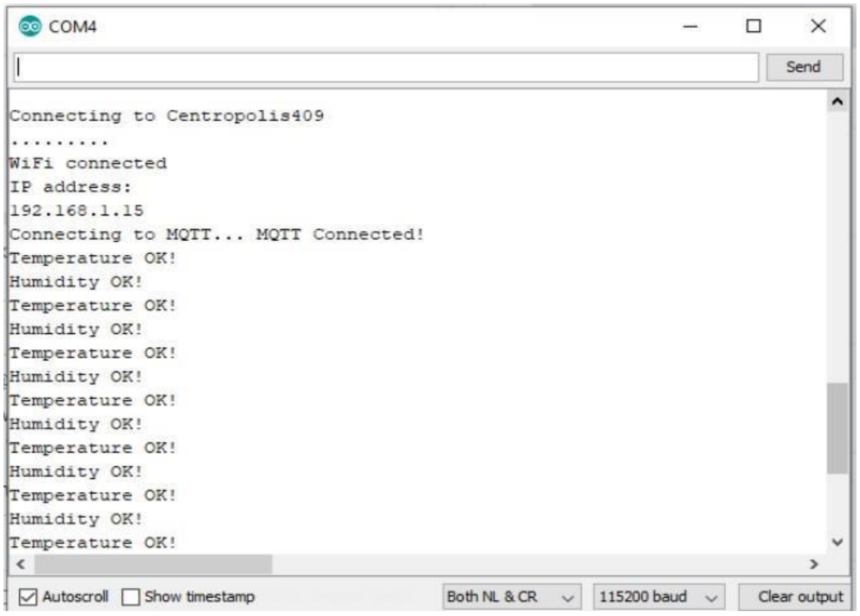


Code:

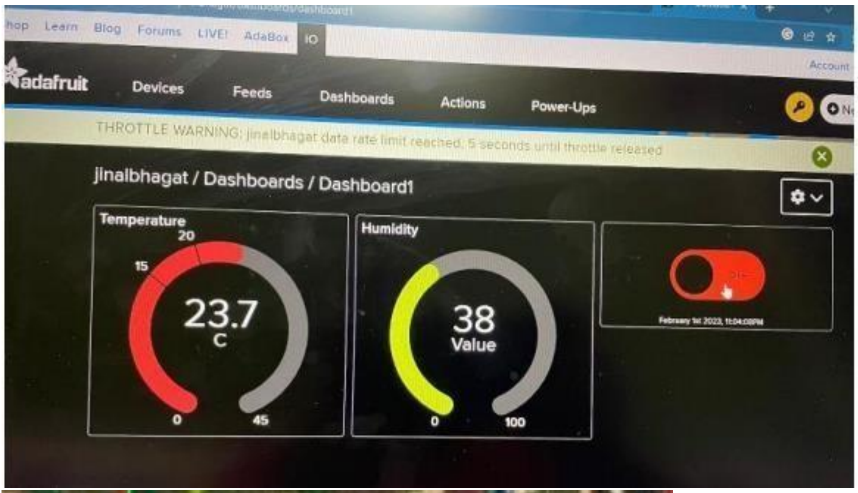
```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include <DHT.h>

#define DHTPin D5
#define DHTType DHT22 DHT dht(DHTPin, DHTType);
float t,h;

#define WLAN_SSID "PLDTHOMEFIBR22218"
```

```
#define WLAN_PASS "PLDTWIFI29SKA"

#define AIO_SERVER "io.adafruit.com"
#define AIO_SERVERPORT 1883
#define AIO_USERNAME "jinalbhagat"
#define AIO_KEY "aio_WOPW54bbyKB0tzfIO33D6WFpynGq"


WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);

Adafruit_MQTT_Publish Temperature = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/Temperature");

Adafruit_MQTT_Publish Humidity =Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/Humidity");

Adafruit_MQTT_Subscribe LED = Adafruit_MQTT_Subscribe(&mqtt,
AIO_USERNAME "/feeds/LED");

void
MQTT_connect();
void setup() {
Serial.begin(115 200);
delay(10);
pinMode(D7, OUTPUT);
dht.begin();
delay(10);
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(WLAN_SSID);
WiFi.begin(WLAN_SSID, WLAN_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println();
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
mqtt.subscribe(&LED);
}
```

```
void loop() {
MQTT_connect();
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(5000))) {
if (subscription == &LED) {
Serial.print(F("Got:”));
String ledstatus = (char *)LED.lastread;
if (ledstatus == "ON") {
digitalWrite(D7, HIGH);
Serial.println(ledstatus);
}
else if (ledstatus == "OFF") {
digitalWrite(D7, LOW);
Serial.println(ledstatus);
}
}
}
t = dht.readTemperature();
h = dht.readHumidity();
if (! Temperature.publish(t)) {
Serial.println(F("Temperature Failed"));
}
else {
Serial.println(F("Temperature OK!"));
}
if (! Humidity.publish(h)) {
Serial.println(F("Humidity Failed"));
}
else {
Serial.println(F("Humidity OK!"));
}
}
void MQTT_connect() {
int8_t ret;
if (mqtt.connected()) {
return;
}
Serial.print("Connecting to MQTT... ");
uint8_t retries = 3;
while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
Serial.println(mqtt.connectErrorString(ret));
Serial.println("Retrying MQTT connection in 5 seconds...");
mqtt.disconnect(); delay(5000); // wait 5 seconds retries--;
if (retries == 0) {
// basically die and wait for WDT to reset me while (1);
}
}
Serial.println("MQTT Connected!");
}
```

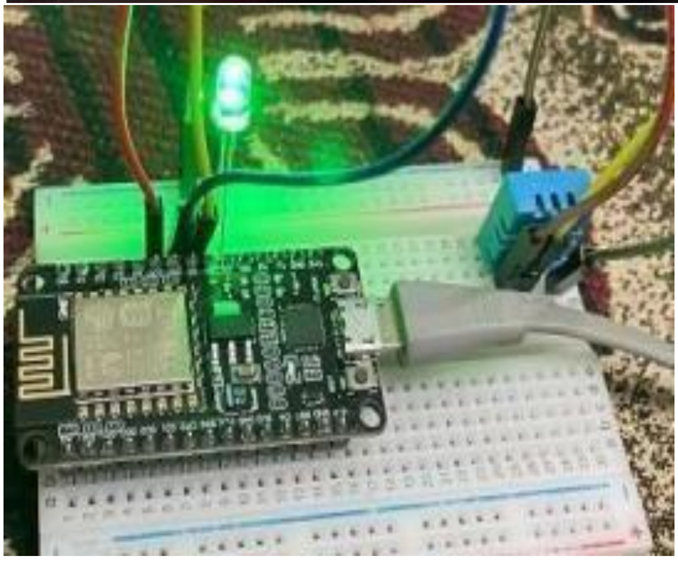


```
Connecting to Centropolis409
.........
WiFi connected
IP address:
192.168.1.15
Connecting to MQTT... MQTT Connected!
Temperature OK!
Humidity OK!
Temperature OK!
Humidity OK!
Temperature OK!
Humidity OK!
Temperature OK!
Humidity OK!
Temperature OK!
Humidity OK!
Temperature OK!
Humidity OK!
Temperature OK!
```



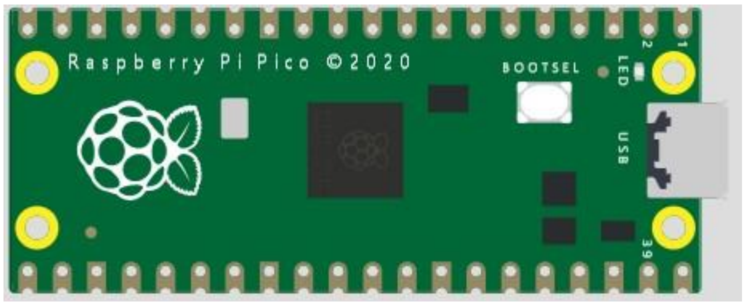
THROTTLE WARNING: jinalbhagat data rate limit reached, 5 seconds until throttle released

jinalbhagat / Dashboards / Dashboard1



Practical 7

AIM: Study and Setup Raspberry Pi board.



What is a Raspberry Pi?

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education.

The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core CPU clocking in at over 1.5GHz, and 4GB RAM. The price point for Raspberry Pi has always been under $100 (usually around $35 USD), most notably the Pi Zero, which costs just $5. All over the world, people use the Raspberry Pi to learn programming skills, build hardware projects, do home automation, implement Kubernetes clusters and Edge computing, and even use them in industrial applications.

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT).

What Raspberry Pi models have been released?

There have been many generations of the Raspberry Pi line: from Pi

1 to 4, and even a Pi 400. There has generally been a Model A and a Model B of most generations. Model A has been a less expensive variant, and tends to have reduced RAM and fewer ports (such as USB and Ethernet). The Pi Zero is a spinoff of the original (Pi 1) generation, made even smaller and cheaper. Here's the lineup so far:

- Pi 1 Model B (2012)
- Pi 1 Model A (2013)
- Pi 1 Model B+ (2014)
- Pi 1 Model A+ (2014)
- Pi 2 Model B (2015)
- Pi Zero (2015)

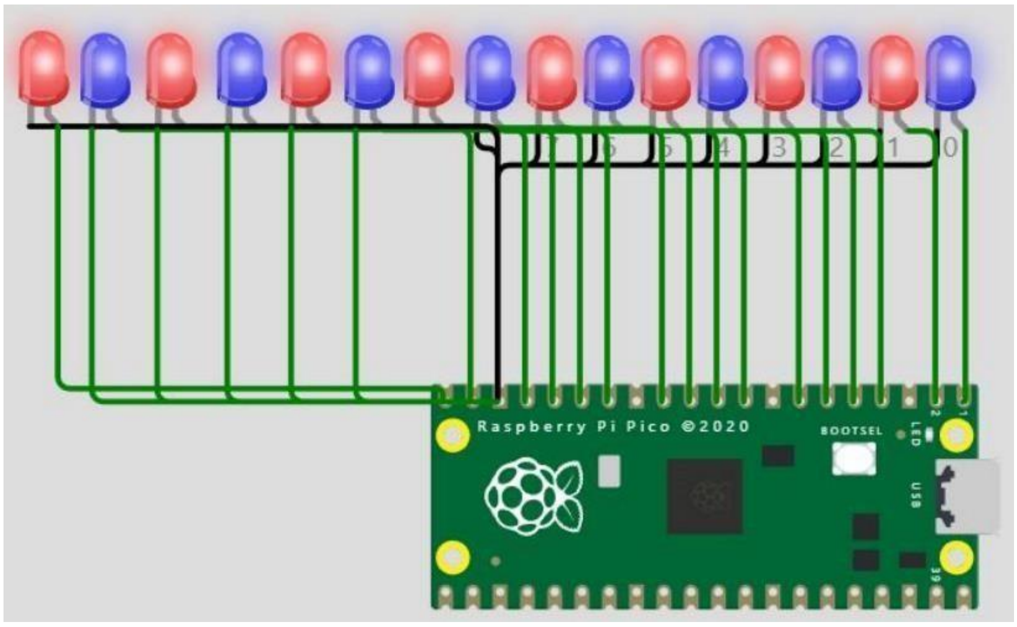
- Pi 3 Model B (2016)
- Pi Zero W (2017)
- Pi 3 Model B+ (2018)
- Pi 3 Model A+ (2019)
- Pi 4 Model A (2019)
- Pi 4Model B (2020)
- Pi 400 (2021)

What can you do with a Raspberry Pi?

Some people buy a Raspberry Pi to learn to code, and people who can already code use the Pi to learn to code electronics for physical projects. The Raspberry Pi can open opportunities for you to create your own home automation projects, which is popular among people in the open-source community because it puts you in control, rather than using a proprietary closed system.

Practical 8

AIM: Study and implementation of LED(s) Interfacing with raspberry pi.



Code:

```
void enable_sio(int pin) {
        uint32_t *PIN_CTRL_REG = (uint32_t*)IO_BANK0_BASE + pin * 2 + 1;
        *PIN_CTRL_REG = 5; // 5 = SIO function
}

void setup() {
// Enable the SIO function for pins GP0 to GP7
for (int i = 0; i < 16; i++) {
        enable_sio(i);
}
// Enable output on pins GP0 to GP7:
// sio_hw->gpio_oe points to 0xd0000020 (GPIO_OE) sio_hw->gpio_oe =
0b1111111111111111;

// Set initial pin pattern
// sio_hw->gpio_out points to 0xd0000010

(GPIO_OUT) sio_hw->gpio_out = 0b1010101010101010;
}


void loop() {
```

```
//pattern 1
for (int i = 0; i < 20; i++) {
        sio_hw->gpio_togl = 0b1111111111111111;
        delay(420 - 20 * i);
}
//pattern 2
for (int i = 0; i < 16; i++) {
        sio_hw->gpio_out = 65535 >> i;
        delay(300);
}
//pattern 3
for (int i = 0; i < 16; i++) {
        sio_hw->gpio_out = (65535 >> i);
        sio_hw->gpio_togl = 0b1111111111111111;
        delay(300);
}
//pattern 4
for (int i = 0; i < 16; i++) {
        sio_hw->gpio_out = 1 << i;
        delay(100);
}
//pattern 5
for (int i = 0; i < 16; i++) {
        sio_hw->gpio_out = 32768 >> i;
        delay(100);
}
//pattern 6
sio_hw->gpio_out = 65535;
for (int i = 0; i < 16; i++)
{
        sio_hw->gpio_togl = 0b1111111111111111;
        delay(200);
}
//pattern 7
sio_hw->gpio_out = 65280;
for (int i = 0; i < 8; i++) {
        sio_hw->gpio_togl = 0b1111111111111111;
        delay(300);
}
//pattern 8
sio_hw->gpio_out = 61680;
for (int i = 0; i < 16; i++) {
        sio_hw->gpio_togl = 0b1111111111111111;
        delay(300);
}
//pattern 9
```

```
sio_hw->gpio_out = 52428;
for (int i = 0; i < 16; i++) {
        sio_hw->gpio_togl = 0b1111111111111111; delay(300);
}
//pattern 10
sio_hw->gpio_out = 43690;
for (int i = 0; i < 16; i++)
{
        sio_hw->gpio_togl = 0b1111111111111111; delay(300);
}
}
```

Practical 9

AIM: Study of Node-red programming tool.

Node-Red

Node-RED is a programming tool that allows users to create visual programming flows for IoT and other applications. It was developed by IBM and is now an open-source project under the JS Foundation. Node-RED uses a web-based visual editor that allows users to drag and drop nodes onto a workspace and connect them with wires to create a flow.

Each node represents a specific action or function, such as reading data from a sensor, performing a computation, or sending a message to another system. The nodes are grouped into categories based on their functionality, making it easy to find the right node for a specific task. Node-RED also supports the creation of custom nodes using JavaScript, allowing users to extend its capabilities.

Node-RED has a built-in library of nodes that can be used to interact with a variety of IoT devices and services, such as MQTT, TCP, HTTP, and CoAP protocols, and popular cloud services like AWS and Azure. This makes it an ideal tool for building IoT applications that require data collection, processing, and transmission.

Node-RED can be easily installed on a local machine or deployed on a cloud platform like IBM Cloud, AWS, or Microsoft Azure. It can also be integrated with other programming languages like Python, JavaScript, and Java.

In summary, Node-RED is a visual programming tool that simplifies the development of IoT applications. It provides a user-friendly interface that allows users to create complex workflows without writing any code. It is easy to install, supports a wide range of IoT protocols and services, and can be extended using custom nodes.

Node-RED has gained popularity in the IoT community due to its ease of use and flexibility. It has a large and active community of developers who contribute to the project by creating and sharing new nodes, providing support, and contributing to the documentation.

One of the key features of Node-RED is its ability to quickly prototype and test IoT applications. With the visual editor, developers can create a working prototype of an application in a matter of minutes, allowing them to quickly validate their ideas and experiment with different approaches. This reduces development time and costs, making it an ideal tool for startups and small businesses.

Another advantage of Node-RED is its ability to integrate with other tools and services. It can be used with popular messaging platforms like Slack and Telegram, as well as data visualization tools like Grafana and Tableau. This makes it easy to create end-to-end solutions that connect IoT devices, cloud services, and user interfaces.

Practical 10

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

Node-Red

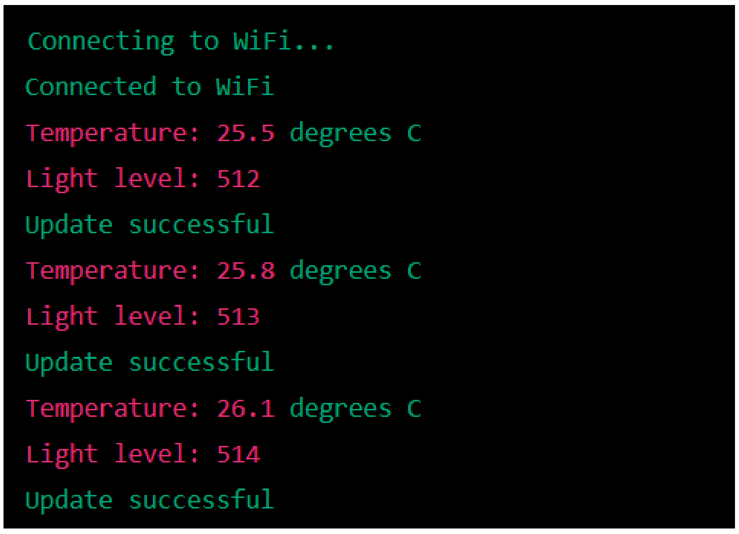
Node-RED is a programming tool that allows users to create visual programming flows for IoT and other applications. It was developed by IBM and is now an open-source project under the JS Foundation. Node-RED uses a web-based visual editor that allows users to drag and drop nodes onto a workspace and connect them with wires to create a flow.

Practical 11

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

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ThingSpeak.h>

// ThingSpeak channel details
const char* ssid = "jio_4G";
const char* password = "Mahek@247";
const char* server = "api.thingspeak.com";
const String apiKey = "AB12CDE34FG56HI78JK9LMN0OP12QR34";
const unsigned long postingInterval = 10000; // 10 seconds between updates

// Define the pins for the temperature and light sensors
const int tempSensorPin = A0;
const int lightSensorPin = A1;

// Initialize the WiFi client and ThingSpeak client objects
WiFiClient client;
ThingSpeakClient thingSpeak(client);

// Initialize the last update time
unsigned long lastUpdate = 0;

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

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

  // Initialize the ThingSpeak client
  thingSpeak.begin(server, apiKey);
}

void loop() {
  // Check if it's time to update ThingSpeak
  if (millis() - lastUpdate > postingInterval) {
    // Read the temperature and light sensor values
    float temperature = analogRead(tempSensorPin) * 0.1; // Assuming 10mV/degree
C conversion
    int light = analogRead(lightSensorPin);

    // Print the sensor values to the serial monitor
    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.println(" degrees C");
    Serial.print("Light level: ");
    Serial.println(light);

    // Update ThingSpeak with the sensor values
    ThingSpeak.setField(1, temperature);
    ThingSpeak.setField(2, light);
    int response = ThingSpeak.writeFields();

    // Check if the update was successful
    if (response == 200) {
      Serial.println("Update successful");
    } else {
      Serial.print("Update failed with error code ");
      Serial.println(response);
    }

    // Update the last update time
    lastUpdate = millis();
  }
}
```

Output:



```
Connecting to WiFi...
Connected to WiFi
Temperature: 25.5 degrees C
Light level: 512
Update successful
Temperature: 25.8 degrees C
Light level: 513
Update successful
Temperature: 26.1 degrees C
Light level: 514
Update successful
```

Practical 12

AIM: Case study on IoT Applications like - Home Automation: This home automation system based on IoT automates the functioning of household appliances over the Internet.

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

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

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

Case Study:

One of the most significant advancements in technology has been the introduction of the Internet of Things (IoT). IoT has paved the way for various applications, including home automation, smart agriculture, smart street lights, and others. In this case study, we will delve into some of the popular IoT applications and their implementations.

Home Automation:

Home automation based on IoT is the automatic control of household appliances over the internet. The system uses a network of interconnected devices such as sensors, controllers, and actuators to control and monitor the appliances. The system enables homeowners to control their home’s temperature, lighting, security, and other aspects remotely. For instance, they can turn on the lights, adjust the thermostat, or check their security cameras from their smartphone or tablet.

Home automation based on IoT has revolutionized the way we interact with our homes. With the help of smart devices and sensors, homeowners can now control their homes' appliances, lighting, temperature, and security from anywhere in the world, using their smartphones or other smart devices.

The IoT-based home automation system consists of a network of devices such as sensors, controllers, and actuators. These devices are connected through a wireless network, enabling them to communicate with each other and with the homeowner's smartphone or tablet.

Some of the key features of home automation based on IoT are:

- Smart Lighting: With the help of smart light bulbs, homeowners can control the lighting in their homes remotely. They can turn on or off the lights, adjust the brightness, and even change the color of the light.

- Temperature Control: Smart thermostats enable homeowners to control the temperature of their homes remotely. They can adjust the thermostat to maintain a comfortable temperature, even when they are not at home.

- Security: Home automation based on IoT also includes smart security systems that enable homeowners to monitor their homes remotely. They can view live footage from their security cameras, receive alerts when there is any suspicious activity, and even remotely lock or unlock their doors.

- Entertainment: IoT-based home automation systems also allow homeowners to control their entertainment systems remotely. They can control their TV, music systems, and even their gaming consoles using their smartphones or other smart devices.

- Energy Efficiency: IoT-based home automation systems can help homeowners save energy by automatically turning off appliances when they are not in use. They can also monitor their energy consumption and adjust their usage accordingly.

Overall, home automation based on IoT offers homeowners a convenient and efficient way to control their homes. It not only makes life more comfortable but also helps save time, energy, and resources.