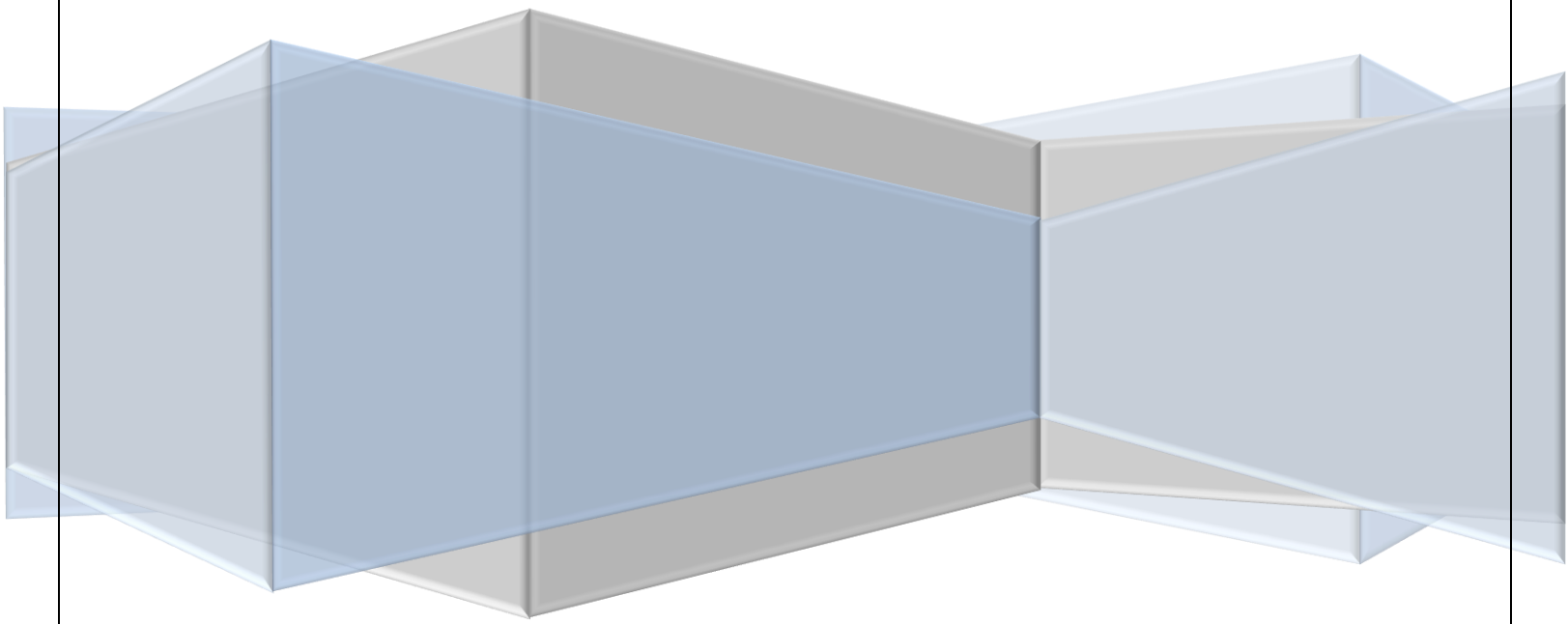
NAME – DIPENDU HAZRA
REGD. NO. – 20BEC0592
SUBJECT – MICROCONTROLLER AND ITS APPLICATIONS
SUBJECT CODE – ECE3003
SLOT – C2
FACULTY – DR. GERARDINE IMMACULATE MARY

J COMPONENT PROJECT REPORT

TITLE – REMOTE ROOM TEMPERATURE MONITORING SYSTEM



DECLARATION

I hereby declare that the Project report entitled “Remote room temperature monitoring” has been written by me as part of our coursework during the Fall Semester 2022-23 under the guidance of Dr. Gerardine Immaculate Mary, Department of Electronics and Communication Engineering, Vellore Institute of Technology, Vellore. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

DIPENDU HAZRA

(20BEC0592)

TABLE OF CONTENTS

ABSTRACT

Today, with the development of industry and technology, overheating and fire can occur in the industrial environment. So, it is important to check and control the temperature the temperature. Sensors are widely used to measure temperature. The IC LM35 is one such temperature sensor. This report presents a simple remote room temperature monitoring system using LM35 temperature sensor and PIC16F877A microcontroller. ESP866 microcontroller is also interfaced with PIC16F877A microcontroller. The system measures the temperature using LM35 and the temperature value is displayed on an IOT platform. In this project, MPLAB IDE is used to program the PIC16F877A microcontroller and Arduino IDE is used to program the ESP8266 microcontroller. The result of the project is displayed in Adafruit IO platform.

ACKNOWLEDGEMENT

I am highly indebted to Dr. Gerardine Immaculate Mary for her guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

It is my privilege to express my sincerest regards to my project coordinator Dr. Gerardine Immaculate Mary for valuable inputs, guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of our project.

I would like to express my gratitude towards my parents & VIT University for their kind co- operation and encouragement which helped me in completion of this project.

AIM

- To interface the LM35 temperature sensor with PIC16F877A.
- To compute the temperature value from the analog voltage obtained from LM35 temperature sensor.
- To transmit the temperature value in ASCII form to ESP8266 from PIC16F877A microcontroller serially.
- To interface ESP8266 with PIC16F877A using jumper wires and program ESP8266 to receive data serially from PIC16F877A.
- To connect ESP8266 over a Wi-Fi network and transmit the data to Adafruit IOT platform and display the temperature value in Celsius with alert signals in the Adafruit IOT platform.

INTRODUCTION

A temperature monitoring system controls and regulates the temperature of a particular environment. A temperature monitoring system has become an essential part of healthcare, hospitals, clinics, food business, and other industries in recent years.

With a temperature monitoring system, we can easily track, control, and regulate the products' temperature in a specific environment. A temperature monitoring system makes sure that our temperature-dependent products stay safe as they are being transported.

So, it is important to check and control the temperature the temperature. Sensors are widely used to measure temperature. The IC LM35 is one such temperature sensor. This report presents a simple remote room temperature monitoring system using LM35 temperature sensor and PIC16F877A microcontroller. ESP866 microcontroller is also interfaced with PIC16F877A microcontroller. The system measures the temperature using LM35 and the temperature value is displayed on an IOT platform. In this project, MPLAB IDE is used to program the PIC16F877A microcontroller and Arduino IDE is used to program the ESP8266 microcontroller. The result of the project is displayed in Adafruit IO platform.

LITERATURE SURVEY

Adamu Murtala Zungeru et al.[1] proposed an **Automatic room heater control system**. The system uses PIC 16F877A microcontroller for the control unit and LM35 as the temperature sensor. The output was varied by setting the temperature at various levels and it was discovered that the Fan was triggered ON when the room temperature was higher than the reference temperature and the heater was triggered ON while the Fan triggered OFF when the room temperature was lower than the reference temperature. The system is exceptionally helpful for people who are disabled. This system can be used in the industry or any enclosure where temperature is needed to be maintaining at a particular value. The system was designed using Proteus and Multisim Software. The system was simulated and working according to the design specifications. In future a GSM module can be integrated with the system so that one can be able to operate their temperature control system from a distance.

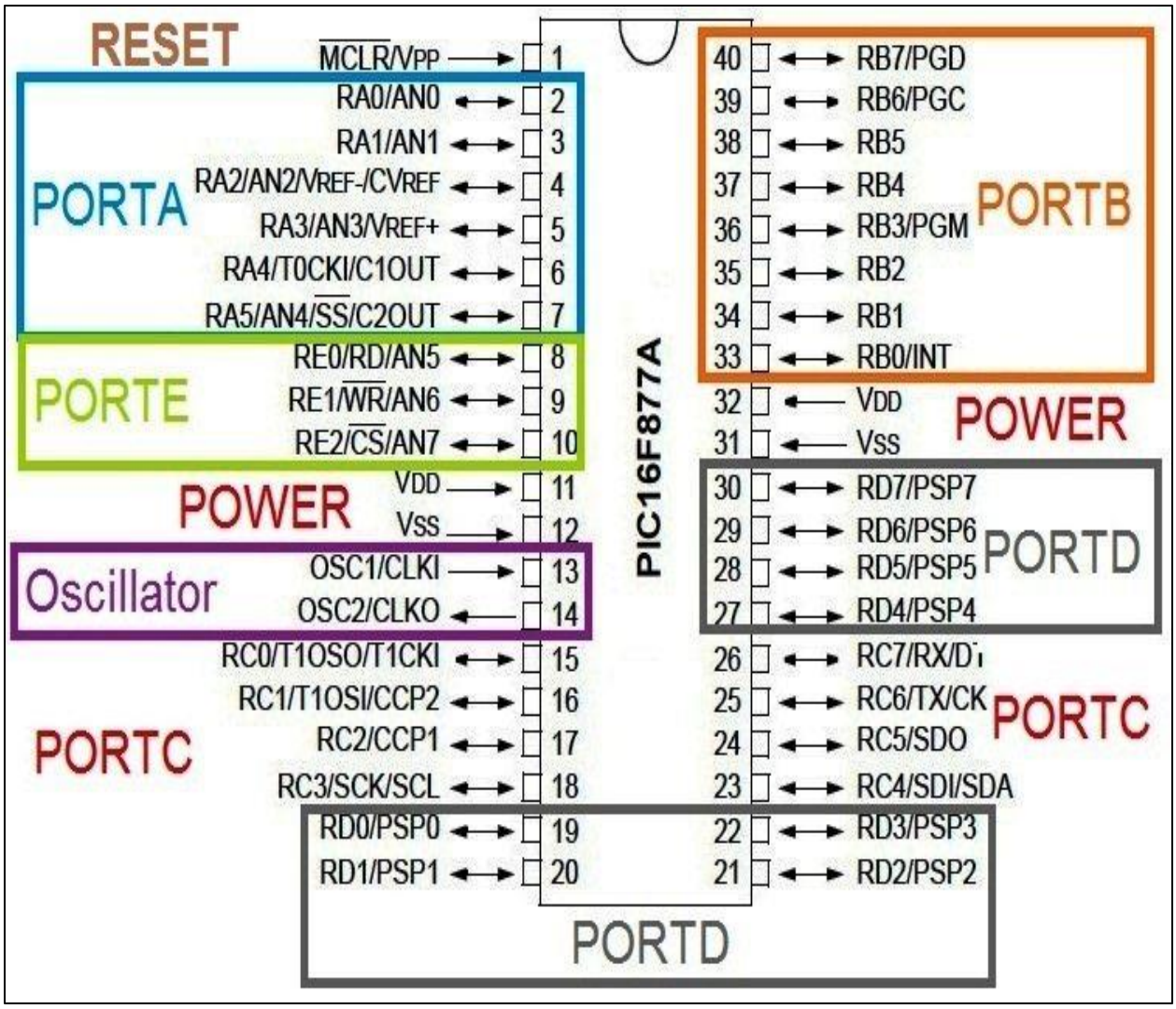
Sadik Kamel Gharghan[2] proposed a **real-time remote monitoring system** (RTRMS) for monitoring the temperature of patients admitted to hospitals. A GSM modem was interfaced with microcontroller PIC16F877A to alerts physicians in real time via short message service (SMS) in emergency cases when the temperature of a patient rises. A sleep/wake energy-efficient algorithm has been implemented inside the microcontroller to reduce the power consumption of GSM and microcontroller. The microcontroller and GSM modem will be in a sleep mode when the temperature of a patient is steady. Consequently, power consumption can be improved and battery lifespan can be prolonged for RTRMS. In addition, the measurement accuracy was confirmed relative to the benchmark (digital thermometer) based on the mean absolute error (MAE). Results show that the proposed RTRMS is achieved 99% power savings relative to traditional RTRMS (i.e., without sleep/wake algorithm). The obtained MAE of 0.205 suggested a close agreement between the benchmark and the proposed RTRMS. The proposed system is a real-time monitoring, applicable, cost-effective, and efficient means for transmitting information because it utilizes the advantages of the infrastructure of GSM network and features a ready mobile device connected with physicians. In addition, the RTRMS can be used in numerous applications and for others parameters rather than temperature with simple modifications (e.g., heart rate, respiratory rate, and electrocardiography for monitoring health care, rehabilitation for monitoring gait speed and cadence, and muscle activity and patient fall detection for safety monitoring).

C. Akshay et al.[3] proposed a system to design, develop and implement a sensor-based wireless communication system to monitor and control the agricultural parameters like temperature and humidity in real time for better management and maintenance of agricultural production and to prevent the severe attack of diseases on the crops caused by the climatic conditions. In green house technology, the automation of agricultural parameters becomes a necessary part. Hence wireless communication with simple hardware and user friendly software like Labview is shown to be an efficient solution for automated green house. In this paper a precision **Green house management** approach to monitor and control the climate and irrigation system is demonstrated. It is proved to be a boon for Hi Tech agricultural field. Although the experimental results have shown for two parameters, the system is completely scalable. The proposed approach has a great potential for remote crop monitoring and control using WSN technology for large scale green house. The system presented here is user friendly, low cost and can be easily implemented.

Hariram M Shetty et al.[4] proposed a **fully automatic hydroponics system** which helps in monitoring and controlling temperature, Humidity, pH and EC in Hydroponics. Hydroponics is a method of growing crops without soil. Plants are grown in rows or on trellises, just like in a traditional garden, but they have their roots in water rather than in dirt. Although, there are different ways in which hydroponics can be implemented, there is no individual system which can measure and control pH and EC level of nutrient solution along with its surrounding temperature and humidity automatically. They have used PIC16F877A microcontroller and four pumps, three of which are used to pump water, nutrient solution, pH solution and the fourth pump is used to control the humidity. A fan is used to control the temperature which increases its speed as the temperature increases. The pumps are turned on depending on the EC and pH values obtained from the electrodes. A passive LCD display is used to display variations in the values. Different Analysis like water usage, plant growth in comparison with regular farming method and hydroponics is successfully completed which results in hydroponics system is significant method in comparison with soiled cultivation method in terms of yield and water usage. This project produced high yield crops by taking minimal space, makes work easier for farmers in growing of plants, and also consumed less amount of water when compared to traditional method resulting in conservation of water.

HARDWARE REQUIREMENTS

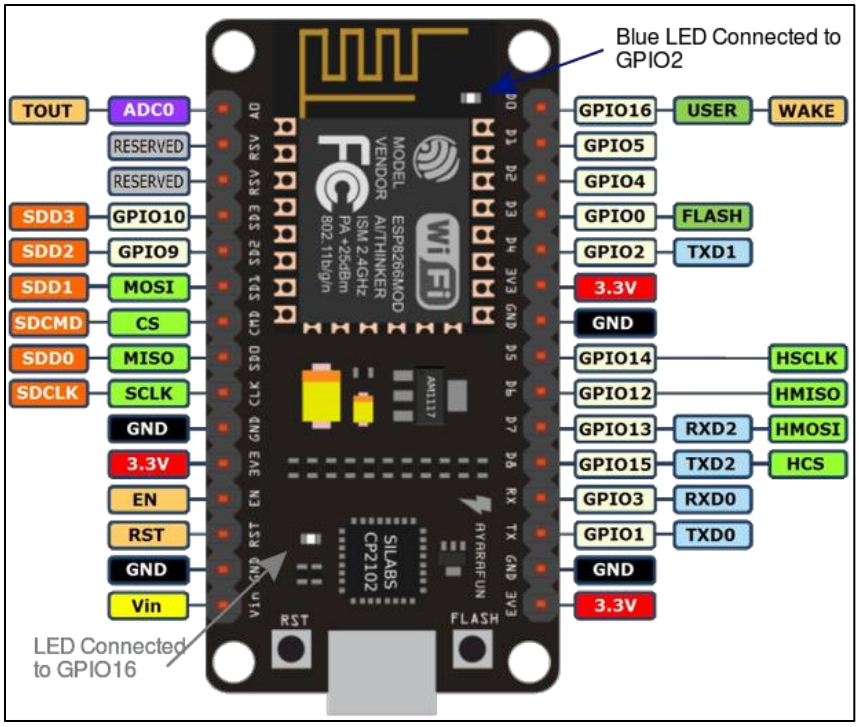
1. **PIC16F877A development board:**
   The PIC16F877A CMOS FLASH-based 8-bit microcontroller is upward compatible with the PIC16C5x, PIC12Cxxx and PIC16C7x devices. It features 200 ns instruction execution, 256 bytes of EEPROM data memory, self programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM functions, a synchronous serial port that can be configured as either 3-wire SPI or 2-wire I2C bus, a USART, and a Parallel Slave Port.



PIN DIAGRAM OF PIC16F877A MICROCONTROLLER

2. **ESP8266:**

The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to a WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, we can simply hook this up to our Arduino device and get about as much WiFi-ability as a WiFi Shield offers. The ESP8266 module is an extremely cost effective board with a huge, and ever growing community.

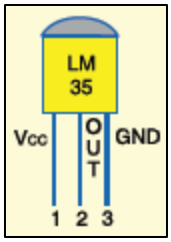


PIN DIAGRAM OF ESP8266 MODULE

3. **LM35 temperature sensor:**

It is a precision integrated-circuit centigrade temperature sensor whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in degree Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. For each degree Celsius change in temperature, the sensor output changes by 10 mV.

The sensor can measure temperature in the range of 0 to 100°C, i.e., the output of the sensor varies from 0 to 1000 mV. The LM35 operates over the temperature range of -55° to +150°C, while the LM35C is rated for a -40°C to +110°C range (-10°C with improved accuracy).

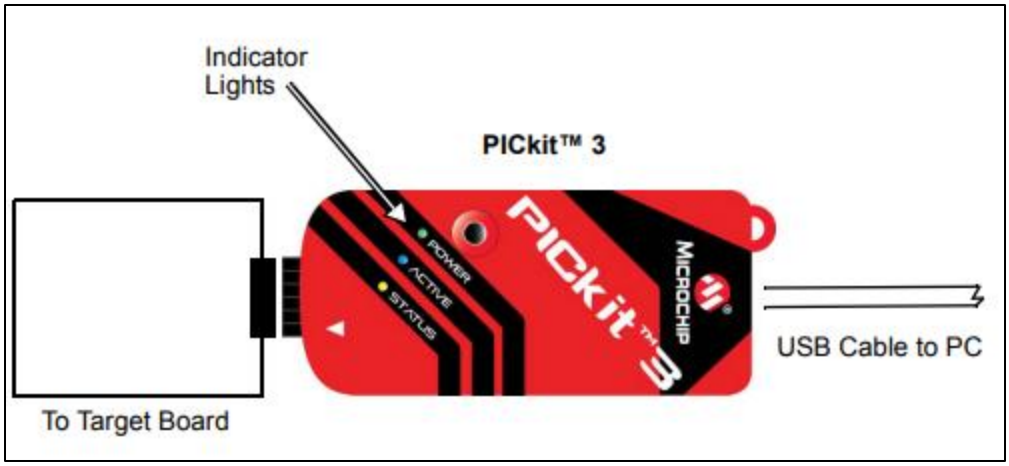


PIN DIAGRAM OF LM35 TEMPERATURE SENSOR

4. **PICkit 3:**

   The PICkit 3 programmer/debugger is a simple, low-cost in-circuit debugger that is controlled by a PC running MPLAB IDE software on a Windows platform. The PICkit 3 programmer/debugger is an integral part of the development engineer's toolsuite. The application usage can vary from software development to hardware integration.

   The PICkit 3 programmer/debugger is a debugger system used for hardware and software development of PIC microcontrollers (MCUs) and Digital Signal Controllers (DSCs) that are based on In-Circuit Serial Programming (ICSP) and Enhanced In-Circuit Serial Programming 2-wire serial interfaces. In addition to debugger functions, the PICkit 3 programmer/debugger system also may be used as a development programmer. The PICkit 3 programmer/debugger is not intended to be used as a production programmer.



BASIC DEBUGGER SYSTEM

5. **5V DC power supply**
6. **Jumper wires**
7. **Micro USB cable**

SOFTWARE REQUIREMENTS

1. **MPLAB IDE and HITECH C COMPILER:**

   MPLAB IDE is a very powerful software development tool for Microchip products (microcontrollers). It consists of tools like text editor, cross-assembler, cross-compiler and simulator.

   Hitech C cross compiler is meant for Microchip PIC10/12/16 series of microcontrollers.

   I have developed an embedded C code in MPLAB IDE and burned the code in PIC16F877A using PICkit 3.

2. **Arduino IDE:**

   I have developed the code for ESP8266 in Arduino IDE and sent the data from ESP866 to Adafruit IOT platform for display of temperature in Celsius.

WORKING PRINCIPLE

The LM35 temperature sensor is interfaced with PIC16F877A microcontroller. The PIC microcontroller provides +5V DC power supply to the Vcc pin of LM35 temperature sensor, in order to activate the sensor. LM35 temperature sensor converts temperature into its proportional analog voltage value. LM35 is three terminal device. Pin number one and three are for 5-volt voltage supply. Pin two is analog voltage output with respect to temperature value. Relation between measured temperature and analog output voltage is:

$1^{\circ}C = 10m\ volt$

Hence for every 1 degree increase in temperature there will be a increment of 10m volt in output voltage of LM35 sensor. PIC16F877A microcontroller is used to measure analog voltage value. PIC16F877A microcontroller has built in ADC (analog to digital converter) is used to measure analog voltage. PIC16F877A microcontroller have eight built in ADC channels. So one can interface up to eight sensors with this microcontroller very easily. I have connected the output pin of LM35 temperature sensor to AN0 pin of PORT A. AN0 pin is used to read the analog output voltage of the LM35 temperature sensor. After reading ADC value, using voltage and temperature relationship, voltage is converted back into temperature.

The voltage output (in volts) of the sensor is: (ADC result × 5) / 1023

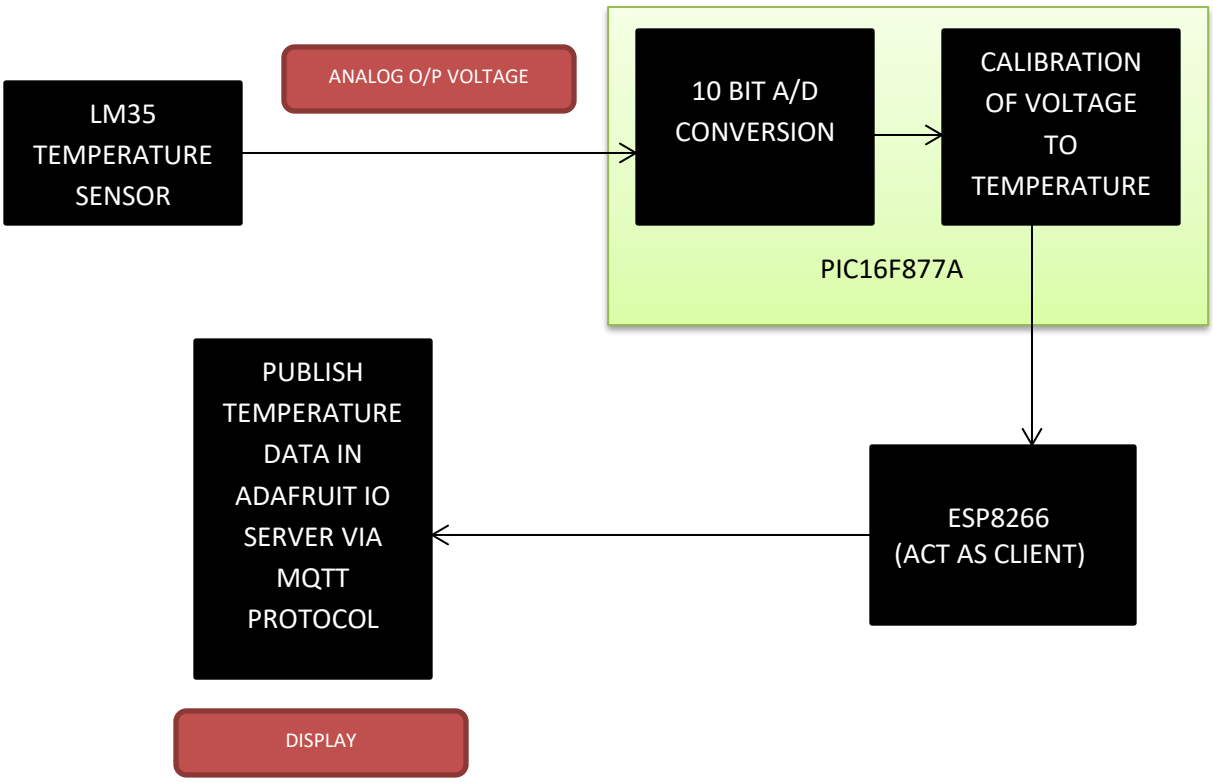
The temperature in degree Celsius is:

Sensor output × 1000 / 10 = Sensor output × 100

This temperature value is converted to ASCII value. The ASCII value is serially transmitted to the Receiver pin of ESP8266 module, which is interfaced with the PIC microcontroller. The Tx pin of PIC16F877A microcontroller is connected to the Rx pin of the ESP8266 module.

Now, the ESP8266 is connected to a Wi-Fi network. An Adafruit account is created in “io.adafruit.com” . The ESP8266 is connected to the Adafruit IOT platform via MQTT protocol. The ASCII value received is converted back to temperature value. The temperature value is published in the Adafruit IO server via MQTT protocol.

In this way , the surrounding temperature is measured and displayed by the system remotely.



SYSTEM BLOCK DIAGRAM

METHODOLOGY

The following methodology to used to build the system:

- The Vcc pin of LM35 temperature sensor is connected to +5V power supply of PIC16F877A.
- The Gnd pin of LM35 temperature sensor is connected the Gnd pin of PIC16F877A.
- The output pin of LM35 is connected to the AN0 pin of PIC16F877A.
- The analog output is read by the AN0 pin of PIC16F877A and the temperature in degree Celsius is calculated using a formula, which is already mentioned in the working principle section.
- The ASCII value is calculated from the temperature value.
- The ASCII value is serially transmitted to the ESP8266. So, the Tx pin of PIC16F877A is connected to the Rx pin of ESP8266.
- The code for PIC16F877A is developed in MPLAB IDE and compiler used is HITECH C compiler.
- The code for PIC16F877A is burnt from MPLAB IDE to the PIC microcontroller using PICkit 3 programmer/debugger.
- Now, ESP8266 is connected to a Wi-Fi network.
- An Adafruit account is created in “io.adafruit.com”.
- The ESP8266 is connected to the Adafruit IOT platform via MQTT protocol.
- The ASCII value received is converted back to temperature value.

- The temperature value is published in the Adafruit IO server via MQTT protocol.
- In this way , the surrounding temperature is measured and displayed by the system remotely.
- The code for ESP8266 is written in ARDUINO IDE software and uploaded to ESP8266 using a micro USB cable.

ALGORITHM OF MPLAB IDE CODE

1. Start the process.
2. Use "#if defined(__PCM__)" command to include the compiler for the PIC microcontroller.
3. Include the library file for PIC16F877A microcontroller.
4. Use "#device adc=10" command to use 10-bit ADC of the PIC microcontroller.
5. Use "#fuses HS,NOWDT,NOPROTECT,NOLVP" to include the library functions for including UART protocol of serial communication.
6. Use "#use DELAY(CLOCK=10000000)" command to produce delay using a clock frequency of 10MHz.
7. Use "#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)" command to use RS232 standard of serial communication with baud rate of 9600 and C6 and C7 pins as Tx and Rx pin, respectively.
8. Declare the prototype of a function with name "rx" and return type void to receive data serially.
9. Declare the prototype of a function with name "tx" and return type char to transmit data serially.
10. Declare two float variables with name a, and v1.
11. Declare 3 variables with name value, hun, ten, and one with unsigned long int datatype.
12. Declare the main() function with return type void.
13. Give a delay of 100 ms.
14. Enable the ADC module and set the clock to internal ADC clock.
15. Setup the ADC pins A0, A1 and A2 of the PIC microcontroller.
16. Give a delay of 250 ms.
17. Configure E1 pin for reading data.
18. Configure A5 pin for writing data.
19. Give delay of 100ms and 1000 ms.
20. Declare a while loop to run indefinitely.
21. Set the ADC channel OFF.
22. Give a delay of 50ms.
23. Read the ADC output.
24. Give a delay of 10 ms.
25. Convert ADC output to temperature in degree Celsius using the formula: v1=(a/1024)*500; where, a=ADC output.
26. Store value of v1 in value.
27. Divide value by 10 and store it in ten.
28. Store the remainder after dividing value by 10, in one.
29. Invoke the Tx(char) function to transmit '$' serially.
30. Give a delay of 50 ms.

31. Transmit the ASCII value of the measured temperature serially by invoking Tx(char) function.
32. Invoke the Tx(char) function to transmit '#' serially.
33. Give a delay of 1250 ms.
34. Use "putc()" function to print the ASCII value of the measured temperature in degree Celsius.
35. If temperature is less than 40 degree Celsius, print '1' with a delay of 50 ms.
36. Else, print '0' with a delay of 50 ms.
37. Print '#' , and give a delay of 1250 ms.
38. End the While loop.
39. End the loop() function.
40. Use "#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7,STOP=1,PARITY=N)" command to use RS232 standard for serial communication, generate baud rate of 9600, declare C6 pin as Tx pin and C7 pin as the Rx pin.
41. Declare Tx(char ch) function with return type void to transmit ASCII values serially.
42. Print the character ch using "putc()" function.
43. End the Tx() function.
44. Declare rx(void) function with return type char to receive data serially.
45. Declare a variable named rx_ch with char datatype.
46. Use getch() function to wait for a character to come in over the RS232 Rx pin and return the received character, which is stored in rx_ch variable.
47. Return the value of rx_ch variable.
48. End the rx() function.
49. Stop the process.

MPLAB IDE CODE

```
#if defined(__PCM__) // COMPILER FOR PIC MICROCONTROLLER
#include <16F877A.h> // TO INCLUDE LIBRARY FILE FOR PIC16F877A
#device adc=10 // 10-BIT ADC OF PIC IS USED
#fuses HS,NOWDT,NOPROTECT,NOLVP //LIBRARY FUNCTIONS FOR INCLUDING UART PROTOCOL FOR
//SERIAL COMMUNICATION
#use DELAY(CLOCK=10000000) //TO USE DELAY USING CLOCK FREQUENCY OF 10MHz
#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7) // TO USE RS232 STD OF SERIAL COMMUNICATION
WITH BAUD RATE OF 9600 THROUGH TX AND RX PIN

char rx(void); //FUNCTION FOR RECEIVING DATA SERIALLY
void tx(char); //FUNCTION FOR TRANSMITTING DATA SERIALLY

float a,v1;
unsigned long int value,hun,ten,one;

void main()
{
delay_ms(100);
setup_adc(ADC_CLOCK_INTERNAL);//enables the adc module  and sets the clock to internal adc clock
setup_adc_ports(RA0_RA1_RA3_ANALOG ); //SETTING UP THE ADC PINS A0, A1, A2
```

```
delay_ms(250);
output_bit(pin_e1,0);   //CONFIGURE E1 PIN FOR READING DATA
output_bit(pin_a5,1);   //CONFIGURE A5 PIN FOR WRITING DATA
delay_ms(100);
delay_ms(1000);

while(1)
{
set_adc_channel(0); //SETTING ADC CHANNEL OFF
delay_ms(50);
a=read_adc();  //READING ADC OUTPUT
delay_ms(10);
v1=(a/1024)*500; //FORMULA TO CONVERT ADC OUTOUT TO TEMPERATURE IN DEGREE CELCIUS
value=v1;
ten=value/10;
one=value%10;

//TRANSMIT THE ASCII VALUE OF THE MEASURED TEMPERATURE
tx('$');
delay_ms(50);
tx(ten+0x30);
delay_ms(50);
tx(one+0x30);
delay_ms(50);
tx('#');
delay_ms(1250);

//PRINT THE ASCII VALUE OF THE MEASURED TEMPERATURE
putc('$');
delay_ms(50);
putc(ten+0x30);
delay_ms(50);
putc(one+0x30);
delay_ms(50);

// IF TEMPERATURE IS LESS THAN 40 DEGREE CELSIUS, PRINT '1' ELSE PRINT '0'
if(v1<40)
{
putc('1');
delay_ms(50);
}
else
{
putc('0');
delay_ms(50);
}
putc('#');
delay_ms(1250);
```

```
}
}

#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7,STOP=1,PARITY=N) //TO USE RS232 FOR SERIAL
//COMMUNICATION
//USING 9600 BAUD TRATE, C6 AS THE TX PIN AND C7 AS RX PIN
//TRANSMITTER FUNCTION TO TRANSMIT ASCII VALUES SERIALLY
void tx(char ch)
{
putc(ch);
}


// RECEIVER FUNCTION TO RECEIVE DATA SERIALLY

char rx(void)

{

 char rx_ch;

 rx_ch=getch(); //This function waits for a character to come in over the RS232 RX pin and returns the
character.

 return(rx_ch);

}
```

NOTE : This code is compiled using HTECH C COMPILER and burnt in the PIC microcontroller using PICkit 3 programmer or debugger.

ALGORITHM OF ARDUINO IDE CODE

1. Start the process.
2. Include the library for ESP8266 to connect to a Wi-Fi network.
3. Include the libraries for using MQTT protocol for sending data to web server Adafruit IO, where, ESP8266 act as the client to the Adafruit IO server.
4. Define the Wi-Fi parameters, i.e., it's SSID and password.
5. Define the details obtained from the Adafruit IO server ,i.e., Adafruit server's URL, Adafruit server port, Adafruit account's username and key.
6. Define a variable named es1 with int datatype.
7. Create an ESP8266 WiFiClient class to connect to the Adafruit IO server via MQTT protocol.
8. Setup the MQTT client class by passing in the WiFi client and Adafruit IO server and login details.
9. Setup temp feed in the Adafruit IO to make storage field on the Adafruit IO server using the username for the Adafruit account.
10. Define variables with name x, p1, p2, and s1 with char datatype.

11. Declare the setup function with return type void.
12. Start serial communication with a baud rate of 9600.
13. Give a delay of 10 ms.
14. Connect to the Wi-Fi network whose SSID and password are provided.
15. After the ESP8266 gets connected to the Wi-Fi network, access the local IP address for the Wi-Fi network and print it in the serial monitor.
16. End the setup function.
17. Declare the loop function with return type void.
18. Declare a while loop to run indefinitely.
19. Invoke the connect() to connect to the Adafruit IO via MQTT protocol.
20. Declare a do while loop and serially read the value obtained from the PIC microcontroller and store the value in x.
21. Run the loop if x is not equal to '$', or else exit the loop.
22. Use Serial.read() function to read the value obtained from the PIC microcontroller after '$' is received. Store the received data in p1, p2, and s1.
23. Declare a do while loop and serially read the value obtained from the PIC microcontroller and store the value in x.
24. Run the loop if x is not equal to '#', or else exit the loop.
25. Print the received data in the serial monitor.
26. Convert the ASCII value to temperature value using p1 and p2 and store the result in temp_data variable.
27. Convert s1 to integer value from ASCII value and store the result in es1.
28. If es1 is equal to 1, keep it 1 only else make es1 equal to 4.
29. Now grab the current state of the sensor by publishing the temperature data in the Adafruit IO server. Use temp.publish() function to publish the temperature data to the Adafruit IO server.
30. Repeat this every 2s.
31. End the while loop.
32. End the loop() function.
33. Declare the function connect() with return type void.
34. Define a 8 bit integer variable with name "ret".
35. Declare a while loop to run the loop until the ESP8266 gets connected to the Adafruit IO server. Use "mqtt.connect()" function to access the value returned by the Adafruit IO server when we are trying to connect the ESP8266 to the Adafruit IO server via MQTT protocol. Store that value in ret variable. Run the while loop until ret is equal to 0.
36. Use switch case to print messages in the serial monitor based on the value of ret variable.
37. If ret is equal to 1, print "Wrong protocol" in the serial monitor.
38. If ret is equal to 2, print "ID rejected" in the serial monitor.
39. If ret is equal to 3, print "Server unavail" in the serial monitor.
40. If ret is equal to 4, print "Bad user/pass" in the serial monitor.
41. If ret is equal to 5, print "Not authed" in the serial monitor.
42. If ret is equal to 6, print "Failed to subscribe" in the serial monitor.
43. Print "Connection failed" in the serial monitor under default case.

44. End the switch case.
45. If ret is greater or equal to 0, disconnect ESP8266 from the Adafruit IO server.
46. Keep repeating the steps 35 to 45 until ret is equal to 0 with a delay of 5s.
47. End the while loop.
48. If ret is equal to 1, come out of the while loop and return to the loop() function.
49. End the connect() function.
50. Stop the process.

ARDUINO IDE CODE

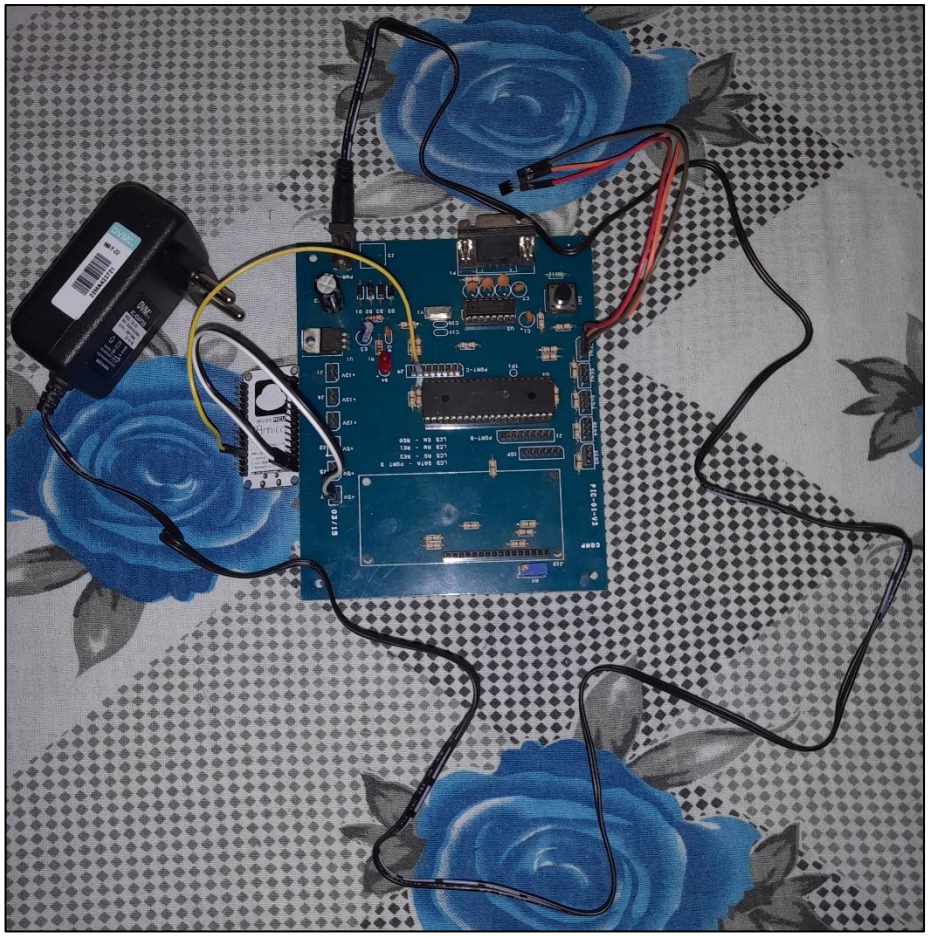
```
// Libraries
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h" //using mqtt protocol for sending data to web server Adafruit

// WiFi parameters
#define WLAN_SSID       "iot"
#define WLAN_PASS       "12345678"

// Adafruit IO
// AIO details to publish data to the aio server
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME  "Iotweather"
#define AIO_KEY       "aio_Bmjb44doK9t4sGenlOoG3kQrqYUn"

// node mcu is serving as mqtt client and sening data to the aio servier

int es1=0;
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;
// Setup the MQTT client class by passing in the WiFi client and MQTT server and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);



// Setup feeds for parameters
Adafruit_MQTT_Publish temp = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/feed_temp");// we are making storage field on the server using our username


char x,p1,p2,s1;



/************************* Sketch Code **********************************/
```

```
void setup() {

 Serial.begin(9600);//begin serial communication with 9600 baud rate
 delay(10);
 //connect to the Wi-Fi network whose SSID and password are given
 Serial.print(F("Connecting to "));
 Serial.println(WLAN_SSID);

 WiFi.begin(WLAN_SSID, WLAN_PASS);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(F("."));
 }
 Serial.println();

 Serial.println(F("WiFi connected"));
 Serial.println(F("IP address: "));
 Serial.println(WiFi.localIP()); //access the local IP of the Wi-Fi network

 }


void loop()
{
while(1)
{
   connect();//call the connect function to connect to Adafruit IO via MQTT

Serial.println("WAITING FOR DATA!");

  do
{
  x=Serial.read();
}while(x!='$');// if x value is not equal to '$',then keep running else exit loop

p1=Serial.read();
p2=Serial.read();
s1=Serial.read();

do
{
  x=Serial.read();
}while(x!='#');// if x value is not equal to '#',then keep running else exit loop

Serial.println("RECEIVED DATA: ");

Serial.write(p1);
Serial.write(p2);
```

```
Serial.write(":");
Serial.write(s1);
Serial.write("/n");
delay(200);

int temp_data=((p1-0x30)*10)+(p2-0x30); //convert ASCII value to temperature value


es1=s1-0x30;
if(es1==1)
{
  es1=1;
  }
  else
  {
    es1=4;
  }


  // Grab the current state of the sensor
  // Publish data
  if (! temp.publish(temp_data))//If temperature data is published to Adafruit IO, else part will be
//executed.
    Serial.println(F("Failed to publish temperature"));
  else
    Serial.println(F("Temperature published!"));


  // Repeat every 2 seconds
  delay(2000);
}
}

// connect to adafruit io via MQTT
void connect() {

  Serial.print(F("Connecting to Adafruit IO... "));

  int8_t ret;

  while ((ret = mqtt.connect()) != 0) {                    // this loop will keep running until the systm is
//connected  to the aio server

    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
```

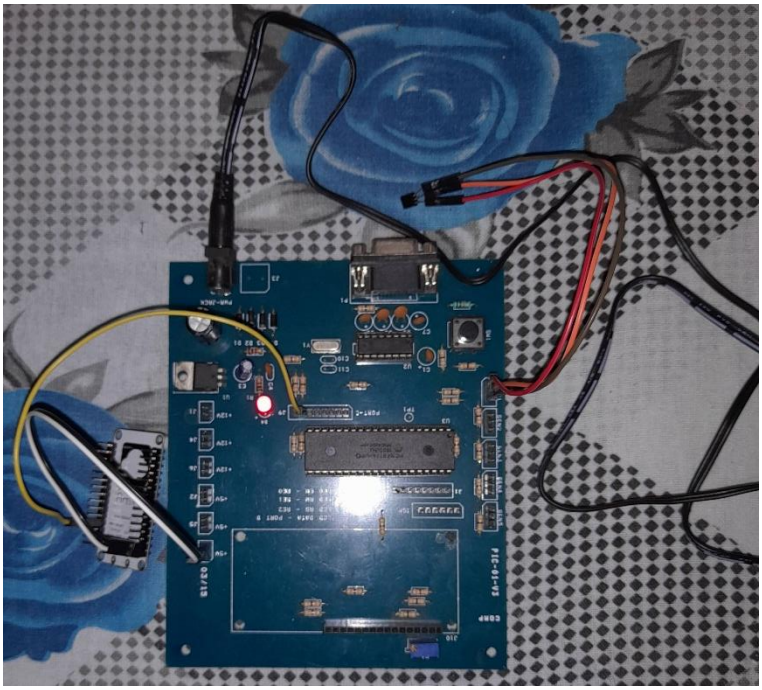
```
    case 5: Serial.println(F("Not authed")); break;
    case 6: Serial.println(F("Failed to subscribe")); break;
    default: Serial.println(F("Connection failed")); break;
   }

   if(ret >= 0)
    mqtt.disconnect();//if ret is greater than equal to 0, disconnect from the Adafruit IO server

   Serial.println(F("Retrying connection..."));
   delay(5000);

  }

  Serial.println(F("Adafruit IO Connected!"));

}
```

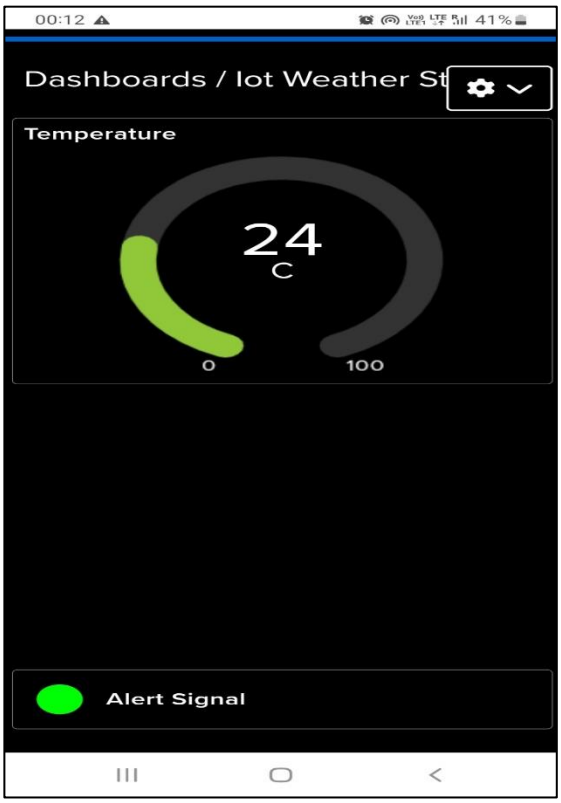
EXPERIMENTAL SETUP



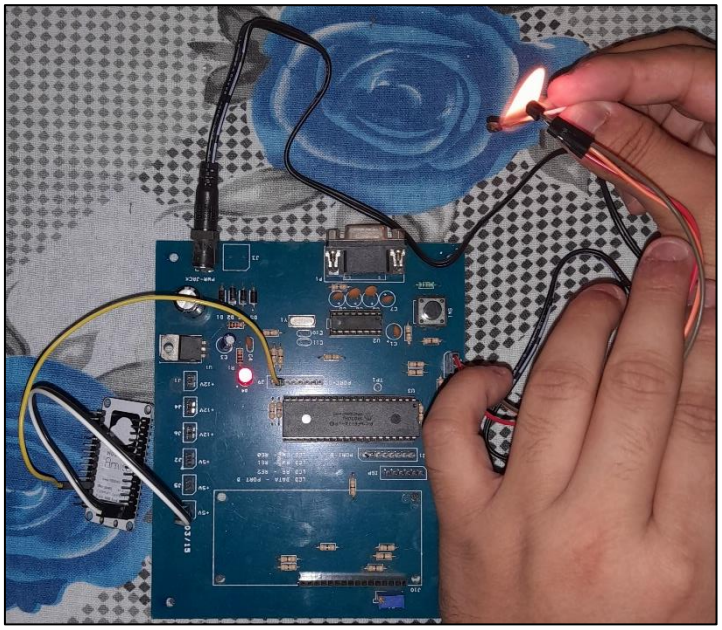
SIMULATION RESULTS

1. **Top figure represents the hardware setup. Bottom figure represents the temperature data in Adafruit IO platform. Here, temperature around the LM35 sensor is 24 degree Celsius which is less than 40 degree Celsius, so, a green alert signal is shown in the Adafruit IO platform, which indicates safety around the system.**

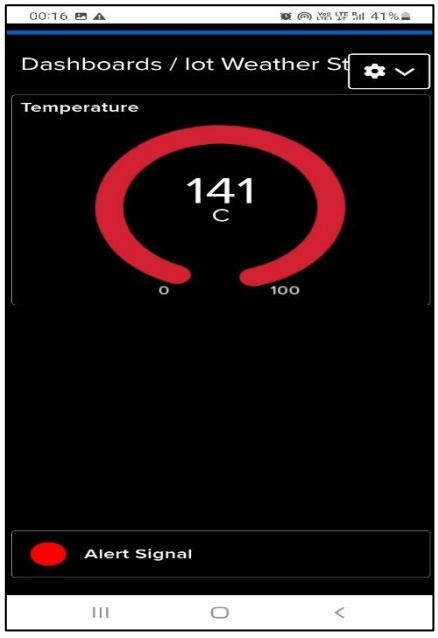




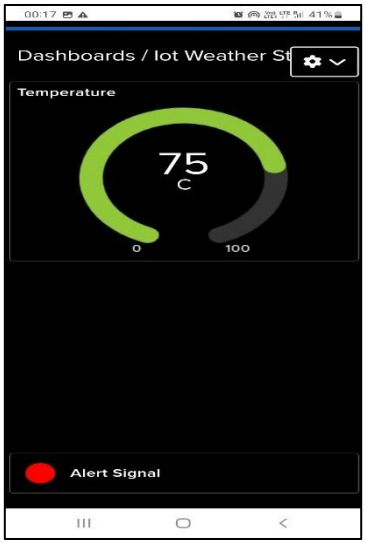
2. **In this case, the LM35 temperature sensor is heated with a flame from a lighted matchstick for a few seconds. So, the surrounding temperature will be very high around the LM35 temperature sensor for a few seconds. The surrounding temperature around LM35 sensor decreases exponentially to a steady state value which is equal to the room temperature.**



3. **The temperature displayed in Adafruit IO platform is 141 degree Celsius. This value is corresponding to the setup discussed in point 2. This value is displayed for some seconds after some seconds of removing the flame from the LM35 temperature sensor. Since, the temperature is greater than 40 degree Celsius, the alert signal is red in colour, which indicates danger around the LM35 temperature sensor. Since, 141>100, so, the temperature scale also turns red**.



4. **The temperature around LM35 sensor is around 75 degree Celsius after a minute of removing the flame from the LM35 temperature sensor. This value is corresponding to the setup discussed in point 2. Since 75<100, so, the temperature scale turn to green colour. Since 75>40, so, the alert signal is still red, which indicated danger around the LM35 temperature sensor.**



5. **The temperature around LM35 sensor is around 29 degree Celsius after approximately 2 minutes of removing the flame from the LM35 temperature sensor. This value is corresponding to the setup discussed in point 2. Since 29<100, so, the temperature scale remains green colour. Since 29<40, so, the alert signal turns to green, which indicated safety around the LM35 temperature sensor.**

CHALLENGES FACED

Initially, I tried to design and implement the system using 8051 microcontroller. Now, LM35 temperature sensor output can't be directly feed to 8051 microcontroller because the output of LM35 temperature sensor is a analog voltage. So, the analog voltage should be first converted to digital voltage and the digital output voltage can be directly feed to one of the ports of 8051 microcontroller. So, an analog to digital convertor is required. So, I tried use interface ADC0804 with 8051 microcontroller. The ADC0804 was not integrated with the 8051 development board. So, the delay value that I was using to interface 8051 microcontroller with ADC0804 was not correct. So, as a result, I was unable to interface ADC0804 with 8051 microcontroller development board.

Due to time constraints, I have to change my decision of using 8051 microcontroller to PIC16F877A microcontroller, which is also an 8-bit microcontroller with many advanced features.

DESIGN CONTRAINTS

Some of the important design constraints noted are as follows:

- The time response of the system is approximately 3 minutes, which is making the entire system unsynchronized.
- The system is not cost effective.

APPLICATIONS

Temperature monitoring and control is used in more ways than we can imagine. Some of the important applications of temperature sensors are as follows:

- Temperature sensors are used in many types of buildings to control heating, ventilation and air conditioning, helping to provide a comfortable environment for occupants as well as improve energy efficiency.
- The healthcare and pharmaceutical industries rely heavily on temperature sensors, with solutions designed specifically to monitor temperatures of fridges and freezers, blood banks, operating theatres, medicines, and medical devices and equipment.
- Monitoring and maintaining the right temperature is critical for food and beverage manufacturers, processing plants, caterers, breweries and dairies. It helps to keep facilities safe and compliant, maintain process efficiency and keep waste to a minimum.

The proposed system can be used in many ways. Some of them are:

- It can monitor the temperature of a patient remotely.
- It can measure the room temperature remotely.
- It can be used to measure the temperature of engine parts in automobiles remotely.

FUTURE WORK

The time response of the system can be improved by using advanced microcontrollers. Efforts can be made to design the system using 8051 microcontroller. Efforts can be made to design and implement the system in a cost effective manner.

CONCLUSION

LM35 temperature sensor is properly interfaced with the PIC16F877A microcontroller. The temperature value from the analog voltage obtained from the LM35 temperature sensor is computed.

ESP8266 module is successfully interfaced with PIC16F877A microcontroller. The temperature is serially transmitted in ASCII form to ESP8266 module from PIN16F877A microcontroller. The ESP8266 module is programmed to receive data serially from PIC16F877A.

ESP8266 is also connected to a Wi-Fi network. ESP8266 is able successfully transmit the temperature data to Adafruit IO platform via MQTT protocol. So, the temperature data in Celsius is displayed in the Adafruit IO platform with proper alert signals.

MPLAB IDE is used to develop the code for PIC16F877A microcontroller. An algorithm for the code developed in MPLAB IDE is also designed.

ARDUINO IDE is used to design the code for ESP8266 module. An algorithm for the code developed in ARDUINO IDE is also designed.

The system properly worked in all conditions. So, finally, a remote room temperature monitoring system is designed and implemented using PIC16F877A microcontroller as the brain of the system.

REFERENCES

1. Adamu Murtala Zungeru, Mmoloki Mangwala, Joseph Chuma, Baboloki Gaebolae, Bokamoso Basutli; Design and simulation of an automatic room heater control system; Heliyon,Volume 4, Issue 6,2018,e00655,ISSN 2405-8440;https://doi.org/10.1016/j.heliyon.2018.e00655.
2. Sadik Kamel Gharghan, Department of Medical Instrumentation Techniques Engineering, Electrical Engineering Technical College, Middle Technical University (MTU), Al Doura 10022, Baghdad-Iraq; Energy-Efficient Remote Temperature Monitoring System for Patients Based on GSM Modem and Microcontroller; Journal of Communications Vol. 12, No. 8, August 2017.
3. C. Akshay et al., "Wireless sensing and control for precision Green house management," 2012 Sixth International Conference on Sensing Technology (ICST), 2012, pp. 52-56, doi: 10.1109/ICSensT.2012.6461735.
4. Hariram M Shetty et al., Information Science and Engineering, Canara Engineering College, Mangalore; Fully Automated Hydroponics System for Smart Farming; I. J. Engineering and Manufacturing, 2021, 4, 33-41.