Table of Contents

List of Figures

1. Background

1.1 Introduction

In the current market, accuracy holds great importance. It can be any physical or software quantity. Temperature and humidity are one of the most important quantity. Inaccuracy in its data can result into unfavorable changes. Temperature and humidity play a very important role in many places such as, cold storage, farms, houses, offices and many other public places.

TempSense had developed a product that is able to detect the real time temperature and humidity with high accuracy. We are using the pic18f45k22 microcontroller which has an EEPROM of 1024 bytes and can work on 16-bit wide instructions. We are using DHT22 sensor measuring temperature and humidity, which sends data in every 2 seconds. For our project we are taking samples every 15 minutes as the temperature will not change in every 2 seconds. 15 minutes will be a good reading time for sensor. For showing the results we are using online HMI which is ThingSpeak. This website is showing the results in the gauge form and it can show the graphical data as well. ThingSpeak cloud can save the data for 24 hours and show the trends for the same. This HMI will be very user friendly and they can see the trends very easily.

1.2 Problem statement

The team has previously observed that the vital data for any system are often temperature and pressure. Currently, the market only has the thermostat that is able to detect the temperature and pressure and that too has lack of efficiency. The inwards and outwards flow of heat and humidity is often missed.

In the products available in the market, consumers have to personally reach to meters for listing down the output. This becomes difficult in the case of the farmers as they need to go to their farms or the places where meters are available for noting these readings. Currently, there is no technology to store and transmit these readings to the end-user. For every different meter, a consumer has to reach different places for taking values. They do not have any systems to integrate these data and receive them on the mobile location.

During a survey of a random 1000 children of 7 years of age, most of the children reported cold and coughing disease. So, their bedrooms were monitored for a week. This happened due to the conditions their bedroom atmosphere was set. Also, it happened because of the mites and molds whose survival is directly proportional to the humidity conditions. (D P Strachan, n.d.)

All the above examples and reasons clearly suggest the demand of the product and the importance of such a system in the market. It will have a huge demand in the market and will be of great value.

2. Product development

2.1 Proposed improvement

TEMPSENSE ™ is developing a new product to measure the temperature and humidity at a specific location without going to that location. The product will be installed one time and then the user can see all the readings on their mobile phone, computer or laptop.

After researching a number of sensors, we have decided to use the DHT22 temperature and humidity sensor for this project. This sensor has the ability to measure the temperature from -40°C to 80°C and humidity from 0 to 100%. The accuracy for temperature readings is ±0.5°C and for humidity readings it is 2-5%. This sensor is very compact and light weight. It will weight around 2.5 grams and has the dimensions 27mm*59mm*13.5mm. (adafruit, n.d.) This sensor can take the readings every 2 seconds. For our application, we will be taking the samples every 15 minutes as in the real weather conditions there is no change in every second.

We are using wireless protocols to develop this system. Wi-Fi will be used for sending data wirelessly to cloud storage. ESP8266 is a low-cost microchip and it is capable of connecting with the internet and send and receive data in real time. The team has decided to use an ESP8266 12E model, which is a microprocessor based. It has a microcontroller attached to it, but according to the need of course we are another microcontroller. It has a 30-pin dual in line package and a micro usb port for powering the device. This type of power source will help reducing the external circuitry. It works on the IEEE 802.11 standard. (20Ze)

The main key element of the product is a pic18f45k22 micro-controller. It is a 40-pin dual in line package. It is a low powered and c programmable device. We will use the MPLAB IDE software programming the microcontroller. PICKIT 3 program loader will help uploading the program to device. Once the program is uploaded on its memory, it has the capability to work as a stand-alone system. With the power provided from the Wi-Fi module, the micro-controller will work as a stand-alone system.

The sensor will be connected with the microcontroller and the microcontroller will send the data serially to the wireless module. Wi-Fi module will send the data to cloud storage and this cloud storage will be accessed with the use of human machine interface. Human machine interface can be defined as the user interface which connects a user to a system, machine or device. We will be using a ThingSpeak HMI for our product. (htt2) This is a free IoT analytics platform for visualizing and analyzing real time data. It can provide immediate graphs of the data received on its cloud storage. This cloud storage is capable of saving the data for up to 24 hours. With the help of this application there is no need for making a web software and setting up servers.

2.2 Block Diagram

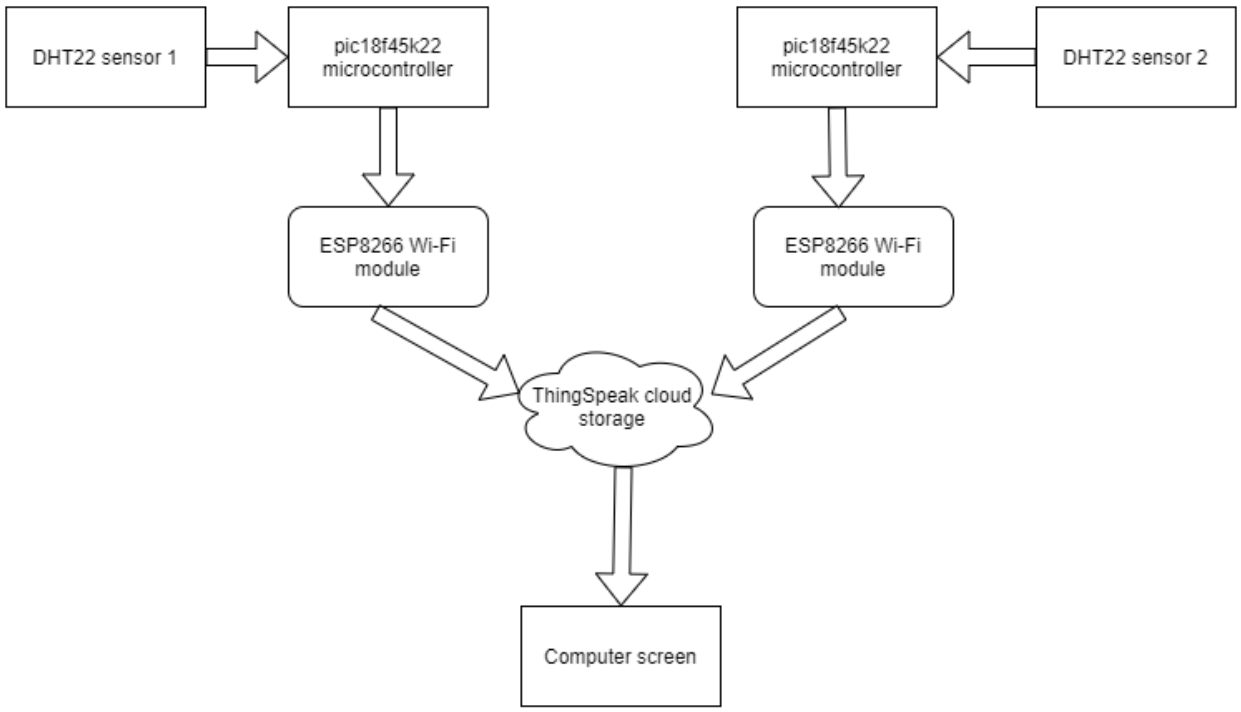


Figure 1. Block diagram of the system

The given circuit diagram describes the connection between the components used in the project. Sensor DHT22 and ESP866 are connected with the main processor PIC18F45K22.The communication between ESP866 and PIC18F45k22 controller will be serial communication. For serial communication, an ESP866 modules RX pin is connected with the TX pin and this TX pin is connected with the RX pin of PIC18F45K22 controller. An ESP866 module and a PIC18F45k22 controller both are connected with same ground. Data pin of DHT22 is connected with the pin no. 33 of microcontroller. The sensor DHT22 will send data through data line to the controller. Vdd pin of a sensor is connected to the positive supply through 4.7kOhm register. PICKIT module will provide 5v power supply to the circuit.

2.3 Circuit Diagram

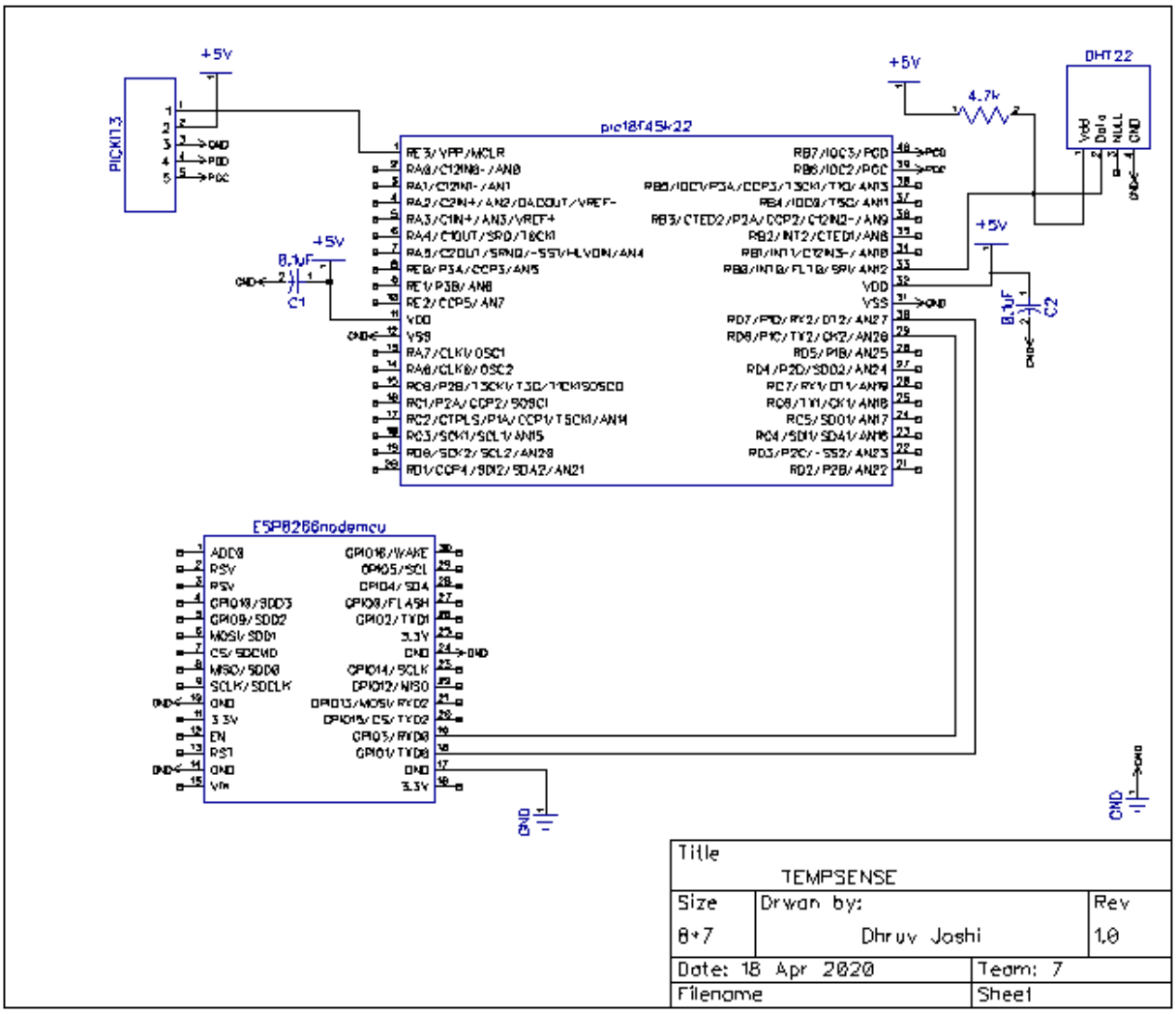


Figure 2. Circuit diagram of the system

The given circuit diagram describes the connection between the components used in the project. Sensor DHT22 and ESP866 are connected with the main processor PIC18F45K22.The communication between ESP866 and PIC18F45k22 controller will be serial communication. For the serial communication, ESP866 modules RX pin is connected with the TX pin and TX pin is connected with RX pin of PIC18F45K22 controller. ESP866 module and PIC18F45k22 controller both are connected with same ground. Data pin of DHT22 is connected with the pin no. 33 of microcontroller. The sensor DHT22 will send data through data line to the controller. Vdd pin of Sensor is connected to the positive supply through 4.7kOhm register. Picket module will provide 5v power supply to the circuit.

2.4 Constraints

As a team, we were not able to find the suitable model of a sensor as per weather conditions in Canada. As the temperature may vary between -40°C to +35°C. After studying datasheets of different models of temperature and humidity sensors, TempSense found one model of sensor DHT22 which is having -40°C to 80°C, which was perfect for our requirements. TempSense selected the same model of the sensor in the project.

The team was not able to find the datasheet of DHT22 in English language. The language of datasheet is Chinese. For the proper coding of the program for the connection of DHT22 to micro-controller pic18f45k22, we required basic information of the DHT22. For coding of the sensor, Professor Chris Talbot helped us.

There was some problem while connecting Wi-fi module ESP866 with the micro-controller. TempSense used AT Command for the configuration and making the connection between ESP866 and controller.

Currently, in the pandemic situation, TempSense faced some issue regarding the access of the laboratory of the collage. As the laboratory contains all the project related equipment, TempSense found problems related to the integration of devices.

3. Results

3.1 Website output

Figure 3. All 4 readings with the graphical demonstration

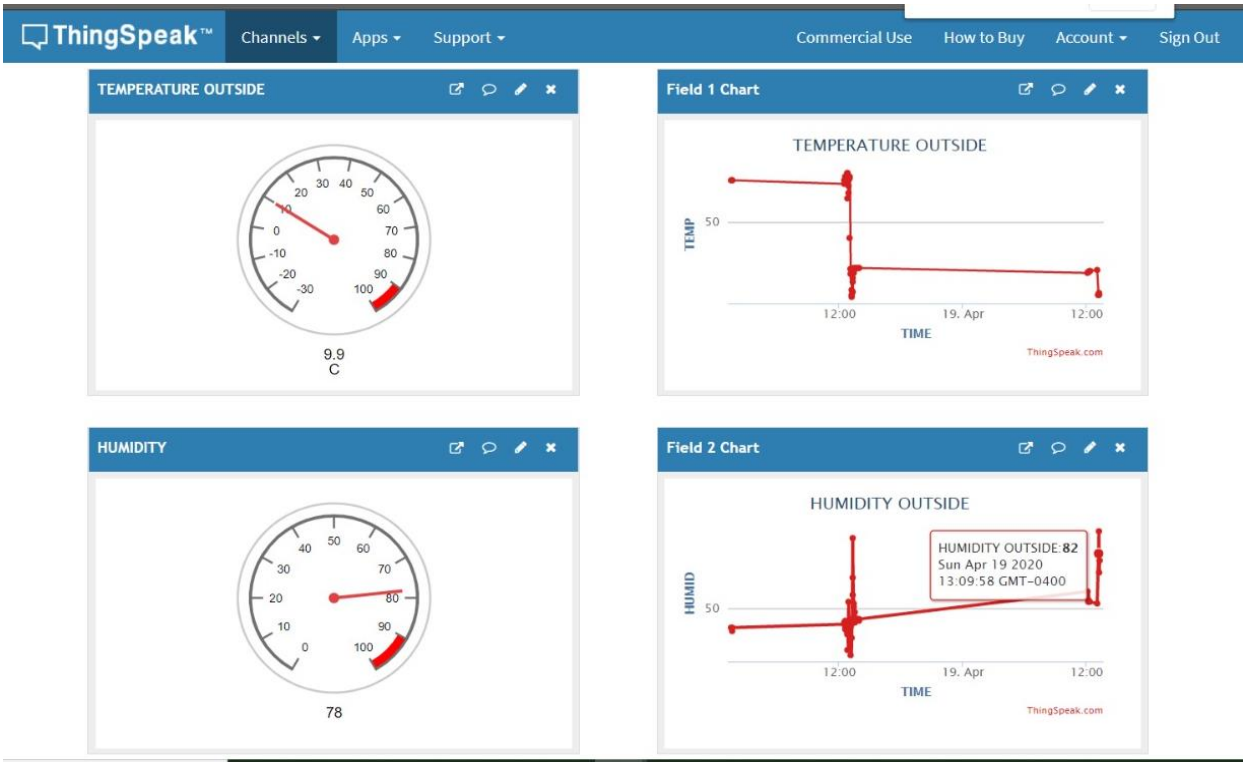


Figure 4. Real time readings of the system

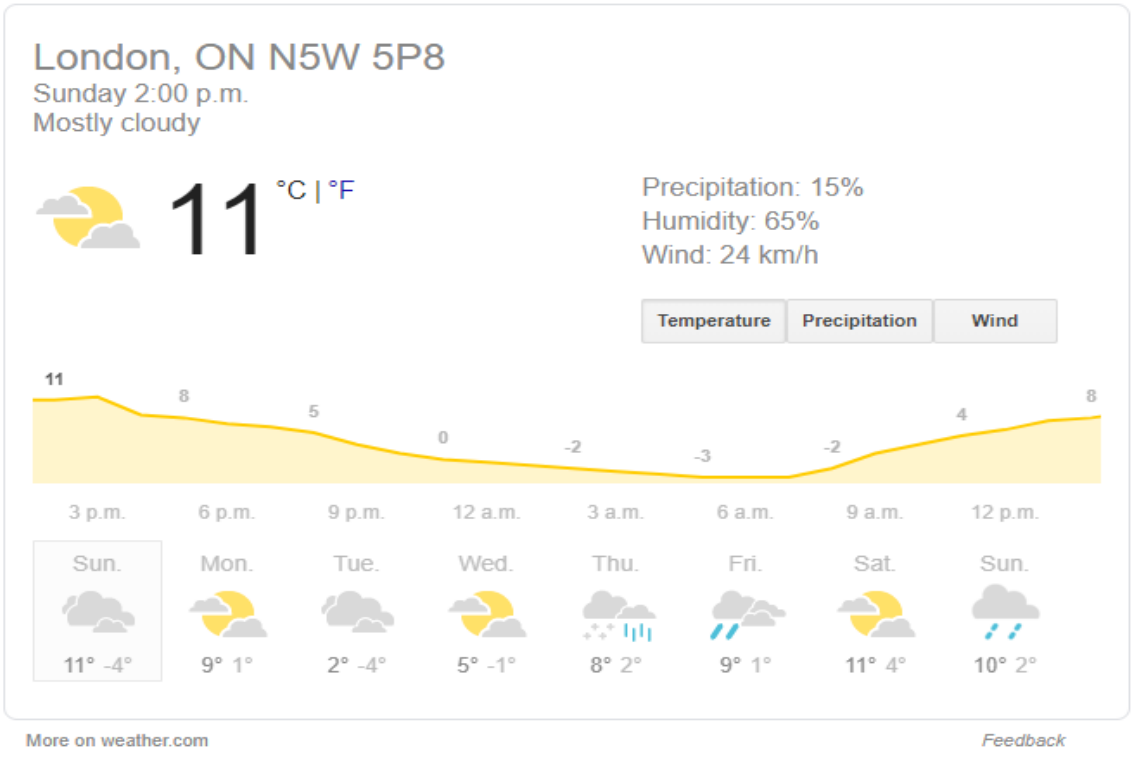


Figure 5. Google temperature results

The above windows represent the data provided by the sensor. There are two circuits to take the input readings. One of the circuits is connected within the house in the controlled atmosphere. Another circuit is kept outside the house in the natural atmosphere. As shown in the first figure the data obtained is the outer atmosphere and the temperature detected was 10.45 °C and 71% humidity. At that particular time the quantities inside the house were 20.9 ℃ and 45% humidity. For better precision, we double checked the data and took the readings again. The temperature outside the house this time was 9.9 ℃ because of the rainy weather the humidity became 78%. When we compared the data obtained from google earth, the temperature in the region was 11℃ and 65% humidity present in the air. This clearly suggest that the data obtained is much rawer and more precise for the designated region and provides accurate results for both quantities.

As one can see, from the above photos, there are 4 gauges on the screen. Out of these 4 gauges, 2 gauges shows temperature data from both the sensors and other 2 gauges show humidity data from both the sensors. Limits for temperature is -30°C to 100°C and for humidity it is set at 0 to 100%. Gauge form will be more convenient for the user than the display. The graphs next to the gauge shows the trend for 24 hours. User can hover around the graph for seeing the particular time readings. When the user goes to a specific point on the graph, they can see the day, date, time and the time zone when the readings have been taken.

3.2 Prototype

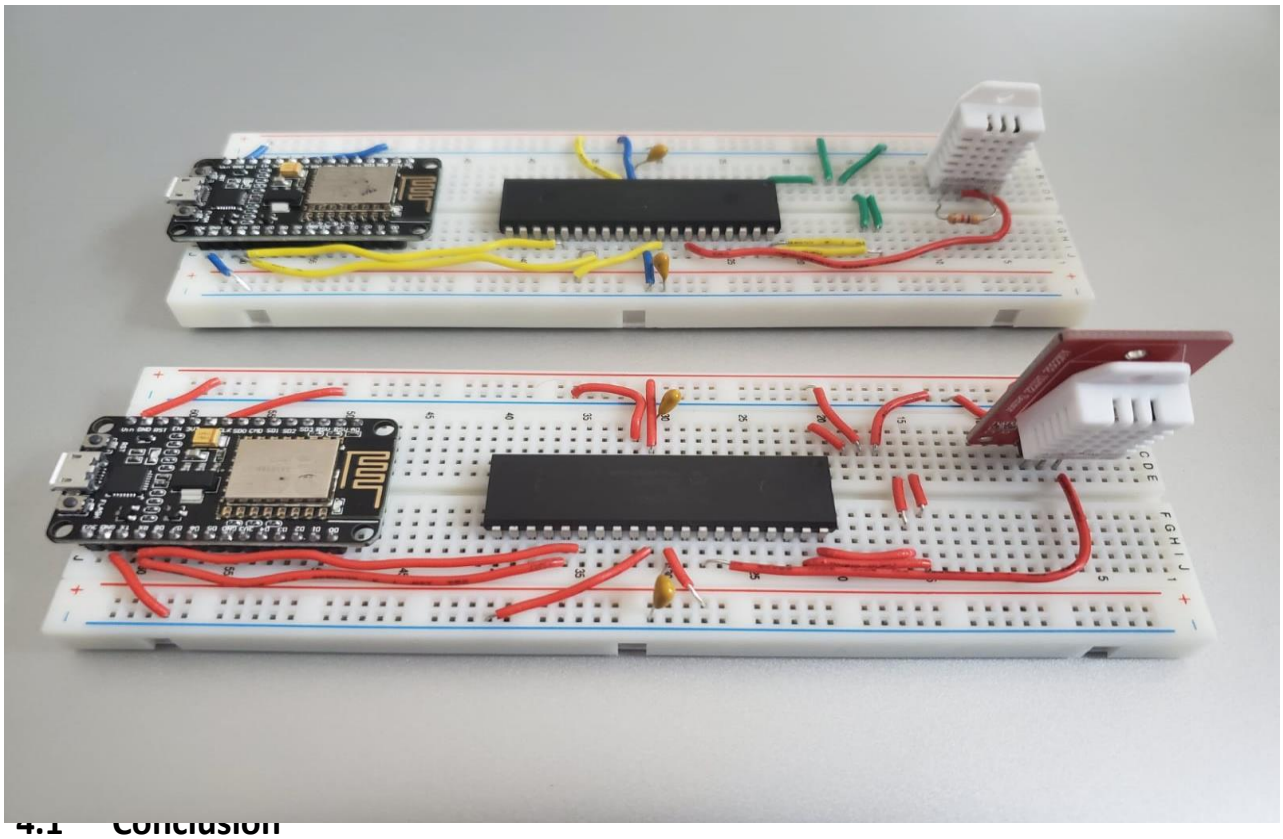


Figure 6. Prototype

4.1 Conclusion

the datasheet was available only in Chinese language and we faced few problems in interfacing this sensor. But with the help of Professor Chris Talbot, we were able to interface the sensor with our main circuit. We faced the similar problem while connecting the ESP8266 module, where Professor Talbot suggested to use AT commands. This was a perfect solution to our problem. The data is fetched and the output is obtained. In the end, both the team collaboratively worked on HMI and finally we got the solution in the form of 'Thingspeak'. Our HMI works properly and it is able to show the graphical and numerical data as per user's requirements. The data will be saved for 24 hours and it will be demonstrated graphically.

4.2 Recommendations

TempSense recommends that in the future, this project can be implemented on a large scale. For further development, we can make a mobile application for the same purpose. In this project, feedback was not provided to the sensor. In future, we can integrate heating, ventilation, and the air conditioning system of the house or any other place with this system for providing feedback remotely.

We should make necessary changes for showing a different large number of location's readings on the same screen.

5. References

NodeMCU v3¶. (n.d.). Retrieved from https://docs.zerynth.com/latest/official/board.zerynth.nodemcu3/docs/index.html

Adafruit Industries. (n.d.). DHT22 temperature-humidity sensor extras. Retrieved from https://www.adafruit.com/product/385

Strachan, & Sanders. (1989, March 1). Damp housing and childhood asthma; respiratory effects of indoor air temperature and relative humidity. Retrieved from https://jech.bmj.com/content/43/1/7.short

Liu, T. (n.d.). Digital-output relative humidity & temperature sensor/module DHT22 (DHT22 also named as AM2302). Retrieved April 15, 2020, from https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf

ESP8266 AT Instruction Set. (n.d.). Retrieved April 14, 2020, from https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf

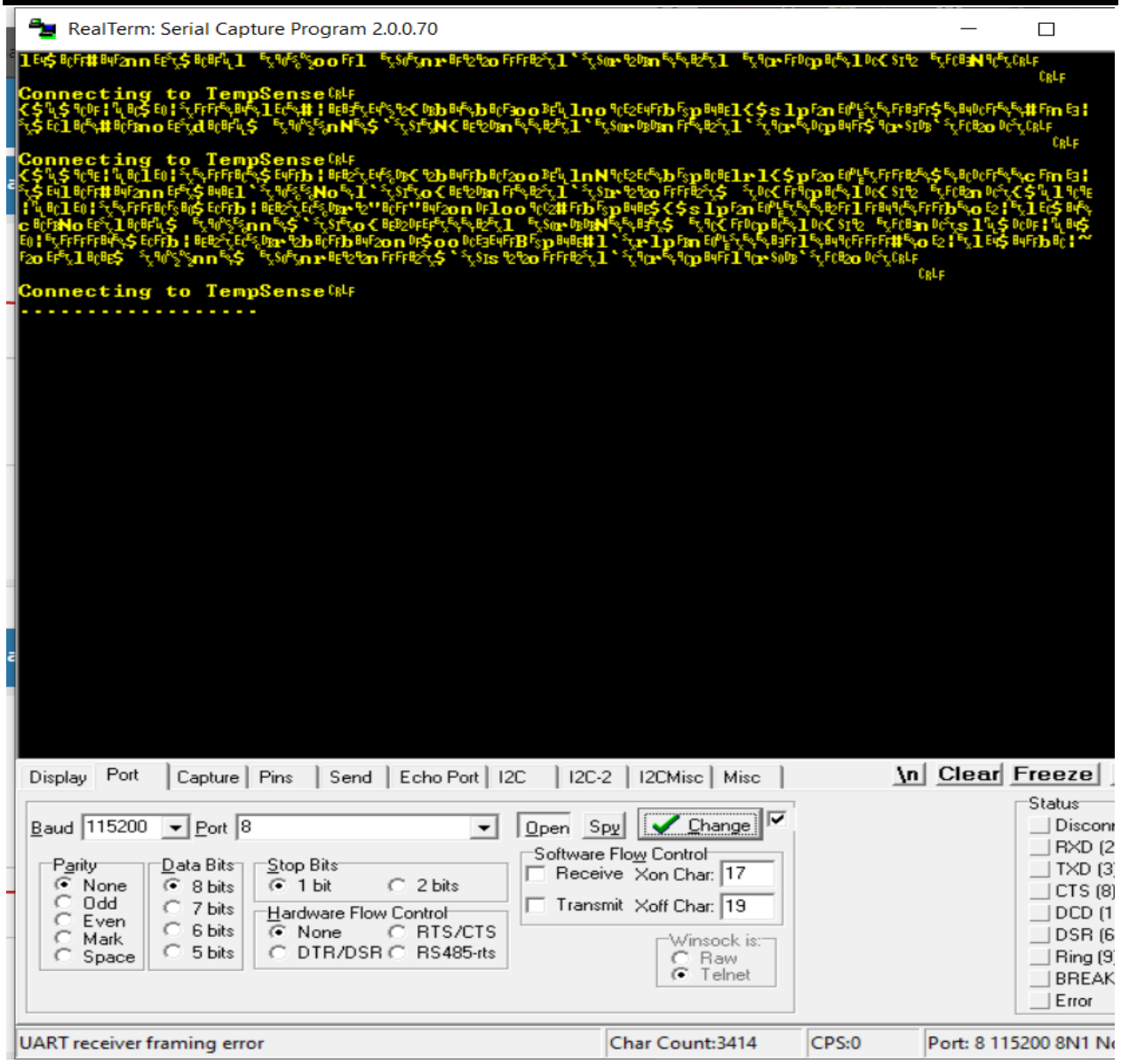
28/40/44-Pin, Low-Power, High-Performance Microcontrollers with XLP Technology. (n.d.). Retrieved April 13, 2020, from https://ww1.microchip.com/downloads/en/DeviceDoc/40001412G.pdf

Appendix I: Bill of material

Sr.No.	Qty.	Value	Product	Description	Manufacturer Part #	Manufacturer	Distributor Part #	Distributor	Unit Price	Link to datasheet
1	2	Temp: -40 to 80 degree celcious Humidity: 0 to 100 %	DHT22	Humidity, Temperature Sensor Gravity Platform Evaluation Expansion Board	385	Adafruit Industries LLC	1738-1039-ND	Digikey	$9.87	https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0137_Web.pdf
2	2		ESP8266	Transceiver; 802.11 b/g/n (Wi-Fi, WiFi, WLAN)	2491	Adafruit Industries LLC	1528-1530-ND	Digikey	$11.29	https://components101.com/sites/default/files/component_datasheet/ESP12E%20Datasheet.pdf
3	2		PIC18F45K22	PIC PIC® XLP™ 18K Microcontroller IC 8-Bit 64MHz 64KB (32K x 16) FLASH 40-PDIP	PIC18F45K22-I/P	Microchip Technology	PIC18F46K22-I/P-ND	Digikey	$4.53	http://ww1.microchip.com/downloads/en/DeviceDoc/40001412G.pdf
4	2	4.7 kOhms	CFR-25JB-52-4K7	4.7 kOhms ±5% 0.25W, 1/4W Through Hole Resistor Axial Carbon Film	CFR-25JR-52-4K7	Yageo	4.7KQBK-ND	Digikey	$0.16	https://www.yageo.com/upload/media/product/products/datasheet/lr/Yageo_LR_CFR_1.pdf

Figure 7. Bill of materials

Appendix II: ESP8266 realterm configuration



The above screen displays the real term window screenshot. One can see that it is showing the process to connect with the TempSense named mobile hotspot. With the help of this software we can see the details of the ESP8266 module such as connection status.

Appendix III: Code for the microcontroller

```
                        //*** TEMPSENSE™ ***
/*****************************************************************************
*************
        File Name:      TestReport1
        Author:         PLad
        Date:           27 Feb, 2020
        Modified:       CTalbot
        © Fanshawe College, 2020

        Description: This program will be used for fetching the temperature and humidity values
                from the dht22 sensor. As the output of the program, the values will be
                    shown on the thingspeack channel to display the data for the temperature and
humidity.

*****************************************************************************
************/
/*Preprocessor ==========================================================*/

#include "pragmas.h"
#include <adc.h>
#include <stdlib.h>
#include <delays.h>
#include <stdio.h>
#include "usart.h"
#include <p18f45k22.h>



// Constants ------------------------------------------------------------------

#define TRUE            1
#define FALSE           0
#define DHTPIN                  PORTBbits.RB0
#define DHTDIR                  TRISBbits.TRISB0

#define SIZE    11

#define OUTPUT                  0
#define INPUT           1
#define MAXDATA                 32
#define MAXCS           8
#define LIMIT           26
#define MAXSIZE   50
#define SSID      TempSense
#define PASS      12345678
```

```
#define API       IDU0J5BBIAEVUG2V
#define T0FLAG        INTCONbits.TMR0IF

// Global Variables ---------------------------------------------------------------


int Temp,humd;
int i;
char str[MAXSIZE];
int index = 0, temp = 0, humd = 0;
int Time=0;


// Functions ----------------------------------------------------------------------
/*** set_osc_p18f45k22_4MHz: ******************************************************
Author:         PLad
Date:           22 Jan, 2020
Modified:
Desc:           Sets the internal Oscillator of the Pic 18F45K22 to 4MHz.
Input:          None
Returns:        None
*********************************************************************************/
void set_osc_p18f45k22_4MHz(void)
{
        OSCCON  =       0x52;
        OSCCON2 =       0x04;
        OSCTUNE =       0x80;

        while (OSCCONbits.HFIOFS != 1);
}
//eo: set_osc_p18f45k22_4MHz:: ************************************************



/*** portConfig: ****************************************************************
Author:         PLad
Date:           21 Feb 2020
Modified:
Desc:           Initializes ports according to project needs.
Input:          None
Returns:        None
*********************************************************************************/
void portConfig(void)
{
        //PORT A

                ANSELA = 0x03;
                LATA   = 0x00;
```

```
                TRISA  = 0xFF;

        //PORT B

                ANSELB = 0x00;
                LATB   = 0x01;
                TRISB  = 0x00;

        //PORT C

                ANSELC = 0x00;
                LATC   = 0x00;
                TRISC  = 0x00;

        //PORT D

                ANSELD = 0x00;
                LATD   = 0x00;
                TRISD  = 0xF0;    // set the out put / input pin 33

}
//eo: portConfig:: ************************************************************

/*** serialConfig: ************************************************************
Author:         pLad
Date:           22 Feb, 2020
Modified:
Desc:           Serial port configuration
Input:          None
Returns:        None
*******************************************************************************/
void serialConfig(void)
{
        SPBRG1= 8;              // set the baud rate value at 115200
        TXSTA1= 0x26;           // set the tx
        RCSTA1= 0X90;           // set the rx
        BAUDCON1=0X40;          // select the baud config
}
//eo: serialConfig::************************************************************


/*** TMR0Config ***************************************************************
Author:         plad
Date:       22/01/2020
Modified:
Desc:           configure the timer 0
Input:          None
Returns:        None
```

```
*************************************************************************/
void TMR0Config(void)
{
        T0CON=0x91;         // set the timer for the one second
        TMR0H=0x0B;         // set the timer0 high bits
        TMR0L=0xDC;         // set the timer0 low bits
        T0FLAG=FALSE;
}
//eo: TMR0Config: ************************************************************



/*** TMR0Reset ***************************************************************
Author:         plad
Date:       22/01/2020
Modified:
Desc:           reset the timer 0
Input:          None
Returns:        None
*************************************************************************/
void TMR0Reset(void)
{
        T0FLAG=FALSE;      //making the timer0 flag 0
        TMR0H=0x0b;        // set the timer0 high bits
        TMR0L=0xdc;       // set the timer0 low bits
}
//eo: TMR0reset: ************************************************************




/*** startSignal**************************************************************
Author:         PLad
Date:           24 Feb, 2020
Modified:       CTalbot
Desc:
Input:          None
Returns:        None
*************************************************************************/
void startSignal(void)
{
        DHTDIR = OUTPUT;        // Output
        DHTPIN = FALSE;                 // Go low
        Delay1KTCYx( 1 );       // Delay 20ms
        DHTPIN = TRUE;
        Delay10TCYx( 4 );       // Delay 40us
        DHTPIN = FALSE;
```

```
	DHTDIR = INPUT;

	// End of start sequence
	// DHT22 should take over at this point and pull the line
	// Ack signal will delay 80 + 80 us

	while( !DHTPIN );
//	while( DHTPIN );

	// Get data - checkResponse
}
//eo: startSignal:: ************************************************************



/*** checkResponse ***************************************************************
Author:		CTalbot
Date:		24 Feb, 2020
Modified:	CTalbot
Desc:
Input:		None
Returns:	None
*********************************************************************************/
void checkResponse()
{
	long data = 0;
	char checksum = 0;
	for( index=0; index< MAXDATA; index++)
	{
		while( !DHTPIN );
		TMR0H=0;
		TMR0L=0;
		while( DHTPIN );

		if( TMR0L > LIMIT )
		{
			data = (data<<1) + 1;
		}
		else if( TMR0L < LIMIT )
		{
			data = (data<<1);
		}
	}
	for( index=0; index< MAXCS; index++)
	{
		while( DHTPIN );
		TMR0H=0;
		TMR0L=0;
```

```
                while( !DHTPIN );

                if( TMR0L > LIMIT )
                {
                        checksum = (checksum<<1)+ 1;
                }
                else if( TMR0L < LIMIT )
                {
                        checksum = (checksum<<1);
                }
        }
        temp = data;
        humd = (data>>16) ;

        Nop();
        DHTDIR = OUTPUT;
        DHTPIN = TRUE;

        printf("H: %i.%i\n\rT: %i.%i\n\r", humd/10, humd%10, temp/10, temp%10 ); // collect the data
of quantity and send it to the serial output
        Nop();
}
//eo: checkResponse:: ************************************************************




/*** readData ***************************************************************
Author:         PLad
Date:           24 Feb, 2020
Modified:
Desc:
Input:          None
Returns:        None
*****************************************************************************/
char readData(char *ptr)
{
        char i;
        for(i=0; i<=7; i++)
        {
                TMR0H=0;
                TMR0L=0;
                while(!DHTPIN)
                {
                        if(TMR0L>100)
                        {
                                return 1;
                        }
```

```
                }
                TMR0H=0;
                TMR0L=0;
                while(DHTPIN)
                {
                        if(TMR0L>100)
                        {
                                return 1;
                        }
                        else if(TMR0L>50)
                        {
                                *ptr |= (1<<(7-i));
                        }
                }

        }
}
//eo: readData:: ************************************************************


/*** display ***************************************************************
Author:         PLad
Date:           24 Feb, 2020
Modified:
Desc:
Input:          None
Returns:        None
*****************************************************************************/
char display()
{
        startSignal();
}
//eo: display:: ************************************************************

/*** clrString *************************************************************
Author:         PLad
Date:           24 Feb, 2020
Modified:
Desc:
Input:          None
Returns:        None
*****************************************************************************/
void clrString()
{
        char index=0;
        for(index=0;index<MAXSIZE;index++)
        {
                str[index]=0;
```

```
        }
}
//eo: clrString:: ************************************************************

/*** initialESP ***************************************************************
Author:         PLad
Date:           24 Feb, 2020
Modified:
Desc:
Input:          initialize the esp module and configure with the local wifi
Returns:        None
******************************************************************************/
void initialESP()
{
        clrString();
        sprintf(str,"AT");          // send first at command         /
        puts1USART(str);
        Delay1KTCYx( 100 );
        clrString();
        sprintf(str,"ATE0");          // set at command to enable the echo command
        puts1USART(str);
        Delay1KTCYx( 100 );
        clrString();
        sprintf(str,"AT+CWMODE=3");       // at command to set the wifi module on both station and
host mode
        puts1USART(str);
        Delay1KTCYx( 100 );
        clrString();
        sprintf(str,"AT+CIPMUX=1");      // to set the esp with the multiple devices
        puts1USART(str);
        Delay1KTCYx( 100 );
        clrString();
        sprintf(str,"AT+CWJAP=");         // to set/connect the esp with the local network.
        sprintf(str,"TempSense,");         // name of the wifi
        sprintf(str,"12345678");          // password to connect the wifi
        puts1USART(str);
        Delay1KTCYx( 100 );
}
//eo: clrString:: ************************************************************




/*** initializeSystem: ********************************************************
Author:         Plad
Date:           26 Feb, 2020
```

```
Modified:       None
Desc:           All the required function for initiation are called within this function.
Input:          None
Returns:        None
*****************************************************************************/
void initializeSystem(void)
{
        set_osc_p18f45k22_4MHz();
        portConfig();
        serialConfig();
        TMR0Config();
        TMR0Reset();
        T0CON = 0x90;
}
// eo initializeSystem::*****************************************************



/*--- MAIN FUNCTION -----------------------------------------------------------
-----------------------------------------------------------------------------*/

void main(void)
{
        initializeSystem();
        initialESP();
        while(1)
        {
                if(Time>= 120)
                {
                        Time=0;
                        startSignal();
                        checkResponse();
                        clrString();
                        sprintf(str,"AT+CIPSTART=4,");    // to set the tcp and api link for the thingspeak
website.
                        sprintf(str,"TCP,");            //  to set the tcp mode
                        sprintf(str,"api.thingspeak.com");   // send the api of thingspeak
                        printf("%s",str);
                        puts1USART(str);
                        Delay1KTCYx( 100 );
                        clrString();
                        sprintf(str,"AT+CIPSEND=80");        // send at command that we can send 80
chacters
                        printf("%s",str);
                        puts1USART(str);
                        Delay1KTCYx( 100 );
```

```
                        sprintf(str,"GET                                                    /update?api_key=
IDU0J5BBIAEVUG2V&field1=%i.%i",temp/10, temp%10); // send the field data of temperature.
                        printf("%s",str);
                        puts1USART(str);
                        sprintf(str,"AT+CIPCLOSE");
                        printf("%s",str);
                        puts1USART(str);
                        Delay1KTCYx( 100 );
                        clrString();
                        sprintf(str,"GET                                                    /update?api_key=
IDU0J5BBIAEVUG2V&field2=%i.%i",humd/10, humd%10);   // send the field data of humidity.
                        printf("%s",str);
                        puts1USART(str);
                        sprintf(str,"AT+CIPCLOSE");
                        printf("%s",str);
                        puts1USART(str);
                        Delay1KTCYx( 100 );
                        Time++;
                        TMR0Reset();
                }
        }//eo while
}// eo main
```