Noor Blum

CPE301  UCLV

Weather Shield

CPE 301 EMBEDDED SYSTEM DESIGN S 2015

Contents

[Goal: 2](#_Toc450292316)

[Deliverables 2](#_Toc450292317)

[Literature Survey: 2](#_Toc450292318)

[Block Diagram 3](#_Toc450292319)

[Flowchart 4](#_Toc450292320)

[4](#_Toc450292321)

[Working 4](#_Toc450292322)

[ Hardware 5](#_Toc450292323)

[ Software 5](#_Toc450292324)

[ Cloud 5](#_Toc450292325)

[Components 5](#_Toc450292326)

[Schematic 6](#_Toc450292327)

[PCB Design 6](#_Toc450292328)

[Implementation: 7](#_Toc450292329)

[Code 7](#_Toc450292330)

[References 7](#_Toc450292331)

**TITLE: Weather Shield using ESP8266 , DHT11 & ATmega328**

# Goal:

* Measure Temperature and humidity
* Send data on Cloud
* Monitor from Android Phone and PC

Deliverables:

* Source Code
* Hardware Schematics
* Android APK ( Generated )
* PCB Design Files
* Report Files

# Literature Survey:

We are always curious to measure our local weather irrespective of weather.com and accuweather.com, it always amused me to measure current room temperature and garden temperature, it is great fun to see and create a log of temperature on my cellphone with the help of cloud.

So this weather shield takes humidity and temperature readings from sensor DH11 and transfer the reference voltages to ADC of ATmega328p, now Uart of ATmega328 will send the data to ESP8266, which will send data to free cloud service, and our android app and web browser will show the current temperature and humidity values with Pass Graph.

# Block Diagram

**Cloud**

**ATMega328P**

**ESP8266**

DH11

Sensor

**OPTIONAL**

**(Only for Debugging)**

16x2 Matrix LCD

**Cloud**



# Flowchart

Initialize ADC

Initialize Port

Start

Send Data to Cloud

Read ADC Values

Initialize Wi-Fi

# 

# Working

Working of Weather Shield is simple enough to understand majorly there are 3 Parts Hardware, Software and Cloud, Basically Work of Hardware is to generate the value and DH11 will generate the value and Send to ADC of ATmega328P , then firmware inside ATmega328P will collect the data from ADC Port , Convert it into Actual readable format then Send the Data to TX of wifi module ,

Now wifi module will interact with internet and free cloud service, it will send data to cloud service and now device where you want to see, I have created one basic app which will see the cloud URL and display the picture.

## Hardware

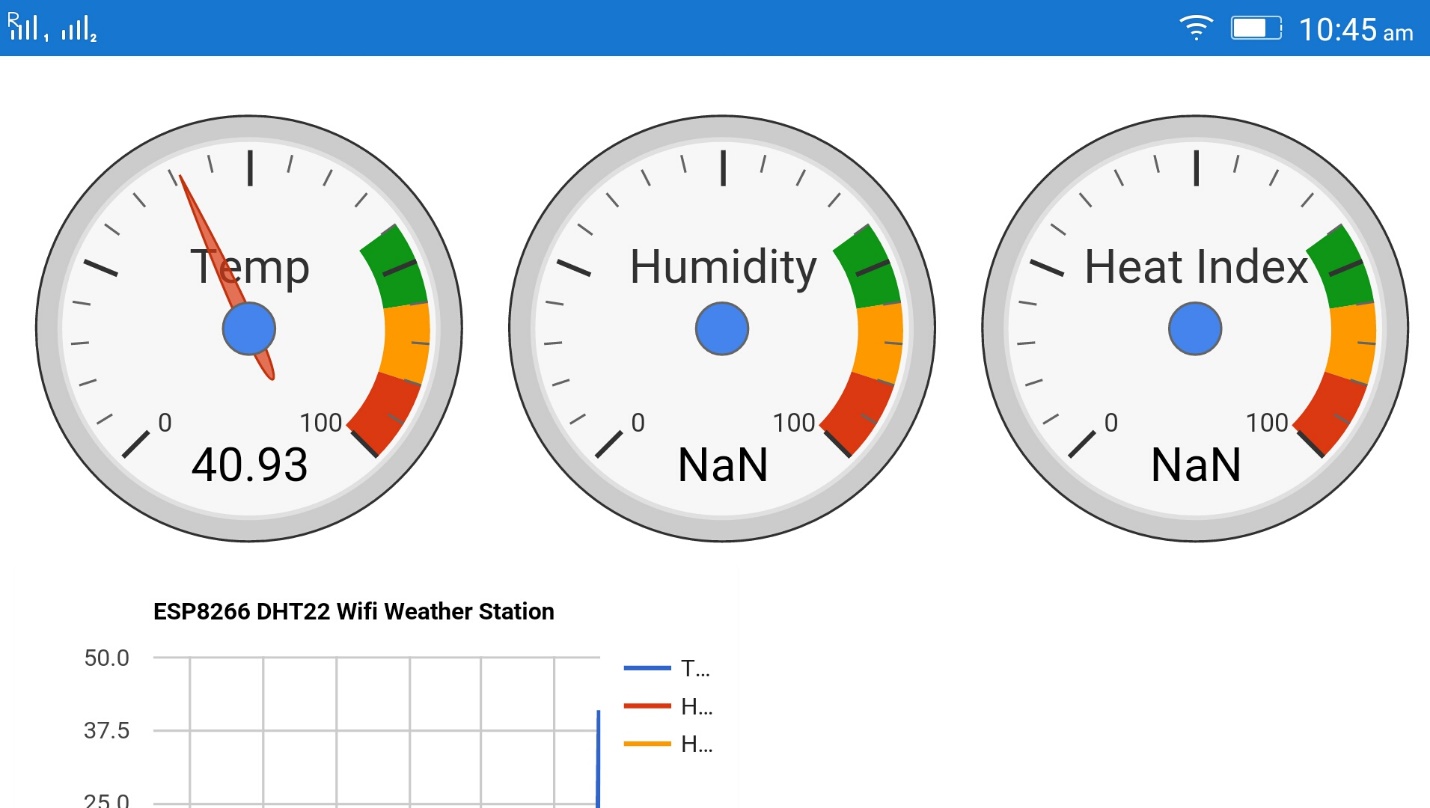
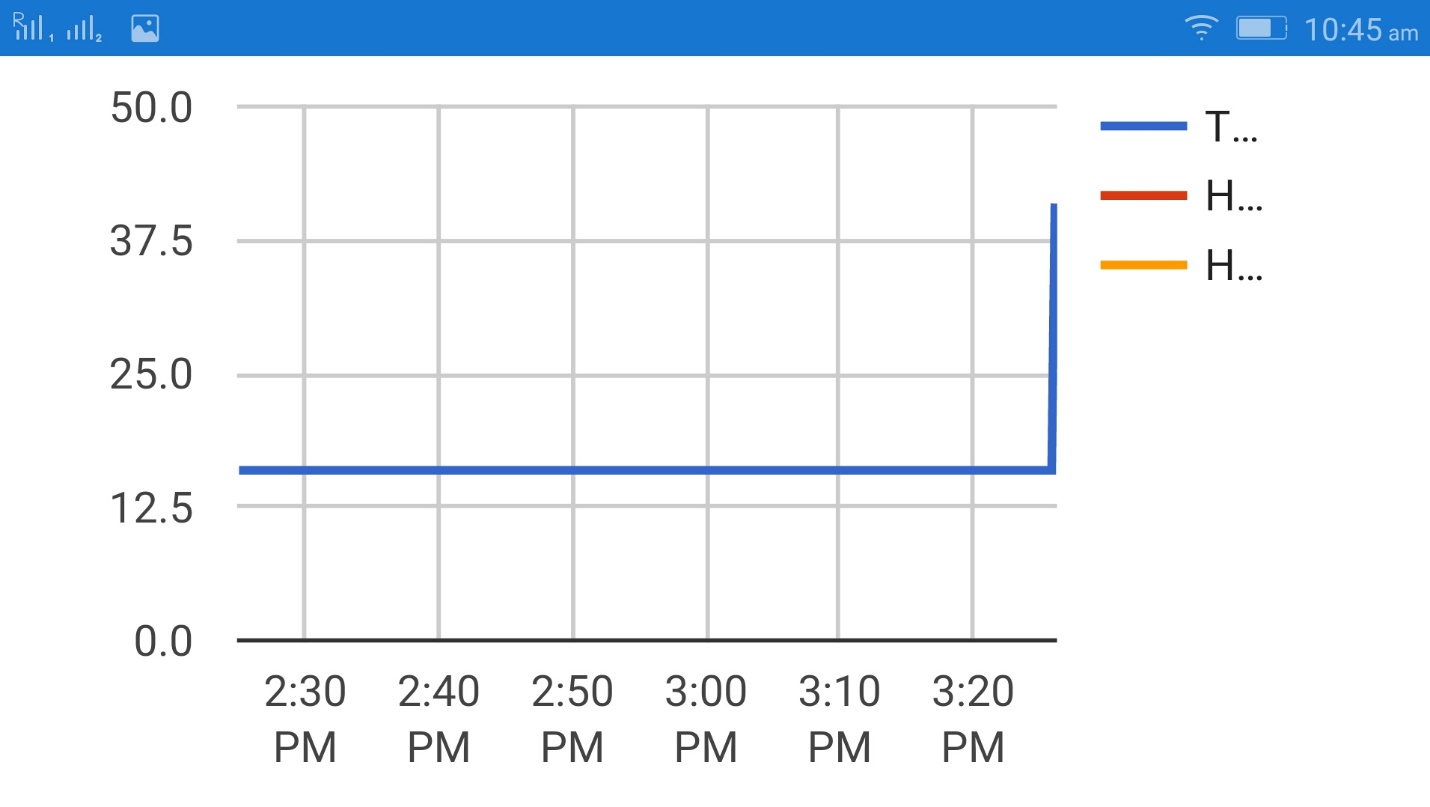
Atmega328P is main hardware of this project , basically it configure ports and convert data into actual ratings , it has various IO ports , in portd we have connected 16x2 LCD just to debug the data taken by sensor ,it can be removed in production version of Weather shield.

## Software

Software has major role in this project, firmware and other is app, the esp8266 will be configured through firmware and we are sending the data through the same firmware. So firmware plays an important role.

## Cloud

We have used free cloud service <http://data.sparkfun.com> , it has provided basic interface which will generate the cloud data in the graphical format



# Components

**Explain the main characteristics, interface, and limitation of the components used**

**Main Characteristic:**

In this project the main components are as below:

Microcontroller (ATMEGA328P)

* High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller
* 32KBytes of In-System Self-Programmable Flash program memory
* 1KBytes EEPROM
* 2KBytes Internal SRAM
* 8-channel 10-bit ADC
* Programmable Serial USART

Temperature Sensor (DTH11)

* Measurement Range - 20-90%RH 0-50 ℃
* Humidity Accuracy - ±5％RH
* Temperature Accuracy - ±2℃
* Resolution -1

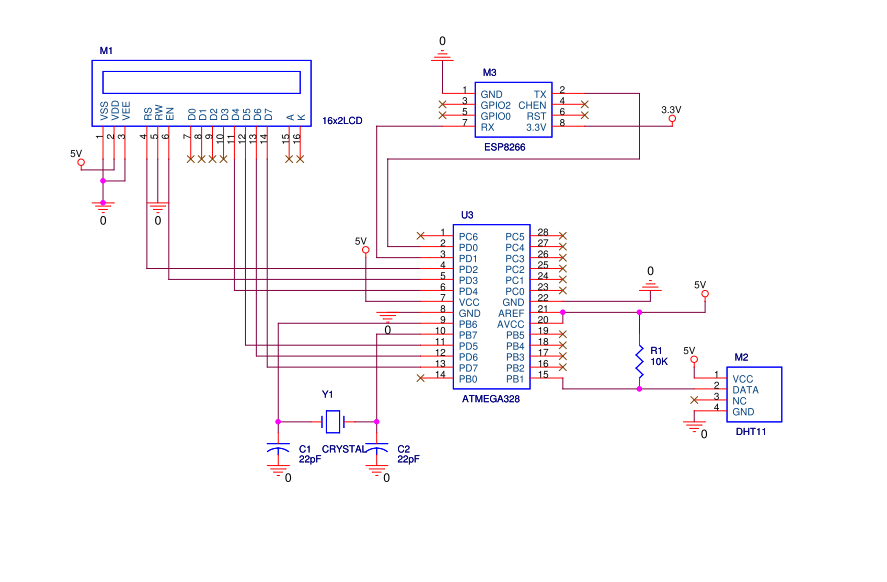
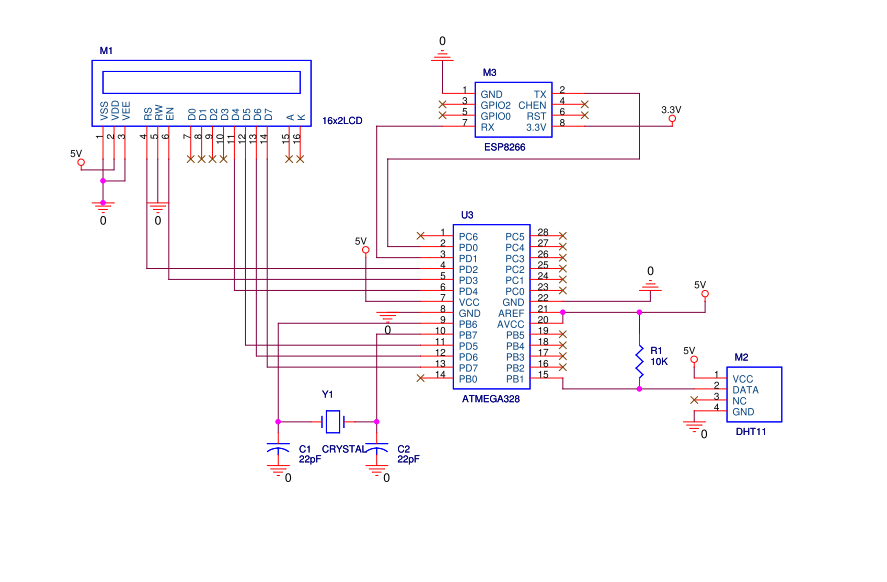
WiFi Module ( ESP8266 )

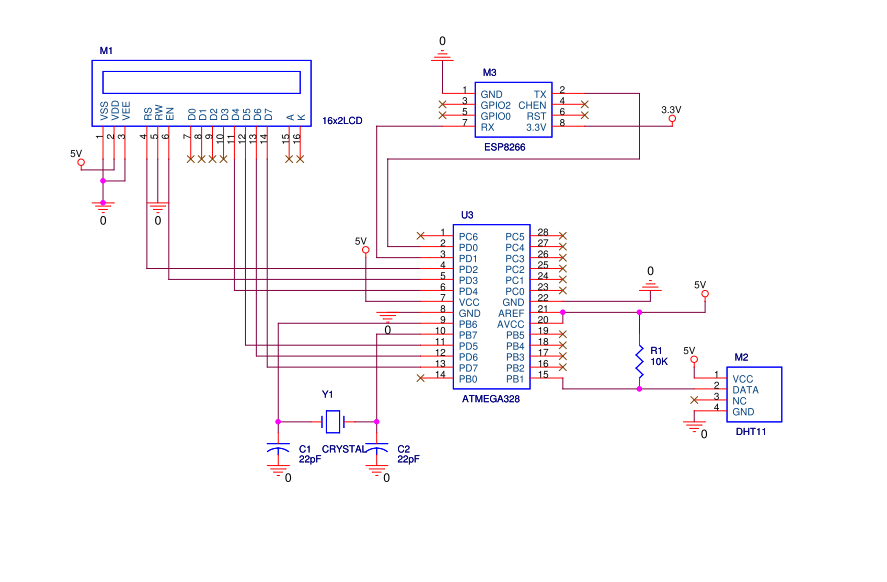
* 802.11 b/g/n
* Integrated low power 32-bit MCU
* Integrated 10-bit ADC
* Integrated TCP/IP protocol stack
* WiFi 2.4 GHz, support WPA/WPA2

LCD Display (16x2 LM016L)

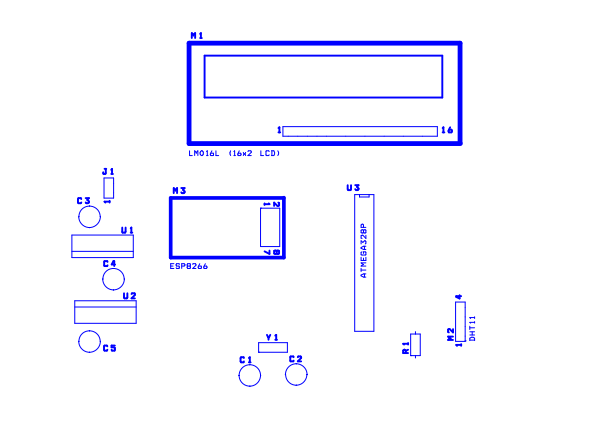
* Mechanical dimension – 80mm x 36mm x 13.5mm
* Power Voltage – 7V max
* Operating Temperature – 0 - 50 C

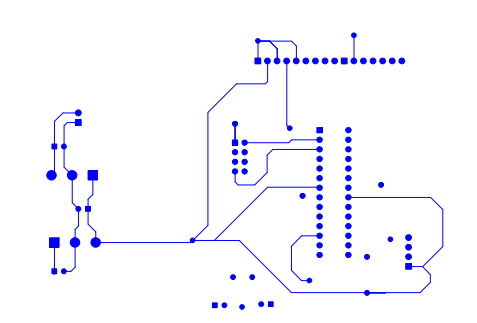
# Schematic

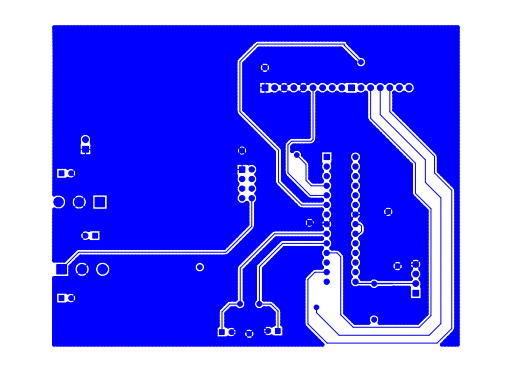


****

# PCB Design







# Implementation

* Create a C Code which communicate with ESP8266 , take temperature from DTH11 and Display the commands on LCD Screen
* Connect the hardware as Schematic
* Now C code will take data from Sensor DTH11 and upload to cloud.
* Change the wifi SSID and password into code as per your requirement.
* Log the data and see the results on HTML Webpage and Andoird App

# Snapshots/Screenshot/Videos

[https://github.com/blumn/weathershield/1.jpg](https://github.com/blumn/weathershield)

<https://github.com/blumn/weathershield/2.jpg>

<https://github.com/blumn/weathershield/3.jpg>

<https://github.com/blumn/weathershield/4.jpg>

<https://github.com/blumn/weathershield/5.jpg>

<https://github.com/blumn/weathershield/6.jpg>

<https://github.com/blumn/weathershield/7.jpg>

<https://github.com/blumn/weathershield/8.jpg>

<https://github.com/blumn/weathershield/videos.txt>

# 

# Code

#define F\_CPU 16000000ul

#include <util/delay.h>

#define rs PD2

#define en PD3

void cmnd()

{

PORTD&=(~(1<<rs));

PORTD|=(1<<en);

*\_delay\_ms*(5);

PORTD&=(~(1<<en));

}

void lcdcmd(char ch)

{

PORTD=ch & 0xF0;

cmnd();

PORTD=(ch<<4) & 0xf0;

cmnd();

}

void lcd\_init()

{

lcdcmd(0x02);

lcdcmd(0x28);

lcdcmd(0x0e);

lcdcmd(0x01);

}

void data()

{

PORTD|=(1<<rs);

PORTD|=(1<<en);

*\_delay\_ms*(5);

PORTD&=(~(1<<en));

}

void lcddata(char ch)

{

PORTD=ch & 0xF0;

data();

PORTD=(ch<<4) & 0xf0;

data();

}

void lcdprint(char \*str)

{

while(\*str)

{

lcddata(\*str);

str++;

}

}

/\* for amtega16 \*/

/\*void serialbegin(unsigned int BAUD, unsigned int FOSC)

{

// unsigned int MYUBRR=((((FOSC\*1000000)/16))/BAUD)-1;

//UBRRH = (MYUBRR >> 8);

// UBRRL = 103;

UCSRB=0x18;

UCSRC=0x86;

UCSRB |= (1 << RXCIE);

UBRRL=103;

}

char serialread()

{

while(!(UCSRA & (1<<RXC)));

return UDR;

}

void serialwrite(unsigned char ch)

{

UDR=ch;

while(!(UCSRA & (1<<UDRE)));

}\*/

/\* for atmega328 \*/

void serialbegin(unsigned int BAUD, unsigned int FOSC)

{

unsigned int MYUBRR=((((FOSC\*1000000)/16))/BAUD)-1;

UBRR0H = (MYUBRR >> 8);

UBRR0L = MYUBRR;

UCSR0B |= (1 << RXEN0) | (1 << TXEN0); // Enable receiver and transmitter

UCSR0B |= (1 << RXCIE0); // Enable reciever interrupt

UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); // Set frame: 8data, 1 stp

}

char serialread()

{

while(!(UCSR0A & (1<<RXC0)));

return UDR0;

}

void serialwrite(unsigned char ch)

{

UDR0=ch;

while(!(UCSR0A & (1<<UDRE0)));

}

void serialprintln(char \*str)

{

serialprint(str);

serialprint("\r\n");

}

void serialprint(char \*str)

{

while(\*str)

{

serialwrite(\*str);

str++;

}

}

#include <avr/io.h>

#include <stdint.h>

#include <avr/interrupt.h>

//#include <string.h>

#include "serial\_header.h"

#include "lcd\_header.h"

//#define sensor PA0

#define sensor PB1

char \*strkey = "c2d40ab9";

static char postUrl[100];

char \*text,mytext[4];

unsigned char a = 0, b = 0,d = 0,t1 = 0,t2 = 0,

rh1 = 0,rh2 = 0,sum = 0;

char temperature[3];

char humidity[3];

int i=0;

char rec[100];

char ReceivedChar=0;

void StartSignal()

{

DDRB|= 1<<sensor; //Configure RD2 as output

PORTB&= ~(1<<sensor); //RD2 sends 0 to the sensor

*\_delay\_ms*(18);

PORTB|=1<<sensor; //RD2 sends 1 to the senso

DDRB&=~(1<<sensor);

PORTB&=~(1<<sensor);

}

void CheckResponse()

{

a = 0;

*\_delay\_us*(40);

while((PINB&(1<<sensor)));

if(!(PINB&(1<<sensor)))

{

*\_delay\_us*(80);

if (PINB&(1<<sensor))

{

a = 1;

*\_delay\_us*(80);

}

}

}

void ReadData()

{

for(b=0;b<8;b++)

{

while(!(PINB&(1<<sensor))); //Wait until PORTD.F2 goes HIGH

*\_delay\_us*(40);

if(!(PINB&(1<<sensor)))

d&=~(1<<(7-b)); //Clear bit (7-b)

else

{

d|= (1<<(7-b)); //Set bit (7-b)

while(PINB&(1<<sensor));

}

//Wait until PORTD.F2 goes LOW

}

}

void show()

{

if(a == 1)

{

// i=0;

ReadData();

rh1 =d;

ReadData();

rh2 =d;

ReadData();

t1 =d;

ReadData();

t2 =d;

ReadData();

sum = d;

if(sum == rh1+rh2+t1+t2)

{

lcdcmd(1);

text = "Temp: .0C";

lcdprint(text);

lcdcmd(192);

text = "Humidity: .0%";

lcdprint(text);

*sprintf*(temperature,"%d",t1);

lcdcmd(0x85);

lcdprint(temperature);

*sprintf*(humidity,"%d",rh1);

lcdcmd(0xc9);

lcdprint(humidity);

}

else

{

lcdcmd(1);

lcdprint("Check sum error");

}

}

else

{

lcdcmd(1);

lcdprint("No response");

lcdcmd(192);

lcdprint("from the sensor");

}

}

//ISR(USART\_RXC\_vect)

ISR(USART\_RX\_vect)

{

ReceivedChar = UDR0; // Read data from the RX buffer

rec[i++]=ReceivedChar; // Write the data to the TX buffer

}

void get\_ip()

{

char IP[16];

char ch=0,j=0;

char flag=0;

while(flag==0)

{

i=0;

serialprintln("AT+CIFSR");

*\_delay\_ms*(2000);

if(i>0)

{

for(j=0;j<i;j++)

{

lcdcmd(1);

lcdprint("Wait....");

if(rec[j]=='S' && rec[j+1]=='T' && rec[j+2]=='A' && rec[j+3]=='I' && rec[j+4]=='P')

{

j=j+6;

int n=0;

while(i!=j)

{

while(rec[j]!='+')

{

IP[n++]=rec[j++];

}

flag=1;

break;

}

}

}

}

}

lcdcmd(1);

lcdprint("IP:");

lcdprint(IP);

lcdcmd(192);

lcdprint("Port:");

lcdprint(80);

i=0;

*\_delay\_ms*(5000);

}

void send(char \*str, unsigned int time)

{

while(1)

{

int j=0,temp=0;

i=0;

serialprintln(str);

lcdcmd(1);

lcdprint(str);

for(int t=0;t<time+2000;t++)

*\_delay\_ms*(1);

//int lenth= strlen\_P(rec);

if(i>0)

{

for(j=0;j<i;j++)

{

if(rec[j]=='O' && rec[j+1]=='K')

{

lcdcmd(192);

lcdprint("OK");

temp=1;

*\_delay\_ms*(1000);

i=0;

break;

}

else if(rec[j]=='E' && rec[j+1]=='R' && rec[j+2]=='R' && rec[j+3]=='O' && rec[j+4]=='R')

{

lcdcmd(192);

lcdprint("Error");

*\_delay\_ms*(1000);

i=0;

}

}

}

if(temp==1)

break;

}

}

void connect\_wifi()

{

send("AT",1000);

send("AT+RST",5000);

send("AT+CWMODE=1",1000);

send("AT+CWQAP",1000);

lcdcmd(1);

lcdprint("Connecting WIFI");

send("AT+CWJAP=\"1st floor\",\"muda1884\"",10000);

}

void httpGet(char \* ip, char \*path, int port)

{

int resp;

port=80;

serialprint("AT+CIPSTART=\"TCP\",\"");

serialprint(ip);

serialprintln("\",80");

*\_delay\_ms*(1000);

send("AT+CIPSEND=112",2000);

serialprint("GET /");

serialprint(path);

serialprintln(" HTTP/1.0\r\n\r\n");

}

int main( void )

{

int z=0;

DDRD=0xfE;

DDRB=0xff;

DDRB|=1<<PB5;

lcd\_init();

serialbegin(9600,16); // baad rate and frequency in MHz

sei();

lcdprint("System Ready");

*\_delay\_ms*(1000);

while(1)

{

lcdcmd(1);

StartSignal();

CheckResponse();

show();

if(z==5)

{

connect\_wifi();

float temp=t1;

float humid=rh1;

long pressure=10;

char tempStr[8];

char humidStr[8];

char presStr[8];

*dtostrf*(temp, 5, 3, tempStr);

*dtostrf*(humid, 5, 3, humidStr);

*dtostrf*(pressure, 5, 3, presStr);

*sprintf*(postUrl, "sites/default/files/datalog/postData.php?temp=%s&humid=%s&pressure=%s&key=%s", tempStr, humidStr, presStr, strkey);

httpGet("www.hobbyist.co.nz", postUrl, 80);

*\_delay\_ms*(100);

serialprintln("AT+CIPCLOSE=0");

*\_delay\_ms*(2000);

z=0;

}

*\_delay\_ms*(2000);

z++;

}

}

# References

* <http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf>
* <http://www.micropik.com/PDF/dht11.pdf>
* <https://cdn-shop.adafruit.com/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf>
* <https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf>
* [www.spartkfun.com](http://www.spartkfun.com)
* [www.hobisyste.com](http://www.hobisyste.com)
* [http://hobbyist.co.nz](http://hobbyist.co.nz/sites/default/files/datalog/graph/graph.html)

Noor Blum Page 17/17