DHANALAKSHMISRINIVASAN COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE & Affiliated to Anna University, Chennai)
East Coast Road, Mamallapuram, Chennai–603104

DEPARTMENT OF INFORMATION TECHNOLOGY

CS3691-EMBEDDED SYSTEM AND IOT LABORATORY

RECORD NOTE BOOK

Name .....
Reg.No .....
Year/Semester .....

DHANALAKSHMI SRINIVASAN
COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE & Affiliated to Anna University,Chennai)

East Coast Road, Mamallapuram, Chennai - 603 104

Name .....

Reg. No .....

Department .....

Semester .....

Subject Name .....

Subject Code .....

Certified that this is the Bonafide record of the Practicals done for the above subject during the period .....
in the academic year 20 - 20

Staff in Charge

Head of the Department

submitted for the University Practical Examination held on .....

Internal Examiner

External Examiner

Course Objectives

CLO1: To learn the internal architecture and programming of an embedded processor.

CLO2: To introduce interfacing, I/O devices to the processor.

CLO3: To introduce the evolution of the Internet of Things (IoT).

CLO4: To build a small low-cost embedded IoT system using Arduino/Raspberry Pi/ open platform.

CLO5: To apply the concept of the Internet of Things in real-world scenarios.

LIST OF EXPERIMENTS

1. Write 8051 Assembly Language experiments using a simulator
2. Test data transfer between registers and memory.
3. Perform ALU operations.
4. Write Basic and arithmetic Programs Using Embedded C.
5. Introduction to Arduino platform and programming
6. Explore different communication methods with IoT devices (Zigbee, GSM,Bluetooth)
7. Introduction to Raspberry PI platform and Python programming
8. Interfacing sensors with Raspberry PI
9. Communicate between Arduino and Raspberry PI using any wireless medium
10. Setup a cloud platform to log the data
11. Log Data using Raspberry PI and upload to the cloud platform
12. Design an IOT-based system

Course Outcomes

CO1: Explain the architecture of embedded processors.

CO2: Write embedded C programs.

CO3: Design simple embedded applications.

CO4: Compare the communication models in IOT

CO5: Design IoT applications using Arduino/Raspberry Pi /open platform

INDEX

Exp.No	Date	Name of the experiment	Viva	Signature

Exp.NO.1

DATE:

Introduction to 8051 Assembly Language Programming

Objective:

To learn how to write an 8051 Assembly Language program and observe Port 1 of the microcontroller on and off at a specific frequency.

Materials:

- General Purpose Computer
- 8051 microcontroller simulator software (e.g., Keil uVision5)

Procedure:

- Setting Up the Simulator Environment
- Open the simulator software (e.g., Keil uVision).
- Create a new project or open a new blank workspace.

Writing the Assembly Code:

```
ORG 0x0000          ; Start of program memory
MAIN:
   MOV P1, #0x00    ; Initialize Port 1 as low
   ACALL DELAY      ; Call delay subroutine
   MOV P1, #0xFF    ; Turn on LED (Port 1 high)
   ACALL DELAY      ; Call delay subroutine
   SJMP MAIN        ; Jump back to MAIN

DELAY:
   MOV R2, #0x64    ; Load R2 with a value for delay

DELAY_LOOP:
   DJNZ R2, DELAY_LOOP; Decrement R2 and loop if not zero
   RET
```

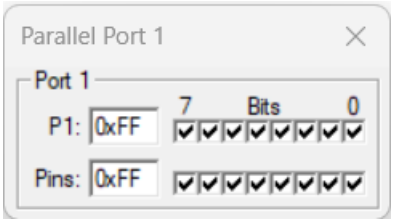
(In this code, P1 represents Port 1, and R2 is used for delay.)

Compiling and Running the Code:

- Compile the code to generate the machine code.

- Load the machine code onto the microcontroller in the simulator.
- Start the simulation.

Output:



Observation and Analysis:

Observed Port 1 for on and off operation at a specific frequency.

Result:

Thus, the 8051 Assembly Language program was written and observed Port 1 of the microcontroller on and off at a specific frequency.

Exp.NO.2

DATE:

Data Transfer between Registers and Memory

Objective:

Understand data transfer between registers and memory using load and store operations.

Materials:

- General Purpose Computer
- 8051 microcontroller simulator software (e.g., Keil uVision5)

Procedure:

- Setting Up the Simulator Environment:
- Open the simulator software.
- Create a new project or open a new blank workspace.

Writing the Assembly Code:

```
ORG 0x0000; Start of program memory

MAIN:
   MOV A, #0xAA    ; Load accumulator A with a test value
   MOV R0, #0x10    ; Load register R0 with a memory address
    MOV A,R0          ; Store the value from accumulator A into memory location   pointed to by R0
   MOV @R0, A       ; Store A into memory location pointed by R0
   MOV A, #0x00      ; Clear accumulator A
   MOV A, @R0       ; Load A with the value from memory location pointed to   by R0
   MOV B, A          ; Transfer the value from accumulator A to register
   MOV R1, #0x20     ; Load register R1 with another memory address
   MOV @R1, B        ; Store B in memory location pointed by R1
   MOV A, @R1 :    ; Load the value from memory location pointed to by R1 into accumulator A
   SJMP MAIN         ; Infinite loop
   END
```

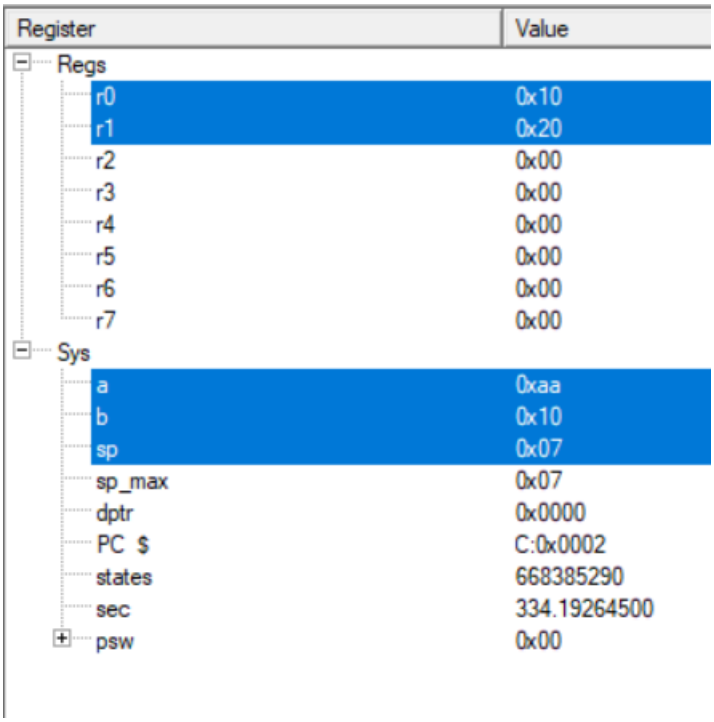
(This code demonstrates data transfer between a register and memory locations using load and store operations.)

Compiling and Running the Code:

- Compile the code to generate the machine code.

- Load the machine code onto the microcontroller in the simulator.
- Start the simulation.

Output:



Observation and Analysis:

- Observed the execution of load and store operations between registers and memory locations.

Result:

Thus, the program to move the value between registers and memory with 8051 microcontrollers has been written and executed.

Exp. NO.3

DATE:

ALU Operations

Objective:

Perform ALU operations like addition, subtraction, and logical operations between two registers**.**

Materials:

- General Purpose Computer
- 8051 microcontroller simulator software (e.g., Keil uVision5)

Procedure:

- Setting Up the Simulator Environment:
- Open the simulator software.
- Create a new project or open a new blank workspace.

Writing the Assembly Code:

```
ORG 0x0000  ; Start of program memory
MAIN:
   MOV A, #0x08   ; Load value 8 into accumulator A
   MOV B, #0x05   ; Load value 5 into register B
   ; Perform addition: A = A + B
   ADD A, B
    MOV R0, A
    MOV A, #0x08   ; Load value 8 into accumulator A
   MOV B, #0x05   ; Load value 5 into register B
   ; Perform subtraction: A = A - B
   SUBB A, B
    MOV R1, A
    MOV A, #0x08   ; Load value 8 into accumulator A
   MOV B, #0x05   ; Load value 5 into register B
   ANL A, B          ; Perform bitwise AND operation: A = A & B
    MOV R2, A
   MOV A, #0x08   ; Load value 8 into accumulator A
   MOV B, #0x05   ; Load value 5 into register B
   ; Perform bitwise OR operation: A = A | B
```

```
    ORL A, B
    MOV R3, A
    MOV A, #0x08   ; Load value 8 into accumulator A
    MOV B, #0x05   ; Load value 5 into register B
    ; Perform bitwise XOR operation: A = A ^ B
    XRL A, B
    MOV R4, A
    SJMP MAIN      ; Infinite loop
END
```

(This code performs various ALU operations on registers A and B.)

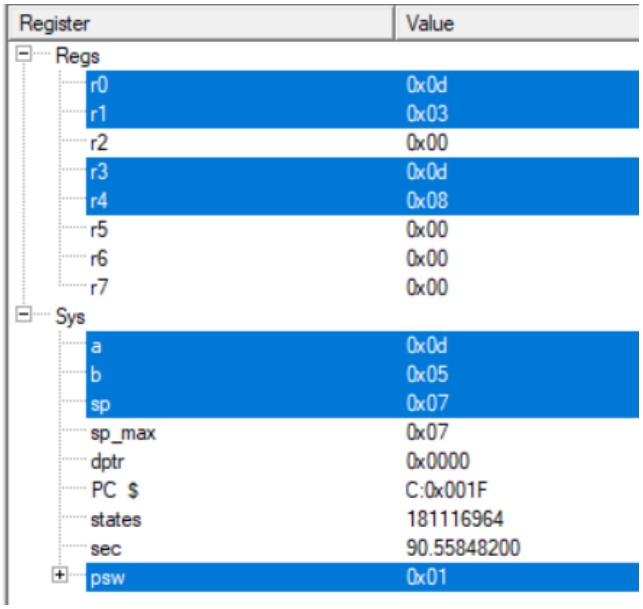
Compiling and Running the Code:

- Compile the code to generate the machine code.
- Load the machine code onto the microcontroller in the simulator.
- Start the simulation.

Output:

Test inputs: Input 1: 08

Input2: 05



Observation and Analysis:

- Observed the effects of each ALU operation on the values of registers A and B.

Result:

Thus, the program to study the arithmetic operations of 8051 microcontrollers has been written and executed.

Exp. NO.4

DATE:

Introduction to 8051 embedded C Programming & ALU Operations

Objective:

To perform basic and ALU operations using embedded C Programming**.**

Materials Required:

- General Purpose Computer
- 8051 microcontroller simulator software (e.g., Keil uVision5)

Procedure:

- Setting Up the Simulator Environment:
- Open the simulator software.
- Create a new project or open a new blank workspace.

(a) Writing the Code for blinking an LED:

```
#include <REG51.h>  // Include the AT89C51 register definitions
// Define constants for LED port and delay
#define LED_PORT P1   // Assuming the LED is connected to Port 1
#define DELAY_MS 500  // Delay in milliseconds
void delay_ms(unsigned int ms) {
   unsigned int i, j;
   for (i = 0; i < ms; i++) {
      for (j = 0; j < 120; j++) {
         // Adjust this loop for the desired delay
      }
   }
}
void main() {
   unsigned char portStatus = 0xFF;  // Initialize portStatus to all 1s (all LEDs off)
   // Initialize the LED port as an output
   LED_PORT = portStatus;
   while (1) {
      // Toggle the LEDs
      portStatus = ~portStatus;
```

```
        LED_PORT = portStatus;

        // Delay for a while
        delay_ms(DELAY_MS);
        // View the output status of the port
        // You can use a logic analyzer or an external device to view the port status
    }
}
```

(b) Writing the Code for ALU operations

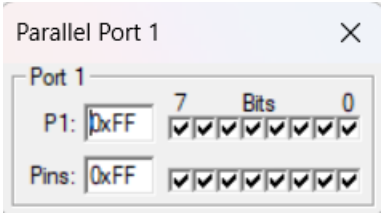
```
#include <REG51.h>
// Function to perform addition
unsigned char add(unsigned char a, unsigned char b)
{
    return a + b;
}
// Function to perform subtraction
unsigned char subtract(unsigned char a, unsigned char b)
 {
    return a - b;
}
void main()
{
    unsigned char num1, num2, result;
    // Input two numbers
    num1 = 0x0A; // Example values (you can change these)
    num2 = 0x05; // Example values (you can change these)
    // Perform addition using the ALU
    result = add(num1, num2);
    // Perform subtraction using the ALU
    result = subtract(num1, num2);
    // The result can be used as needed
    while (1)
{
        // Your main program loop
    }
}
```

Compiling and Running the Code:

- Compile the code to generate the machine code.
- Load the machine code onto the microcontroller in the simulator.
- Start the simulation.

Output:

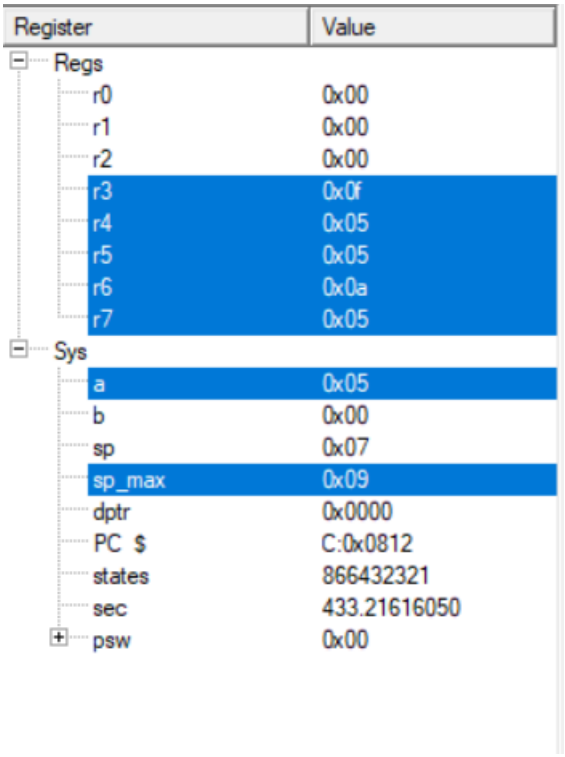
(a)



(b)

Test inputs: num1=0A

num2=05



Observation and Analysis:

- Observed the effects of each ALU operation on the values of a and b.

Result

Thus, the programs to study the blinking of LED and arithmetic operations of 8051 microcontrollers using embedded C have been written and executed.

Exp. NO.5

DATE:

Introduction to Arduino platform and programming

Objective:

To introduce basic Arduino programming by creating a simple program that blinks an LED connected to the Arduino board.

Materials Required:

- Arduino board (e.g., Arduino Uno)
- USB cable for connecting Arduino to the computer
- 220-ohm resistor
- LED (any colour)
- Breadboard
- Jumper wires

Theory:

Arduino is a software as well as hardware platform that helps in making electronic projects. It is an open-source platform and has a variety of controllers and microprocessors. There are various types of Arduino boards used for various purposes.

It also provides an IDE (Integrated Development Environment) project, which is based on the Processing Language to upload the code to the physical board.

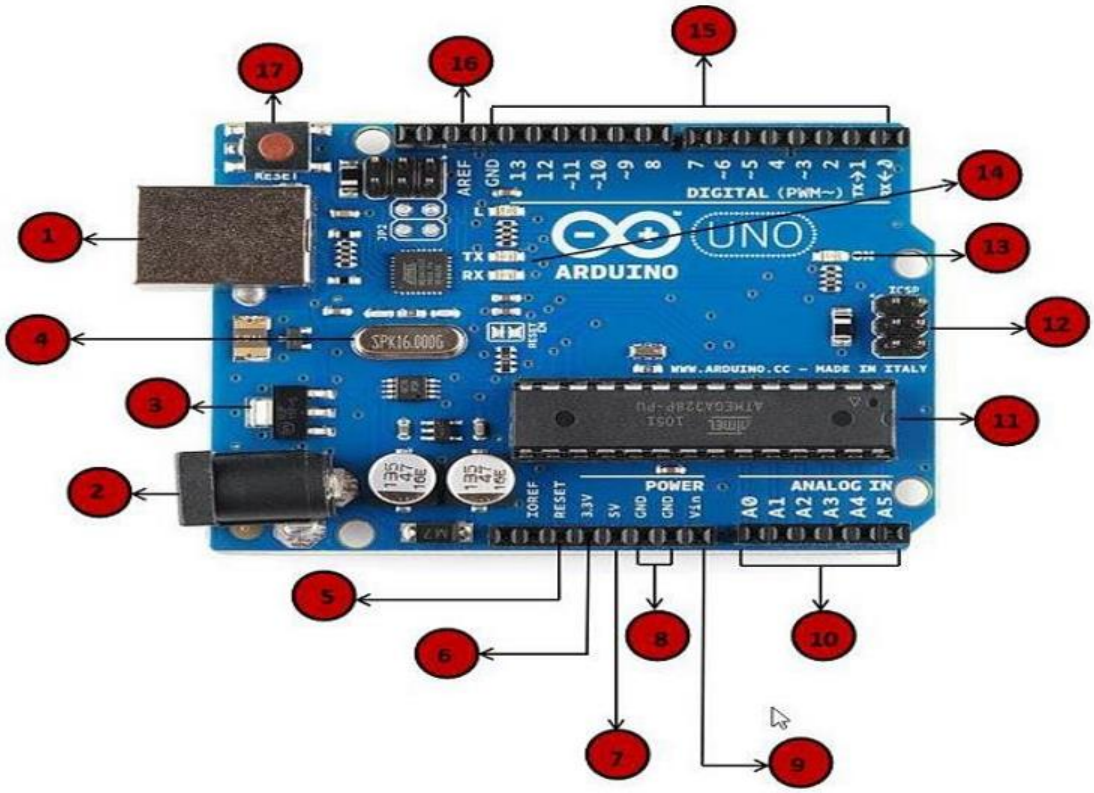
The Arduino is a single circuit board, which consists of different interfaces or parts. The board consists of a set of digital and analog pins that are used to connect various devices and components, which we want to use for the functioning of the electronic devices.

The Arduino board consists of sets of analog and digital I/O (Input / Output) pins, which are further interfaced to breadboard, expansion boards, and other circuits. Such boards feature the model, Universal Serial Bus (USB), and serial communication interfaces, which are used for loading programs from the computers.

The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as a powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog

Input/Output pins (I/O), shields, and other circuits.



1	**Power USB** Arduino board can be powered by using the USB cable from your computer.
2	**Power (Barrel Jack)** Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).
3	**Voltage Regulator** The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
4	**Crystal Oscillator** **The crystal oscillator helps Arduino in dealing with time issues. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.**
5,17	**Arduino Reset** **You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).**
	Pins (3.3, 5, GND, Vin) • **3.3V (6) − Supply 3.3 output volt** • **5V (7) − Supply 5 output volt** • **Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.**

6,7 8,9	• **GND (8) (Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit.** • **Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.**
10	**Analog pins** **The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.**
11	**Main microcontroller** **Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC..**
12	**ICSP pin** **Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output.**
13	**Power LED indicator** **This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.**

Procedure:

1. Setting Up the Hardware:
2. Place the Arduino board on a stable surface.
3. Insert the LED into the breadboard. Connect the longer leg (anode) of the LED to one end of the 220-ohm resistor.
4. Connect the other end of the resistor to one of the digital pins on the Arduino board (e.g., pin 7).
5. Connect the shorter leg (cathode) of the LED to the ground (GND) pin on the Arduino board.
6. Ensure all connections are secure.
7. Connecting Arduino to Your Computer:
8. Use a USB cable to connect your Arduino board to your computer.
9. Writing the Arduino Sketch (Program):

10. Launch the Arduino IDE on your computer.
11. In the Arduino IDE, click on File > New to create a new sketch.
12. Replace the default code with the following LED blinking program:
13. Make sure you have selected the correct Arduino board and port under Tools > Board and Tools > Port.
14. Click the right arrow icon (Upload) in the Arduino IDE to compile and upload your sketch to the Arduino board.

Program:

```
int led1=7;
void setup()
{
 //pinMode(pin,mode);
 pinMode(led1,OUTPUT);
}
void loop()
{
  digitalWrite(led1,HIGH);
  delay(1000);
  digitalWrite(led1,LOW);
  delay(1000);
 }
```

Observation:

LED connected to pin 7 on the Arduino board was observed and seen that the LED blinked on and off with a 1-second interval.

Result:

Arduino programming was executed with a simple program that blinked an LED connected to the Arduino board.

Exp. NO.6

DATE:

Explore different communication methods with IoT devices (Zigbee)

Objective:

To introduce Zigbee-based communication between two nodes

Materials Required:

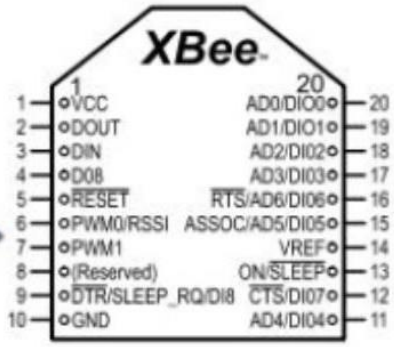
- Zigbee (configured as Router)
- Zigbee (configured as Co-Ordinator)
- PC /Laptops with XCTU installed 02 Nos
- USB Cable 02 Nos

Theory:

Zigbee is a wireless communication protocol targeted for battery-powered devices (it has both low power and low cost). It generally operates in the 2.4GHz range and supports data ranges from 20 to 250 kbits/s.

Digi International developed the XBee modules, a family of wireless communication devices. These modules support various wireless communication protocols, including Zigbee, Wi-Fi, and cellular, and they can communicate over UART (Universal Asynchronous Receiver-Transmitter) interfaces.

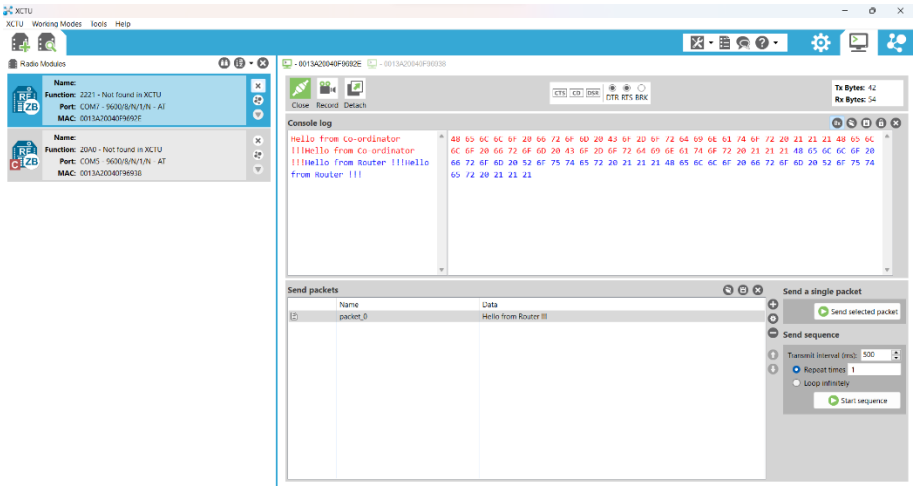
These modules utilize Zigbee communication, a low-power wireless communication protocol commonly used in home automation, industrial control, and sensor networks. XBee Zigbee modules are capable of forming mesh networks, making them suitable for applications where reliability and long-range communication are essential



Procedure:

1. Setting Up the Hardware:
2. Place the First Zigbee module connected to the PC with a USB cable.
3. Open the XCTU Software installed
4. Click the Add button to identify the Zigbee module
5. Identify the module connected and configure it as the router
6. Place the Second Zigbee module connected to the PC with a USB cable.
7. Open the XCTU Software installed
8. Click the Add button to identify the Zigbee module
9. Identify the module connected and configure it as the Co-Ordinator
10. Open the Terminal for both modules
11. Transmit data string “Hello Router” from Co-Ordinator and “Hello Co-Ordinator” from Router
12. Check both messages are transmitted and received properly

Output:



Observation:

- The Router and Co-Ordinator are configured properly
- Send messages using API commands

Result:

Thus, communication was established between the two Zigbee Modules.

Exp. NO.7

DATE:

Introduction to Raspberry PI platform and Python programming

Objective:

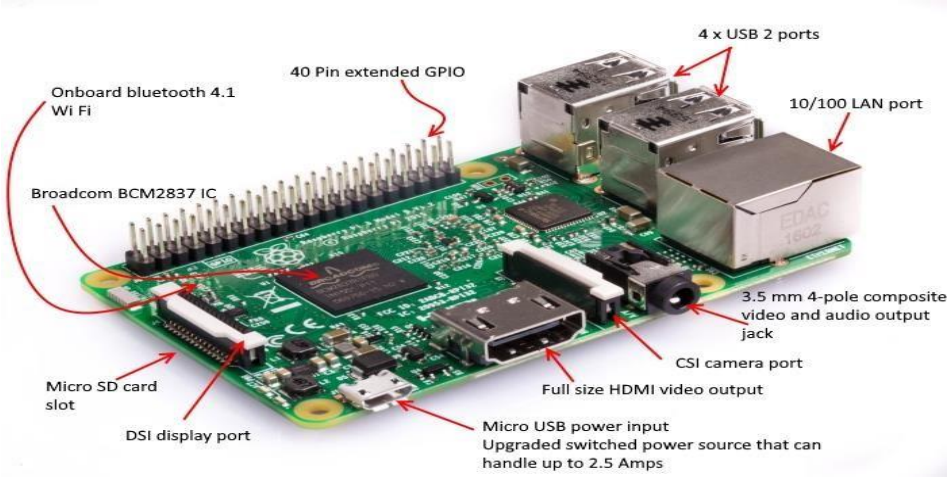
To introduce the Raspberry PI platform and Python programming

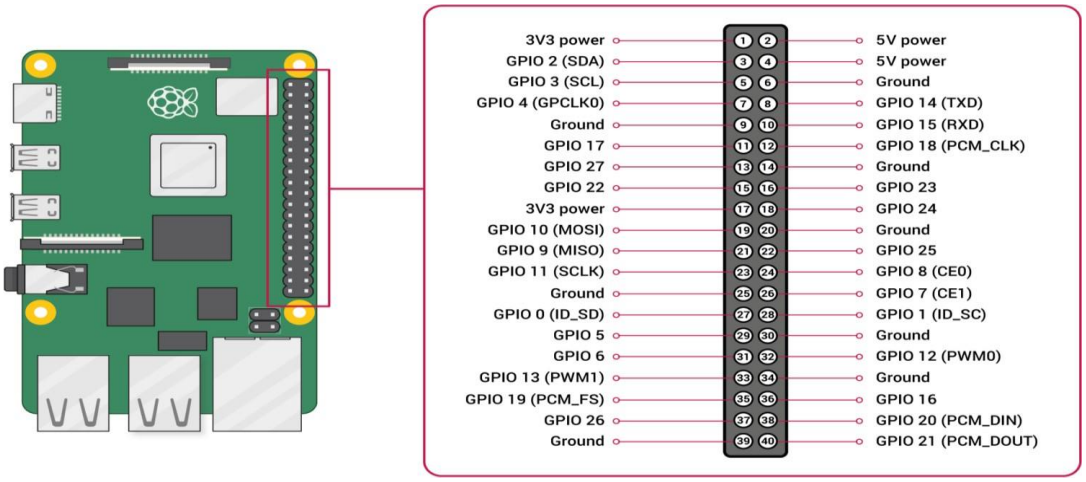
Materials Required:

- Raspberry PI
- Monitor
- Keyboard
- Mouse
- USB Cable 02 Nos
- HDMI Cable 01 No

Theory

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. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word processing, and playing games.





COMPONENTS:

- System on a Chip (SoC):
  - This is the main component of the Raspberry Pi, integrating the CPU, GPU, RAM, and other essential elements.
- Central Processing Unit (CPU):
  - The CPU is the primary processing unit responsible for executing instructions, running programs, and handling computations.
- Graphics Processing Unit (GPU):
  - The GPU handles graphical computations and is responsible for rendering video and images on the screen.
- RAM (Random Access Memory):
  - This is the memory that the Raspberry Pi uses to temporarily store data for quick access by the CPU. It's crucial for multitasking and running applications.
- GPIO (General Purpose Input/Output):
  - These pins allow the Raspberry Pi to interact with external components like sensors, LEDs, buttons, and other devices.
- USB (Universal Serial Bus) Ports:
  - These ports enable the connection of various peripherals such as keyboards, mice, storage devices, and other USB-compatible hardware.
- HDMI (High Definition Multimedia Interface):
  - The HDMI port provides high-definition audio and video output, allowing the Raspberry Pi to connect to monitors, TVs, or displays.

- Audio Output Jack:
  - This allows the connection of speakers, headphones, or audio systems for sound output.
- MicroSD Card Slot:
  - The Raspberry Pi uses a microSD card as its primary storage, where the operating system and user data are stored.
- Ethernet Port:
  - Some models include an Ethernet port for a wired network connection, enabling internet access and local network communication.
- Wi-Fi and Bluetooth (on some models):
  - Integrated wireless connectivity allows the Raspberry Pi to connect to Wi-Fi networks and Bluetooth-enabled devices.
- Camera and Display Ports:
  - Some models feature ports for connecting the Raspberry Pi Camera Module and official Display Module, enabling camera and display functionalities.
- Power Port:
  - The power port supplies the Raspberry Pi with the necessary electrical power, often through a micro-USB or USB-C port.

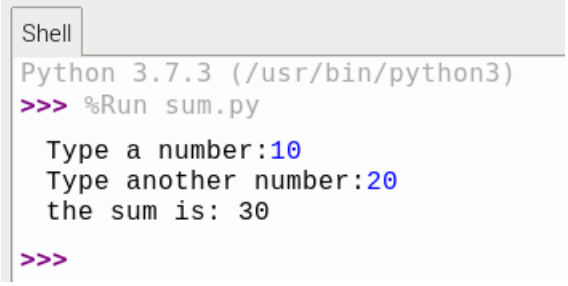
Procedure:

1. Connect Raspberry PI with monitor, keyboard and mouse
2. Open the command terminal.
3. Open the Thonny python
4. Write a sample program in Python
5. Execute the program and check for the result

Program to add two numbers using Thonny IDE :

```
x = input("Type a number: ")
y = input("Type another number: ")
sum = int(x) + int(y)
print("The sum is: ", sum)
```

Output:



Observation:

Raspberry PI is loaded properly and a sample Python program is written using the Thonny Python editor and executed.

Result:

Thus, the study of Raspberry Pi has been done and a sample program using Python is executed.

Exp. NO.8

DATE:

Interfacing sensors with Raspberry PI

Objective:

To introduce the sensor interfacing with the Raspberry PI platform

Materials Required:

- Raspberry PI Kit
- DTH11 Temp and Humidity sensor
- Breadboard and jumpers

Theory:

DHT11 Temperature & Humidity Sensor features a temperature & humidity sensor complex with a calibrated digital signal output. By using the exclusive digital signal acquisition technique and temperature & humidity sensing technology, it ensures high reliability and excellent long-term stability. This sensor includes a resistive-type humidity measurement component and an NTC temperature measurement component, and connects to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness.

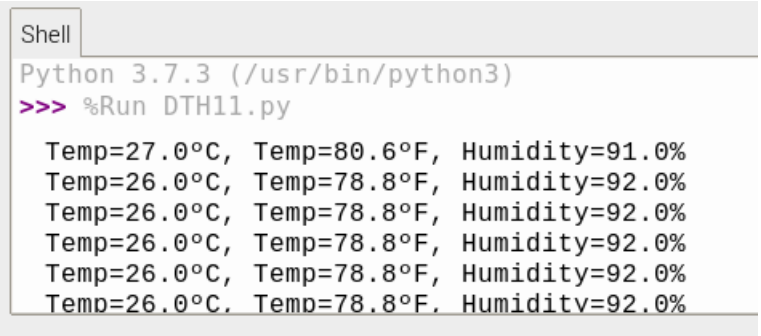
Procedure:

1. Connect Raspberry PI with monitor, keyboard and mouse
2. Open the command terminal.
3. Open the Thonny python
4. Write a sensor interface program in Python using DTH11
5. Execute the program and check for the result

Program:

```
    # Raspberry Pi - Reading Sensor Data (e.g., DHT11 temperature and humidity sensor)
import Adafruit_DHT
sensor = Adafruit_DHT.DHT11
pin = 4  # GPIO pin number
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
if humidity is not None and temperature is not None:
    print(f"Temperature: {temperature}°C, Humidity: {humidity}%")
else:
    print("Failed to retrieve data from the sensor.")
```

Output:



Observation:

Raspberry PI is loaded properly and the DTH11 python program is written using Thonny python editor and executed.

Result: The DTH11 sensor is interfaced with Raspberry PI successfully

Exp. NO.9

DATE:

Communicate between Arduino and Raspberry PI using any wireless medium

Objective:

To introduce the sensor interfacing with the Raspberry PI platform

Materials Required:

- Arduino Uno kit
- Raspberry PI Kit
- Zigbee – 2 modules
- LM35-temperature sensor
- USB Cable -02 Nos
- DC jack
- Breadboard and jumper wires

Procedure:

1. Write Arduino Code
2. Replace Coordinator XBee's MAC ID and calculate Checksum
3. Connect Arduino with Router ZigBee Module (4-Tx, 5-Rx, 3.3V-3.3V, GND-GND)
4. Connect Coordinator ZigBee XBee with the USB Port of Raspberry Pi4
5. Use the awspublish.py file and replace AWS credentials
6. If needed run the command, sudo pip install xbee
7. Arduino publishes sensor data on its Zigbee to Raspberry Pi's Zigbee module
8. You will start getting temperature data in the Raspberry PI console

Program: Arduino

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

#define ssRX  9 /* Rx pin for software serial */
#define ssTX  8 /* Tx pin for software serial */

/* Create object named xbee_ss of the class SoftwareSerial */
SoftwareSerial xbee_ss(ssRX, ssTX); /* Define pins for software serial instance named xbee-ss(any name of your choice) to be connected to xbee */
/* ssTx of Arduino connected to Din (pin 3 of xbee) */
/* ssRx of Arduino connected to Dout (pin 2 of xbee) */
```

```
XBee xbee = XBee(); /* Create an object named xbee(any name of your choice) of the class XBee */

ZBRxIoSampleResponse ioSamples = ZBRxIoSampleResponse();
/* Create an object named ioSamples(any name of your choice) of the class ZBRxIoSampleResponse */

void setup() {
  Serial.begin(9600); /* Define baud rate for serial communication */
  xbee_ss.begin(9600); /* Define baud rate for software serial communication */
  xbee.setSerial(xbee_ss); /* Define serial communication to be used for communication with xbee */
  /* In this case, software serial is used. You could use hardware serial as well by writing "Serial" in place of "xbee_ss" */
  /* For UNO, software serialis required so that we can use hardware serial for debugging and verification */
  /* If using a board like Mega, you can use Serial, Serial1, etc. for the same, and there will be no need for software serial */
}

void loop() {
  xbee.readPacket(); /* Read until a packet is received or an error occurs */

  if(xbee.getResponse().isAvailable()) /* True if response has been successfully parsed and is complete */
  {

if(xbee.getResponse().getApiId()==ZB_IO_SAMPLE_RESPONSE)
/* If response is of IO_Sample_response type */
   {
     xbee.getResponse().getZBRxIoSampleResponse(ioSamples);     /* Get the IO Sample Response */

     Serial.print("Received I/O Sample from: ");

     Serial.print(ioSamples.getRemoteAddress64().getMsb(), HEX); /* DH(in HEX format) of the sending device */
     Serial.print(ioSamples.getRemoteAddress64().getLsb(), HEX); /* DL(in HEX format) of the sending device */
```

```
    Serial.println("");

    if (ioSamples.containsAnalog()) { /* If Analog samples present in the response */
      Serial.println("Sample contains analog data");
    }

    if (ioSamples.containsDigital()) { /* If Digital samples present in the response */
      Serial.println("Sample contains digtal data");
    }

    /* Loop for identifying the analog samples present in the received sample data and to print it */
    for (int i = 0; i <= 4; i++) { /* Only 4 Analog channels */
      if (ioSamples.isAnalogEnabled(i)) { /* Check Analog channel mask to see if the any pin is enabled for analog input sampling */
        Serial.print("Analog (AI");
        Serial.print(i, DEC);
        Serial.print(") is ");
        Serial.println(ioSamples.getAnalog(i), DEC);
      }
    }

    for (int i = 0; i <= 12; i++) { /* 12 Digital IO lines */
      if (ioSamples.isDigitalEnabled(i)) { /* Check Digital channel mask to see if any pin is enabled for digital IO sampling  */
        Serial.print("Digital (DI");
        Serial.print(i, DEC);
        Serial.print(") is ");
        Serial.println(ioSamples.isDigitalOn(i), DEC);
      }
    }
  }
  else
  {
    Serial.print("Expected I/O Sample, but got ");
    Serial.print(xbee.getResponse().getApiId(), HEX); /* Print response received instead of IO_Sample_Response */
  }
}
```

```
  else if (xbee.getResponse().isError()) { /* If error detected, print the
error code */
    Serial.print("Error reading packet.  Error code: ");
    Serial.println(xbee.getResponse().getErrorCode());
  }
}
```

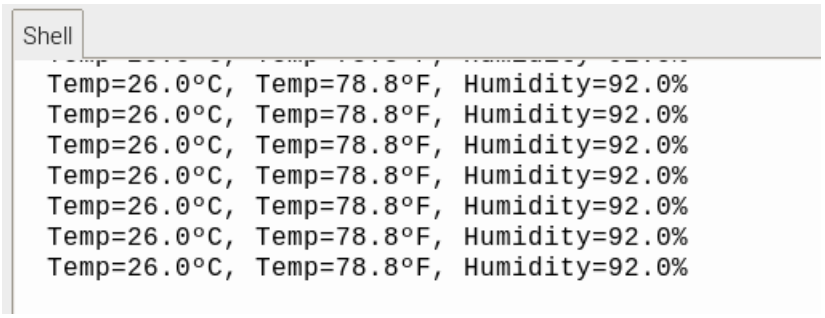
Program: Raspberry Pi

```
import serial

# Enable USB Communication
ser = serial.Serial('/dev/ttyUSB0', 9600,timeout=.5)

while True:
    ser.write('Hello User \r\n')        # write a Data
    incoming = ser.readline().strip()
    print 'Received Data : '+ incoming
```

Output:



Observation:

Temperature and humidity data is communicated between Arduino and Raspberry PI using the ZigBee module.

Result:

Thus, the communication between Arduino and Raspberry Pi using Zigbee module is executed.

Exp. NO.10

DATE:

Setting up a cloud platform for data logging

Objective:

To introduce the cloud platform setting for data logging

Materials Required:

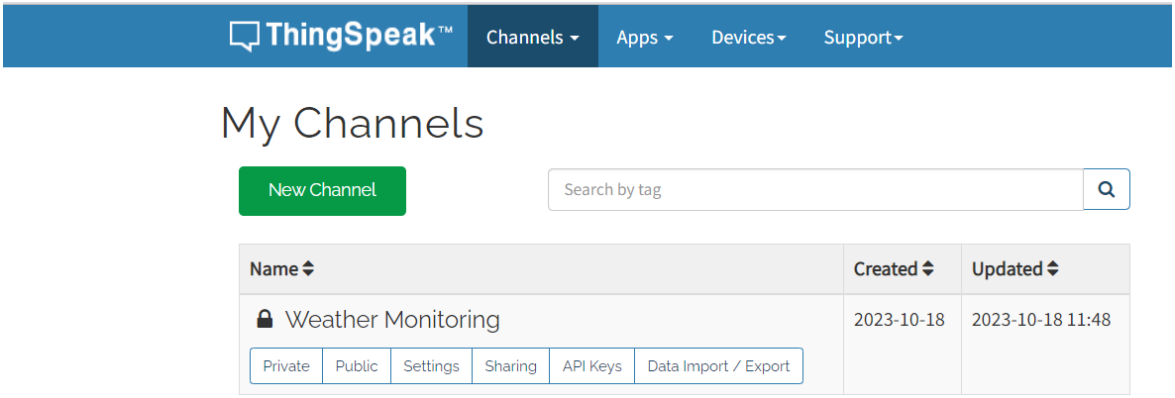
- General PC
- Internet connectivity

Procedure:

1. Open any browser in the PC
2. Open https://thingspeak.com/
3. Create an account by click on “Getting started for free”
4. Create an account with username and password
5. Logging in to cloud using username and password
6. Create a new channel in cloud named “Weather Monitoring”
7. Add fields to channel as “Temperature” and “Humidity”
8. In the API Keys section note API Keys for read and write
9. Attach the channel to dashboard for monitoring

Observation:

Cloud platform has been set up using thingpeak.com



Result:

Thus, a cloud platform has been created using thingspeak.com.

Exp:11

Date:

Log Data using Raspberry PI and upload it to the cloud platform

Objective:

To introduce the sensor data logging to the cloud using the Raspberry PI platform

Materials Required:

- Raspberry PI Module
- DHT11 Sensor
- USB Cable
- Jumper Wires

Procedure:

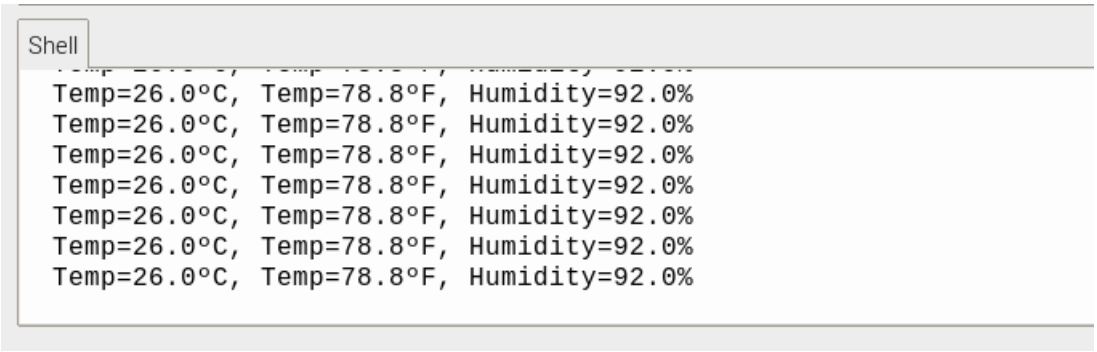
1. DHT11 pin connected to the 4-pin of Raspberry PI.
2. Login to the thingspeak cloud platform
3. Create a channel in the cloud with fields such as temperature and humidity
4. Note the API read and API write key
5. Write a Python program in Raspberry PI using the Thonny editor
6. Install all necessary packages
7. Log the data using the write API key function
8. Observe the data in the dashboard of the cloud

Program:

```
import thingspeak
import time
import Adafruit_DHT
channel_id = 1391845 # put here the ID of the channel you created before
write_key = 'TNXXJJII892UHJ1C' # update the "WRITE KEY"
pin = 4
sensor = Adafruit_DHT.DHT11
def measure(channel):
    try:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        if humidity is not None and temperature is not None:
            print('Temperature = {0:0.1f}*C Humidity = {1:0.1f}%'.format(temperature, humidity))
        else:
            print('Did not receive any reading from sensor. Please check!')
        # update the value
        response = channel.update({'field1': temperature, 'field2': humidity})
    except:
        print("connection failure")
```

```
if __name__ == "__main__":
    channel = thingspeak.Channel(id=channel_id, write_key=write_key)
    while True:
        measure(channel)
    #free account has a limitation of 15sec between the updates
        time.sleep(15)
```

Output:



Observation:

Raspberry PI is loaded properly with the DTH11 and python program is written for observing the temperature and humidity.

Result:

Thus, sensor data is logged to cloud platform using Raspberry Pi.

Exp. NO.12

DATE:

Design an IoT-based System

Objective:

To introduce the IOT-based system design for weather monitoring

Materials Required:

- General PC
- Internet connectivity
- Raspberry PI Module
- DHT11 Sensor
- USB Cable
- Jumper Wires

Procedure:

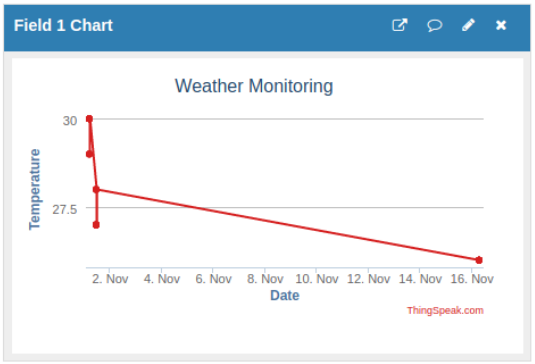
1. Open any browser on the PC
2. Open https://thingspeak.com/
3. Create an account by clicking on “Getting started for free”
4. Create an account with a username and password
5. Logging in to the cloud using username and password
6. Create a new channel in the cloud named “Weather Monitoring”
7. Add fields to the channel as “Temperature” and “Humidity”
8. In the API Keys section note API Keys for reading and write
9. Attach the channel to dashboard for monitoring
10. DHT11 pin connected to the 4-pin of Raspberry PI.
11. Login to Thingspeak cloud platform
12. Create a channel in the cloud with fields as temperature and humidity
13. Note the API read and API write key
14. Write a Python program in Raspberry PI using the Thonny editor
15. Install all necessary packages
16. Log the data using the write API key function
17. Observe the data in the dashboard of the cloud

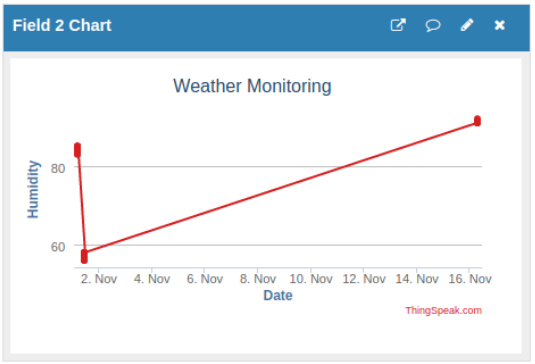
Program:

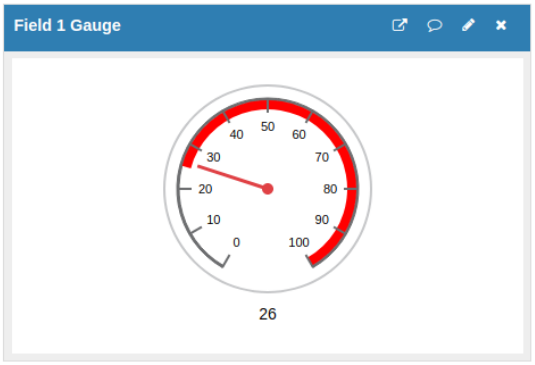
```
import thingspeak
import time
import Adafruit_DHT
channel_id = 1391845 # put here the ID of the channel you created before
write_key = 'TNXXJJII892UHJ1C' # update the "WRITE KEY"
pin = 4
sensor = Adafruit_DHT.DHT11
def measure(channel):
```

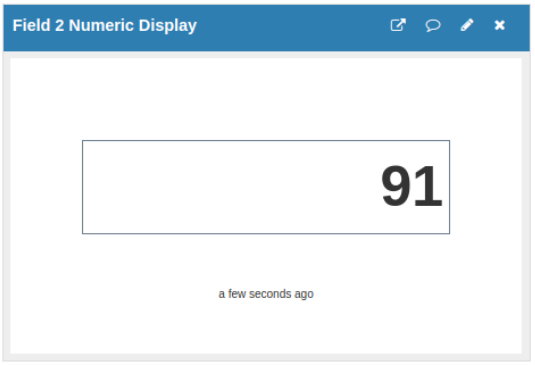
```
        try:
            humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
            if humidity is not None and temperature is not None:
                print('Temperature = {0:0.1f}*C Humidity = {1:0.1f}%'.format(temperature, humidity))
            else:
                print('Did not receive any reading from sensor. Please check!')
            # update the value
            response = channel.update({'field1': temperature, 'field2': humidity})
        except:
            print("connection failure")
    if __name__ == "__main__":
        channel = thingspeak.Channel(id=channel_id, write_key=write_key)
        while True:
            measure(channel)
        #free account has a limitation of 15sec between the updates
            time.sleep(15)
```

Output:









Observation:

Raspberry PI is loaded properly with DTH11 and python program is written to monitor the weather continuously at a time interval of 15 seconds.

Result:

Thus, IoT- based system for weather monitoring is designed and implemented.