IoT LAB MANUAL

Lab Course Outcomes

Upon the completion of **IOT** Laboratory, the student will be able to:

CO No.	Course Outcomes
1	Have understanding of Arduino/Raspberry Pi
2	Apply the skills learned by designing, building, and testing a microcontroller-based embedded system
3	Publishing/Subscribing to connect, collect data, monitor and manage assets
4	Remotely monitor data and control devices
5	Perform experiments and mini projects on IoT

Name: ______________________________

Semester: ______________________________

Branch: ______________________________

LIST OF EXPERIMENTS

.

EXPERIMENT NO.1

AIM: Familiarization with Arduino/Raspberry Pi and perform necessary software installation.

OBJECTIVES: student should get the knowledge of Arduino ide and different types of Arduino board

OUTCOMES: Student will be get knowledge of Arduino ide and different types of Arduino board hardware

Arduino:

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.

Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Arduino is a prototype platform (open-source) based on an easy-to-use hardware and software. It consists of a circuit board, which can be programed (referred to as a microcontroller) and a ready-made software called Arduino IDE (Integrated

Development Environment), which is used to write and upload the computer code to the physical board.

The key features are −

- Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

- You can control your board functions by sending a set of instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).

- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.

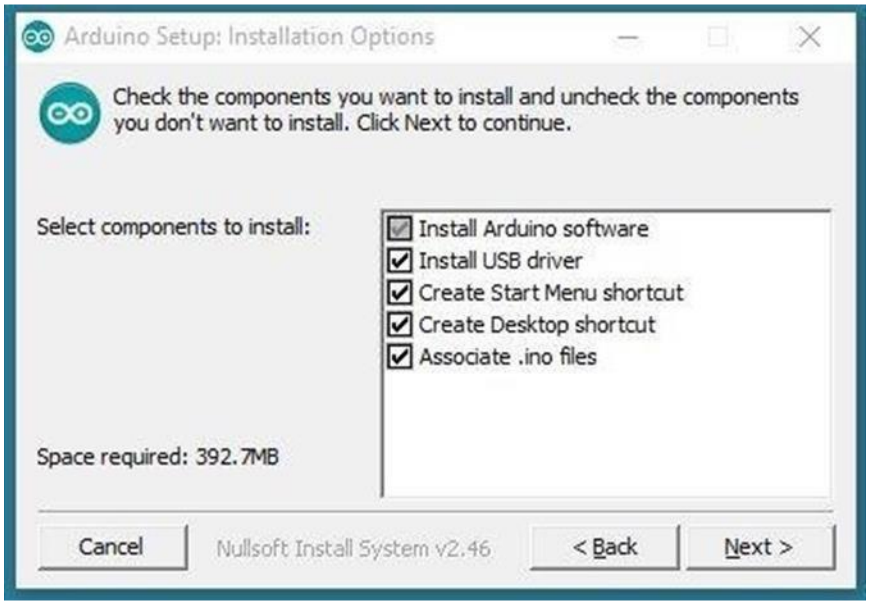
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.

- Finally, Arduino provides a standard form factor that breaks the functions of the micro-controller into a more accessible package.

Download the Arduino Software (IDE)

- Get the latest version from the arduino.cc web site. You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.

- When the download finishes, proceed with the installation and please allow the driver installation process when you get a warning from the operating system.


Arduino Setup: Installation Options
Cancel
Nullsoft Install System v2.46
< Back
Next >

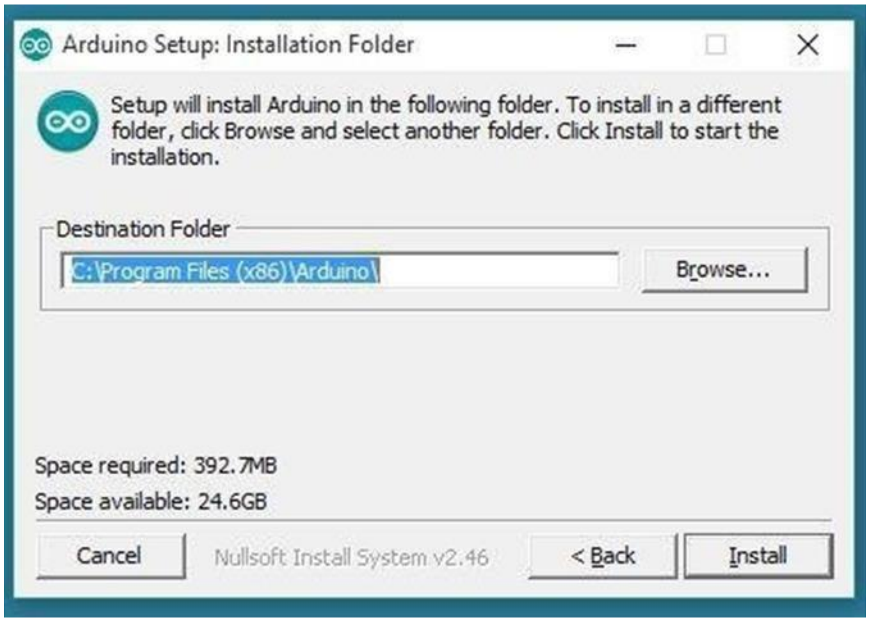
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Install Arduino software
- [✓] Install USB driver
- [✓] Create Start Menu shortcut
- [✓] Create Desktop shortcut
- [✓] Associate .ino files

Space required: 392.7MB

- Choose the components to install


Arduino Setup: Installation Folder

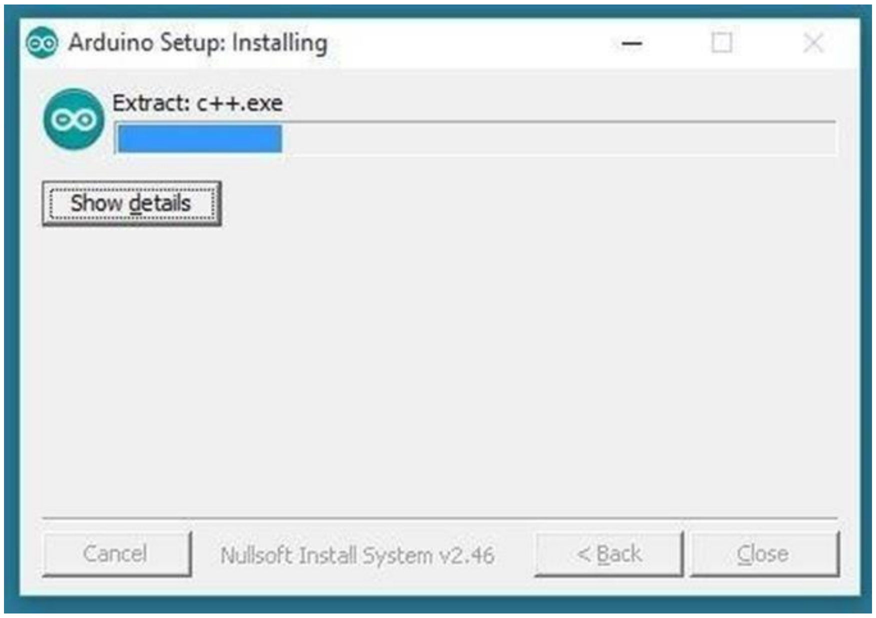
Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\Arduino\

Space required: 392.7MB

Space available: 24.6GB

- Choose the installation directory (we suggest to keep the default one)


Arduino Setup: Installing

Extract: c++.exe

- The process will extract and install all the required files to execute properly the Arduino Software (IDE)
- Proceed with board specific instructions
- When the Arduino Software (IDE) is properly installed you can go back to the IDE.

Different Arduino Boards:

Arduino USB

1. Arduino Uno

This is the latest revision of the basic Arduino USB board. It connects to the computer with a standard USB cable and contains everything else you need to program and use the board.

2. Arduino NG REV-C

Revision C of the Arduino NG does not have a built-in LED on pin 13 - instead you'll see two small unused solder pads near the labels "GND" and "13".

Arduino Bluetooth

The Arduino BT is a microcontroller board originally was based on the ATmega168, but now is supplied with the 328, and the Bluegiga WT11 bluetooth module. It supports wireless serial communication over bluetooth.

Arduino Mega

The original Arduino Mega has an ATmega1280 and an FTDI USB-to- serial chip.

Arduino NANO

The Arduino Nano 3.0 has an ATmega328 and a two-layer PCB. The power LED moved to the top of the board.

EXPERIMENT NO.2

AIM: To interface LED/Buzzer with Arduino/Raspberry Pi and write a program to turn ON LED for 1 sec after every 2 seconds.

OBJECTIVES: Student should get the knowledge of Arduino board and different types of LED.

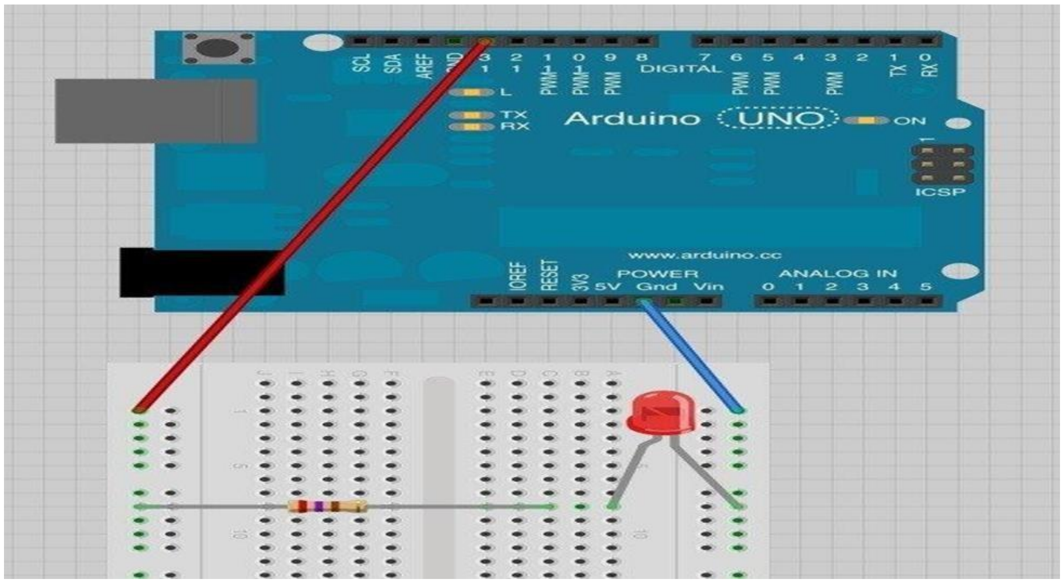
OUTCOMES: Student will be write program using Arduino ide for blink LED

HARDWARE REQUIREMENTS:

- **1X BREADBOARD**
- **1X ARDUINO UNO R3**
- **1X RGB LED**
- **1X 330Ω RESISTOR**
- **2X JUMPER WIRES BLINKING THE RGB LED**

With a simple modification of the breadboard, we could attach the led to an output pin of the Arduino. Move the red jumper wire from the ARDUINO 5v connector to

13, as shown below



now load the 'blink' example sketch from lesson 1. you will notice that both the built-in 'l' LED and the external LED should now blink.

/*

2. blink

3. turns on an led on for one second, then off for one second, repeatedly. 4.

5. this example code is in the public domain. 6.*/

7.

8. // pin 13 has an led connected on most arduino boards.

9. // give it a name:

10. int led = 13;

11.

```
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinmode(led, output);
}

// the loop routine runs over and over again forever:
void loop() {
digitalwrite(led, high);    // turn the led on (high is the voltage level)
delay(1000);// wait for a second
digitalwrite(led, low);     // turn the led off by making the voltage low
delay(1000);// wait for a second
}
```

lets try using a different pin of the Arduino – say D7. Move the red jumper lead from pin d13 to pin d7 and modify the following line near the top of the sketch:

```
int led = 13;
```

so that it reads:

```
int led = 7;
```

Upload the modified sketch to your Arduino board and the led should still be blinking, but this time using pin D7.

EXPERIMENT NO.3

AIM: Write program using Arduino IDE for Blink LED

OBJECTIVES: Student should get the knowledge of Arduino Board and different types of LED

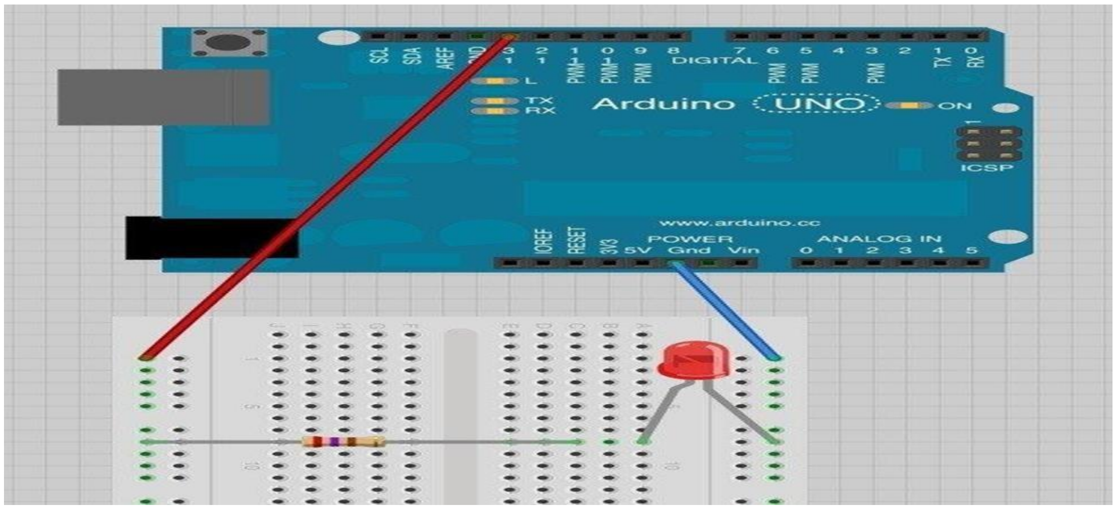
OUTCOMES: Student will be Write program using Arduino IDE for Blink LED

HARDWARE REQUIREMENTS:

- 1x Breadboard
- 1x Arduino Uno R3
- 1x RGB LED
- 1x 330Ω Resistor
- 2x Jumper Wires

Blinking the RGB LED

With a simple modification of the breadboard, we could attach the LED to an output pin of the Arduino. Move the red jumper wire from the Arduino 5V connector to D13, as shown below:



```
/*
  Blink
  Turns on an LED on for one second, then off for one second,
repeatedly.
  This example code is in the public
domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int
led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);

}

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

Lets try using a different pin of the Arduino – say D7. Move the red jumper lead from pin D13 to pin D7 and modify the following line near the top of the sketch:
1. int led = 13;

so that it reads:
1. int led = 7;

Upload the modified sketch to your Arduino board and the LED should still be blinking, but this time using pin D7.

EXPERIMENT NO.4

AIM: WRITE PROGRAM FOR RGB LED USING ARDUINO.

OBJECTIVES: student should get the knowledge of Arduino ide and RGB led

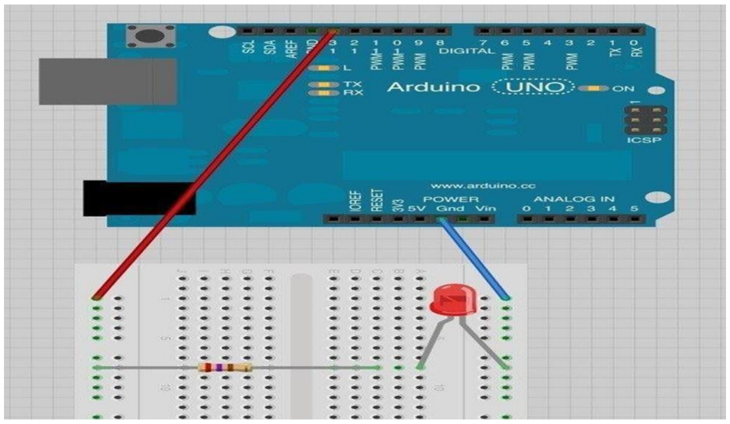
OUTCOMES: student will be developed programs using Arduino IDE and Arduino board for RGB LED

HARDWARE REQUIREMENTS:

- **1X BREADBOARD**
- **1X ARDUINO UNO R3**
- **1X LED**
- **1X 330Ω RESISTOR**
- **2X JUMPER WIRES**

BLINKING THE LED

With a simple modification of the breadboard, we could attach the led to an output pin of the Arduino. Move the red jumper wire from the Arduino 5v connector to D13, as shown below:



now load the 'blink' example sketch from lesson 1. You will notice that both the built-in 'l' led and the external led should now blink.

The following test sketch will cycle through the colors red, green, blue, yellow, purple, and aqua. These colors being some of the standard internet colors.

```
/*
adafruit arduino - lesson 3. rgb led
*/

int redPin = 11;
int greenPin = 10;
int bluePin = 9;

//uncomment this line if using a common anode  led
//#define common_anode
```

```

void setup()
{
    pinmode(redPin, output);
    pinmode(greenPin, output);
    pinmode(bluePin, output);
}

void loop()
{
    setcolor(255, 0, 0); //red
    delay(1000);
    setcolor(0, 255, 0); //green
    delay(1000);
    setcolor(0, 0, 255);// blue
    delay(1000);
    setcolor(255, 255, 0);      // yellow
    delay(1000);
    setcolor(80, 0, 80);// purple
    delay(1000);
    setcolor(0, 255, 255);      // aqua
    delay(1000);
}
```

```

void setcolor(int red, int green, int blue)
{
    #ifdef common_anode
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
    #endif
    analogwrite(redpin, red);
    analogwrite(greenpin, green);
    analogwrite(bluepin, blue);
}
```

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;
```

The next step is to write the 'setup' function. As we have learnt in earlier lessons, the setup function runs just once after the Arduino has reset. In this case, all it has to do is define the three pins we are using as being outputs.

```
VOID SETUP()
{
    PINMODE(REDPIN, OUTPUT);
    PINMODE(GREENPIN, OUTPUT);
    PINMODE(BLUEPIN, OUTPUT);
}
void setcolor(int red, int green, int blue) {
    analogwrite(redPin, red);
    analogwrite(greenPin, green);
    analogwrite(bluePin, blue); }
```

This function takes three arguments, one for the brightness of the red, green and blue LEDS. In each case the number will be in the range 0 to 255, where 0 means off and 255 means maximum brightness. The function then calls 'analogwrite' to set the brightness of each led.

if you look at the 'loop' function you can see that we are setting the amount of red, green and blue light that we want to display and then pausing for a second before moving on to the next color.

```
void loop()
{
    setcolor(255, 0, 0); //red
    delay(1000);
    setcolor(0, 255, 0); //green
    delay(1000);
    setcolor(0, 0, 255);// blue
delay(1000);
setcolor(255, 255, 0);// yellow
delay(1000);
setcolor(80, 0, 80);// purple
delay(1000);
setcolor(0, 255, 255);// aqua
delay(1000);
}
```

Try adding a few colors of your own to the sketch and watch the effect on your LED.

EXPERIMENT NO.5

AIM: Study the temperature sensor and write program foe monitor temperature using Arduino.

OBJECTIVES: student should get the knowledge of temperature sensor

OUTCOMES: student will be developed programs using Arduino ide and Arduino board for temperature sensor

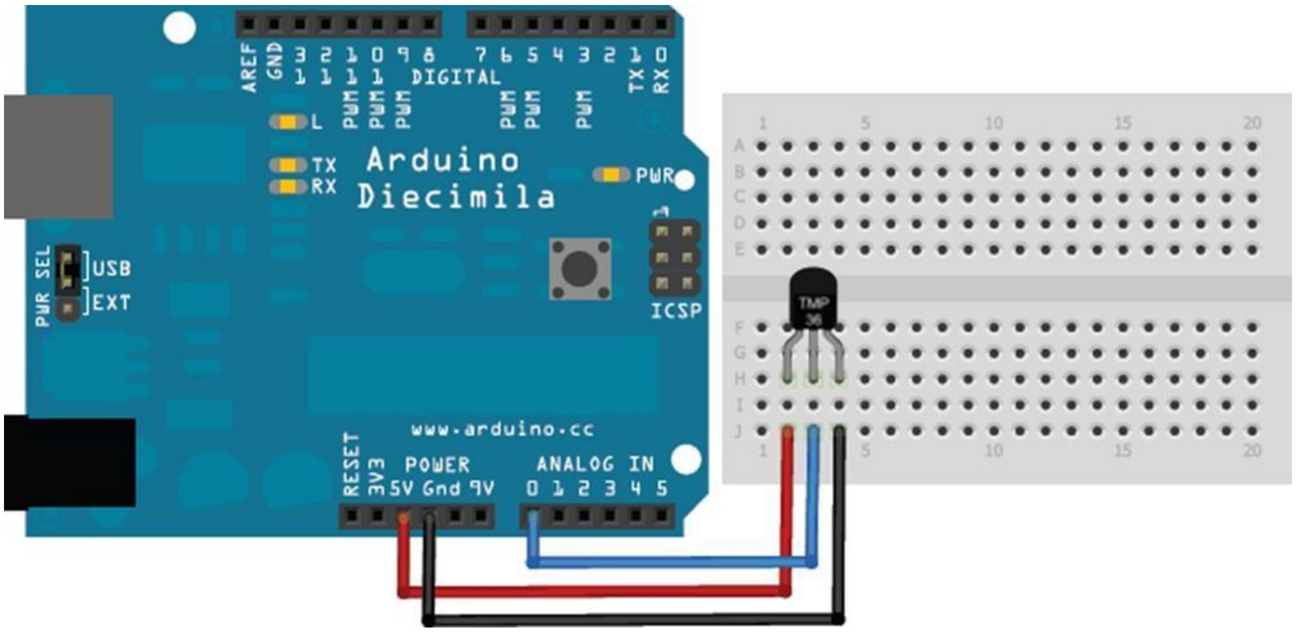
CONNECTING TO A TEMPERATURE SENSOR

These sensors have little chips in them and while they're not that delicate, they do need to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5v dc - so don't try to use a 9v battery!

They come in a "to-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard. You can also solder to the pins to connect long wires. If you need to waterproof the sensor, you can see below for an instructable for how to make an excellent case.

READING THE ANALOG TEMPERATURE DATA

Unlike the FSR or photocell sensors we have looked at, the TMP36 and friends doesn't act like a resistor. Because of that, there is really only one way to read the temperature value from the sensor, and that is plugging the output pin directly into an analog (ADC) input.



Remember that you can use anywhere between 2.7v and 5.5v as the power supply. for this example i'm showing it with a 5v supply but note that you can use this with a 3.3v supply just as easily. no matter what supply you use, the analog voltage reading will range from about 0v (ground) to about 1.75v.

if you're using a 5v arduino, and connecting the sensor directly into an analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:

VOLTAGE AT PIN IN MILLIVOLTS = (reading from adc) * (5000/1024)

this formula converts the number 0-1023 from the adc into 0-5000mv (= 5v) if you're using a 3.3v arduino, you'll want to use this:

VOLTAGE AT PIN IN MILLIVOLTS = (reading from adc) * (3300/1024)

this formula converts the number 0-1023 from the adc into 0-3300mv (= 3.3v)

then, to convert millivolts into temperature, use this formula:

CENTIGRADE TEMPERATURE = [(analog voltage in mv) - 500] / 10

simple thermometer

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

```
1.      //TMP36 Pin Variables
2.      int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is
connected to
3.      //the resolution is 10 mV / degree centigrade with a
4.      //500 mV offset to allow for negative temperatures 5.
6. /*
7.      * setup() - this function runs once when you turn your Arduino on
8.      * We initialize the serial connection with the computer 9. */
10. void setup()
11.{
12.     Serial.begin(9600);       //Start the serial connection with the computer
13.     //to view the result open the serial monitor
14.}
15.
16. void loop()     // run over and over again
17.{
18.     //getting the voltage reading from the temperature sensor
19.	int reading = analogRead(sensorPin); 20.
21.	// converting that reading to voltage, for 3.3v arduino use 3.3
22.	float voltage = reading * 5.0; 23. voltage /= 1024.0;
24.
25.	// print out the voltage
26.	Serial.print(voltage); Serial.println(" volts"); 27.
28.	// now print out the temperature
29.	float temperatureC = (voltage - 0.5) * 100 ; //converting from 10 mv per
degree wit 500 mV offset
30.	//to degrees ((voltage - 500mV)
times 100)
31.	Serial.print(temperatureC); Serial.println(" degrees C"); 32.
33.	// now convert to Fahrenheit
34.	float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
35.	Serial.print(temperatureF); Serial.println(" degrees F"); 36.
37. delay(1000);
38.}
```

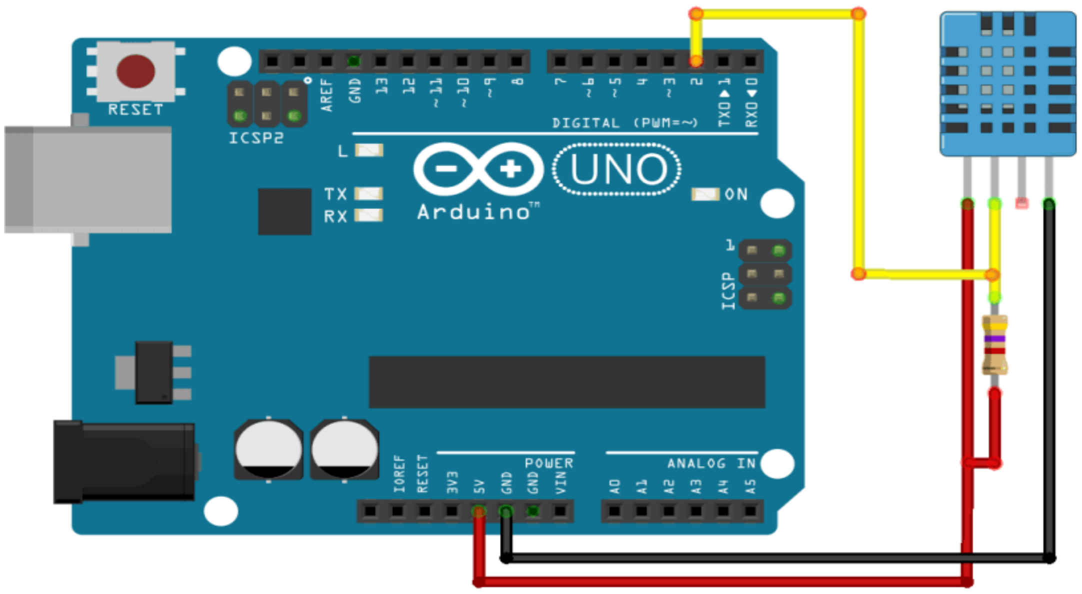
EXPERIMENT NO.6

AIM: To interface DHT11 sensor with Arduino/Raspberry Pi and write a program to print temperature and humidity readings.

OBJECTIVES: student should get the knowledge of DHT11 temperature sensor

OUTCOMES: student will be developed programs using Arduino ide and Arduino board for temperature sensor

SCHEMATIC:



Follow the next schematic diagram to wire the DHT11 (or DHT22) temperature a nd humidity sensor to the Arduino.

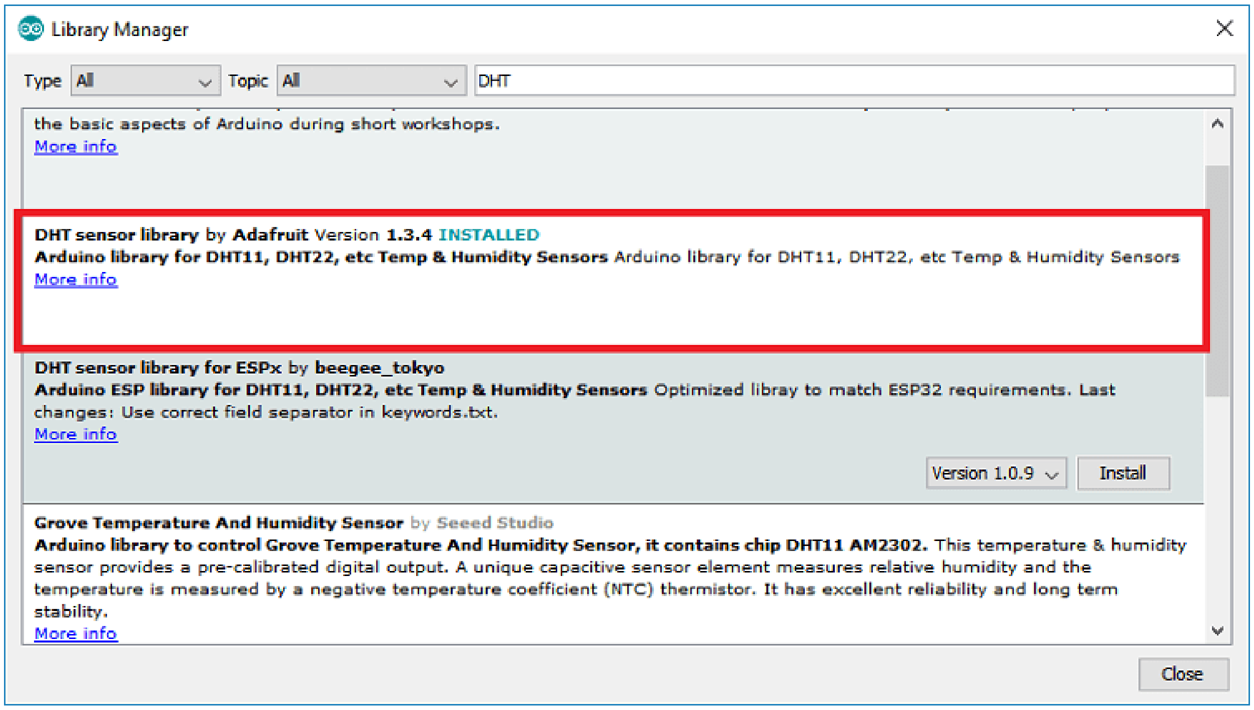
DHT Pin	Arduino
Pin 1	5V
Pin 2	D2 or any other digital pin
Pin 3	don’t connect
Pin 4	GND

Installing Libraries

To read from the DHT sensor, we’ll use the DHT library from Adafruit. To use this library you also need to install the Adafruit Unified Sensor library. Follow the next steps to install those libraries.

Open your Arduino IDE and go to Sketch > Include Library > Manage Libraries. The Library Manager should open.

Search for “DHT” on the Search box and install the DHT library from Adafruit.



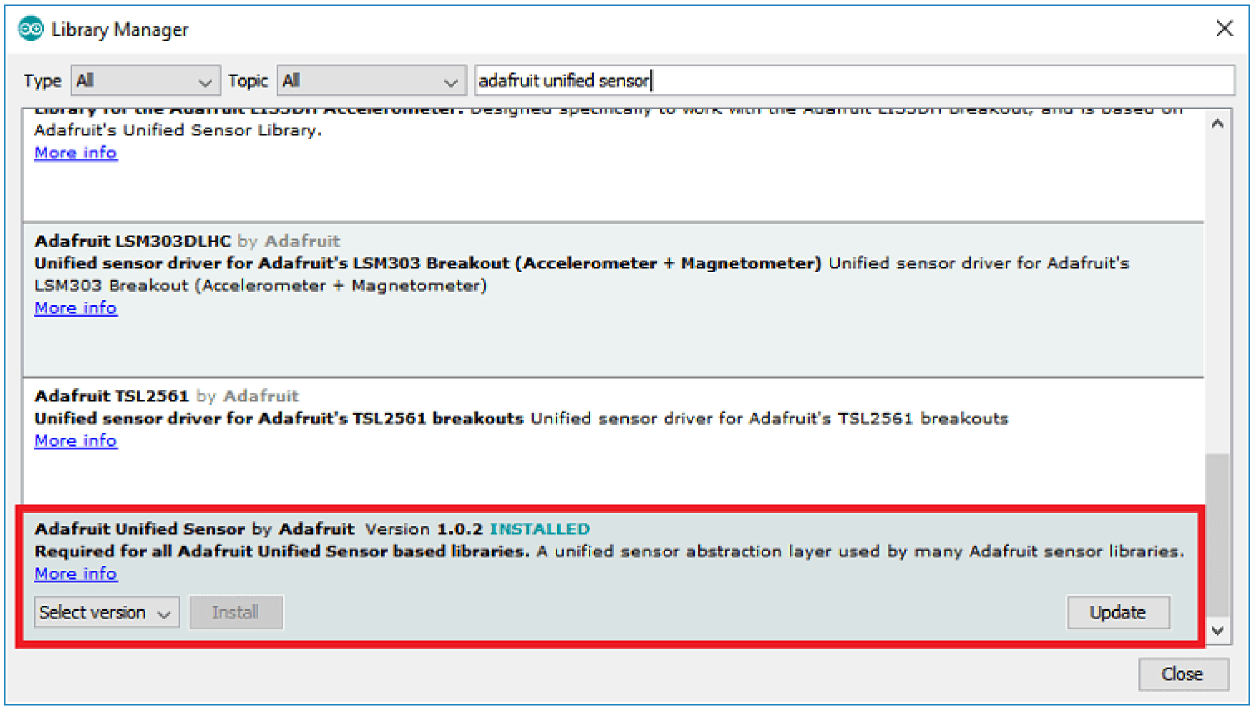
the basic aspects of Arduino during short workshops.
More info

DHT sensor library by **Adafruit** Version **1.3.4** **INSTALLED**
Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors Arduino library for DHT11, DHT22, etc Temp & Humidity Sensors
More info

DHT sensor library for ESPx by **beegee_tokyo**
Arduino ESP library for DHT11, DHT22, etc Temp & Humidity Sensors Optimized libray to match ESP32 requirements. Last changes: Use correct field separator in keywords.txt.
More info

Grove Temperature And Humidity Sensor by Seeed Studio
Arduino library to control Grove Temperature And Humidity Sensor, it contains chip DHT11 AM2302. This temperature & humidity sensor provides a pre-calibrated digital output. A unique capacitive sensor element measures relative humidity and the temperature is measured by a negative temperature coefficient (NTC) thermistor. It has excellent reliability and long term stability.
More info

After installing the DHT library from Adafruit, type “Adafruit Unified Sensor” in the search box. Scroll all the way down to find the library and install it.



Library for the Adafruit LIS3DH Accelerometer. Designed specifically to work with the Adafruit LIS3DH breakout, and is based on Adafruit's Unified Sensor Library.
More info

Adafruit LSM303DLHC by Adafruit
Unified sensor driver for Adafruit's LSM303 Breakout (Accelerometer + Magnetometer) Unified sensor driver for Adafruit's LSM303 Breakout (Accelerometer + Magnetometer)
More info

Adafruit TSL2561 by Adafruit
Unified sensor driver for Adafruit's TSL2561 breakouts Unified sensor driver for Adafruit's TSL2561 breakouts
More info

Adafruit Unified Sensor by **Adafruit** Version **1.0.2** INSTALLED
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer used by many Adafruit sensor libraries.
More info

Code

After installing the necessary libraries, you can upload an example code from the library.
In your Arduino IDE, go to **File** > **Examples** > **DHT Sensor library** > **DHTtester**
The following code should load. It reads temperature and humidity, and displays the results in the Serial Monitor.

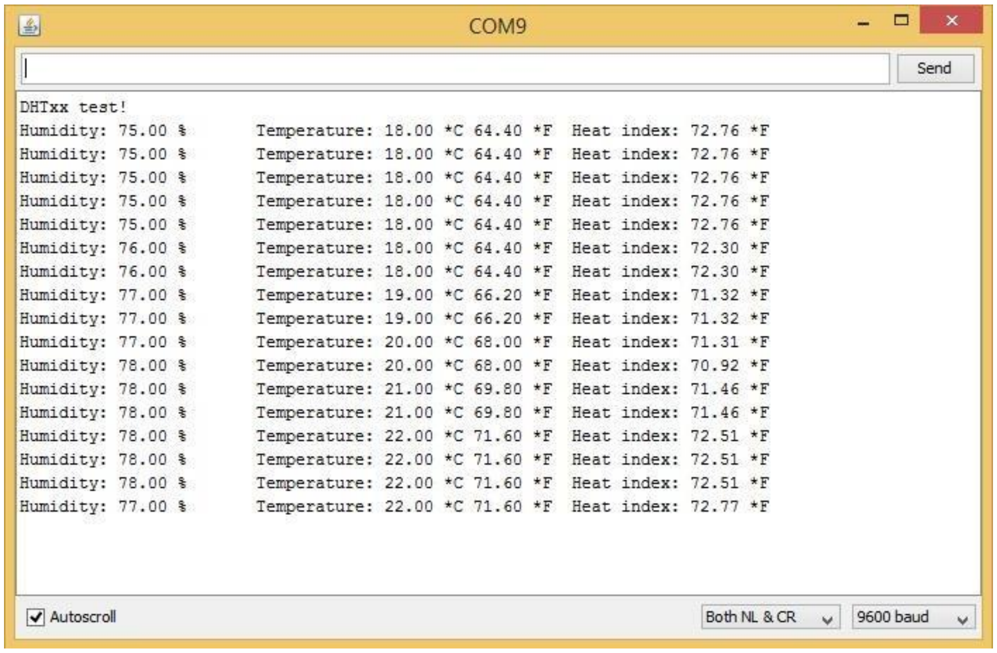
```
 // Example testing sketch for various DHT humidity/temperature sensors
// Written by ladyada, public domain

#include "DHT.h"

#define DHTPIN 2 // what pin we're connected to

// Uncomment whatever type you're using!
```

```
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)

// Initialize DHT sensor for normal 16mhz Arduino
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
  }

  // Compute heat index
  // Must send in temp in Fahrenheit!
```

```
float hi = dht.computeHeatIndex(f, h);

Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *F\t");
Serial.print("Heat index: ");
Serial.print(hi);
Serial.println(" *F");
```

Demonstration

After uploading the code to the Arduino, open the Serial Monitor at a baud rate of 9600. You should get sensor readings every two seconds. Here's what you should see in your Arduino IDE Serial Monitor.



```
DHTxx test!
Humidity: 75.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.76 *F
Humidity: 75.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.76 *F
Humidity: 75.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.76 *F
Humidity: 75.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.76 *F
Humidity: 75.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.76 *F
Humidity: 76.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.30 *F
Humidity: 76.00 %	Temperature: 18.00 *C 64.40 *F	Heat index: 72.30 *F
Humidity: 77.00 %	Temperature: 19.00 *C 66.20 *F	Heat index: 71.32 *F
Humidity: 77.00 %	Temperature: 19.00 *C 66.20 *F	Heat index: 71.32 *F
Humidity: 77.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 71.31 *F
Humidity: 78.00 %	Temperature: 20.00 *C 68.00 *F	Heat index: 70.92 *F
Humidity: 78.00 %	Temperature: 21.00 *C 69.80 *F	Heat index: 71.46 *F
Humidity: 78.00 %	Temperature: 21.00 *C 69.80 *F	Heat index: 71.46 *F
Humidity: 78.00 %	Temperature: 22.00 *C 71.60 *F	Heat index: 72.51 *F
Humidity: 78.00 %	Temperature: 22.00 *C 71.60 *F	Heat index: 72.51 *F
Humidity: 78.00 %	Temperature: 22.00 *C 71.60 *F	Heat index: 72.51 *F
Humidity: 77.00 %	Temperature: 22.00 *C 71.60 *F	Heat index: 72.77 *F
```

EXPERIMENT NO.7

AIM: To interface motor using relay with arduino/raspberry pi and write a program to turn on motor when push button is pressed.

OBJECTIVES: student should get the knowledge of Relay & motor.

OUTCOMES: student will be developed programs using Arduino IDE and Arduino board Relay and Motor.

Hardware Required

1	×	Arduino UNO
1	×	USB 2.0 cable type A/B
1	×	Button
1	×	Relay
1	×	12V Power Adapter
1	×	Breadboard
1	×	Jumper Wires

```
/*
 * Code for Relay & Motor

 */

const int BUTTON_PIN = 7; // Arduino pin connected to button's pin
const int RELAY_PIN = 3; // Arduino pin connected to relay's pin

void setup() {
  Serial.begin(9600);              // initialize serial
  pinMode(BUTTON_PIN, INPUT_PULLUP); // set arduino pin to input pull-up
mode
  pinMode(RELAY_PIN, OUTPUT);        // set arduino pin to output mode
}

void loop() {
  int buttonState = digitalRead(BUTTON_PIN); // read new state

  if (buttonState == LOW) {
    Serial.println("The button is being pressed");
    digitalWrite(RELAY_PIN, HIGH); // turn on
  }
  else
  if (buttonState == HIGH) {
    Serial.println("The button is unpressed");
    digitalWrite(RELAY_PIN, LOW); // turn off
  }
}
```

EXPERIMENT NO.8

AIM: To interface OLED with Arduino/Raspberry Pi and write a program to print temperature and humidity readings on it.

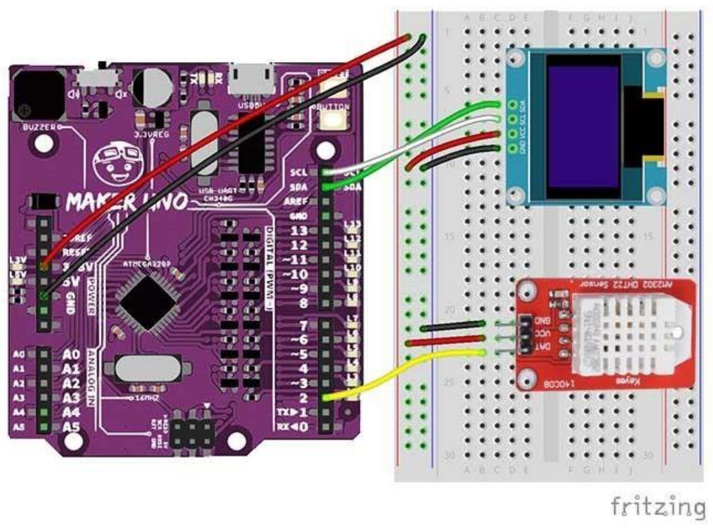
OBJECTIVES: student should get the knowledge of Relay & motor.

OUTCOMES: student will be developed programs using Arduino IDE and Arduino board Relay and Motor.

Hardware Required

Introduction

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed). It’s easy to use, but requires careful timing to grab data. The only real downside of this sensor is you can only get new data from it once every 2 seconds, so when using our library, sensor readings can be up to 2 second



```
#include <SPI.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include "DHT.h"


#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 64 // OLED display height, in pixels


// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

#define OLED_RESET     4 // Reset pin # (or -1 if sharing Arduino reset pin)

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


#define DHTPIN 2

#define DHTTYPE DHT22   // DHT 22 (AM2302), AM2321

DHT dht(DHTPIN, DHTTYPE);


#define PIEZO 8
```

```
#define ALARM  1000

int Sound[] = {ALARM, ALARM};
int SoundNoteDurations[] = {4, 4};

#define playSound() playMelody(Sound, SoundNoteDurations, 2)

char inChar;
String inString;

void setup() {

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  dht.begin();

}
```

```
void loop() {

  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
  }
  // Compute heat index in Fahrenheit (the default)
  float hif = dht.computeHeatIndex(f, h);
  // Compute heat index in Celsius (isFahreheit = false)
  float hic = dht.computeHeatIndex(t, h, false);

  if (hic >= 41) {
   displayOled();
   playSound();
  }
  else {
    displayOled();
    noTone(PIEZO);
  }



}

void displayOled() {
```

```
// Read humidity
float h = dht.readHumidity();
// Read temperature as Celsius (the default)
float t = dht.readTemperature();
// Read temperature as Fahrenheit (isFahrenheit = true)
float f = dht.readTemperature(true);
// Compute heat index in Fahrenheit (the default)
float hif = dht.computeHeatIndex(f, h);
// Compute heat index in Celsius (isFahreheit = false)
float hic = dht.computeHeatIndex(t, h, false);

display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(5, 15);
display.print("Humidity:");
display.setCursor(80, 15);
display.print(h);
display.print("%");
display.setCursor(5, 30);
```

```
  display.print("Temperature:");
  display.setCursor(80, 30);
  display.print(t);
  display.print((char)247); // degree symbol
  display.print("C");
  display.setCursor(5, 45);
  display.print("Heat Index:");
  display.setCursor(80, 45);
  display.print(hic);
  display.print((char)247); // degree symbol
  display.print("C");
  display.display();

}

void playMelody(int *melody, int *noteDurations, int notesLength)
{
  pinMode(PIEZO, OUTPUT);

  for (int thisNote = 0; thisNote < notesLength; thisNote++) {
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(PIEZO, melody[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(PIEZO);
  }
}
```

EXPERIMENT NO.9

AIM: Study and Implement MQTT Protocol using Arduino.

OBJECTIVES: Student should get the knowledge of MQTT Protocol using Arduino.

OUTCOMES: Student will be developed programs using Arduino IDE and Arduino Board for MQTT Protocol

MQTT:

MQ Telemetry Transport (MQTT) is an open source protocol for constrained devices and low-bandwidth, high-latency networks. It is a publish/subscribe messaging transport that is extremely lightweight and ideal for connecting small devices to constrained networks.

MQTT is bandwidth efficient, data agnostic, and has continuous session awareness. It helps minimize the resource requirements for your IoT device, while also attempting to ensure reliability and some degree of assurance of delivery with grades of service. MQTT targets large networks of small devices that need to be monitored or controlled from a back-end server on the Internet. It is not designed for device-to-device transfer. Nor is it designed to "multicast" data to many receivers. MQTT is extremely simple, offering few control options.

MQTT METHODS

MQTT defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

CONNECT

Waits for a connection to be established with the server.

DISCONNECT

Waits for the MQTT client to finish any work it must do, and for the TCP/IP session to disconnect.

SUBSCRIBE

Waits for completion of the Subscribe or Unsubscribe method.

UNSUBSCRIBE

Requests the server unsubscribe the client from one or more topics.

PUBLISH

Returns immediately to the application thread after passing the request to the MQTT client.

EXPERIMENT NO.10

AIM: Study and Configure Raspberry Pi.

OBJECTIVES: Student should get the knowledge of Raspberry Pi.

OUTCOMES: Student will be get knowledge of Raspberry Pi

RASPBERRY PI

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundationto promote the teaching of basic computer science in schools and in developing countries.[4][5][6] The original model became far more popular than anticipated, selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. Some accessories however have been included in several official and unofficial bundles.

According to the Raspberry Pi Foundation, over 5 million Raspberry Pis have been sold before February 2015, making it the best-selling British computer.[8] By November 2016 they had sold 11 million units[9][10], reaching 12.5m in March 2017, making it the third best-selling "general purpose computer" ever.

To get started with Raspberry Pi, you need an operating system. NOOBS (New Out Of Box Software) is an easy operating system install manager for the Raspberry Pi.

HOW TO GET AND INSTALL NOOBS

DOWNLOAD NOOBS OS FROM

We recommend using an SD card with a minimum capacity of 8GB.

1. GO to the https://www.raspberrypi.org/downloads/

2. Click on NOOBS, then click on the Download ZIP button under ‘NOOBS (offline and network install)’, and select a folder to save it to.
3. Extract the files from the zip.

FORMAT YOUR SD CARD

It is best to format your SD card before copying the NOOBS files onto it. To do this:

1. Download SD Formatter 4.0 for either Windows or Mac.
2. Follow the instructions to install the software.

3. Insert your SD card into the computer or laptop's SD card reader and make a note of the drive letter allocated to it, e.g. G:/

4. In SD Formatter, select the drive letter for your SD card and format it.

DRAG AND DROP NOOBS FILES

1. Once your SD card has been formatted, drag all the files in the extracted NOOBS folder and drop them onto the SD card drive.

2. The necessary files will then be transferred to your SD card.

3. When this process has finished, safely remove the SD card and insert it into your Raspberry Pi.

FIRST BOOT

1. Plug in your keyboard, mouse, and monitor cables.

2. Now plug the USB power cable into your Pi.

3. Your Raspberry Pi will boot, and a window will appear with a list of different operating systems that you can install. We recommend that you use Raspbian – tick the box next to Raspbian and click on Install.

4. Raspbian will then run through its installation process. Note that this can take a while.

5. When the install process has completed, the Raspberry Pi configuration menu (raspi- config) will load. Here you are able to set the time and date for your region, enable a Raspberry Pi camera board, or even create users. You can exit this menu by using Tab on your keyboard to move to Finish.

LOGGING IN AND ACCESSING THE GRAPHICAL USER INTERFACE

The default login for Raspbian is username pi with the password raspberry. Note that you will not see any writing appear when you type the password. This is a security feature in Linux.

To load the graphical user interface, type startx and press Enter.