HACETTEPE UNIVERSITY

ELE417 – EMBEDDED SYSTEM DESIGN

PROJECT NAME :

IOT PROJECT SERVICE API

GROUP MEMBERS

ALİ SERDAR YURTCAN 21428628

SÜLEYMAN AKİF ÜNLÜ 21428537

ŞAFAK GALİP GÜNEŞ 21428058

CONTENTS

INTRODUCTION

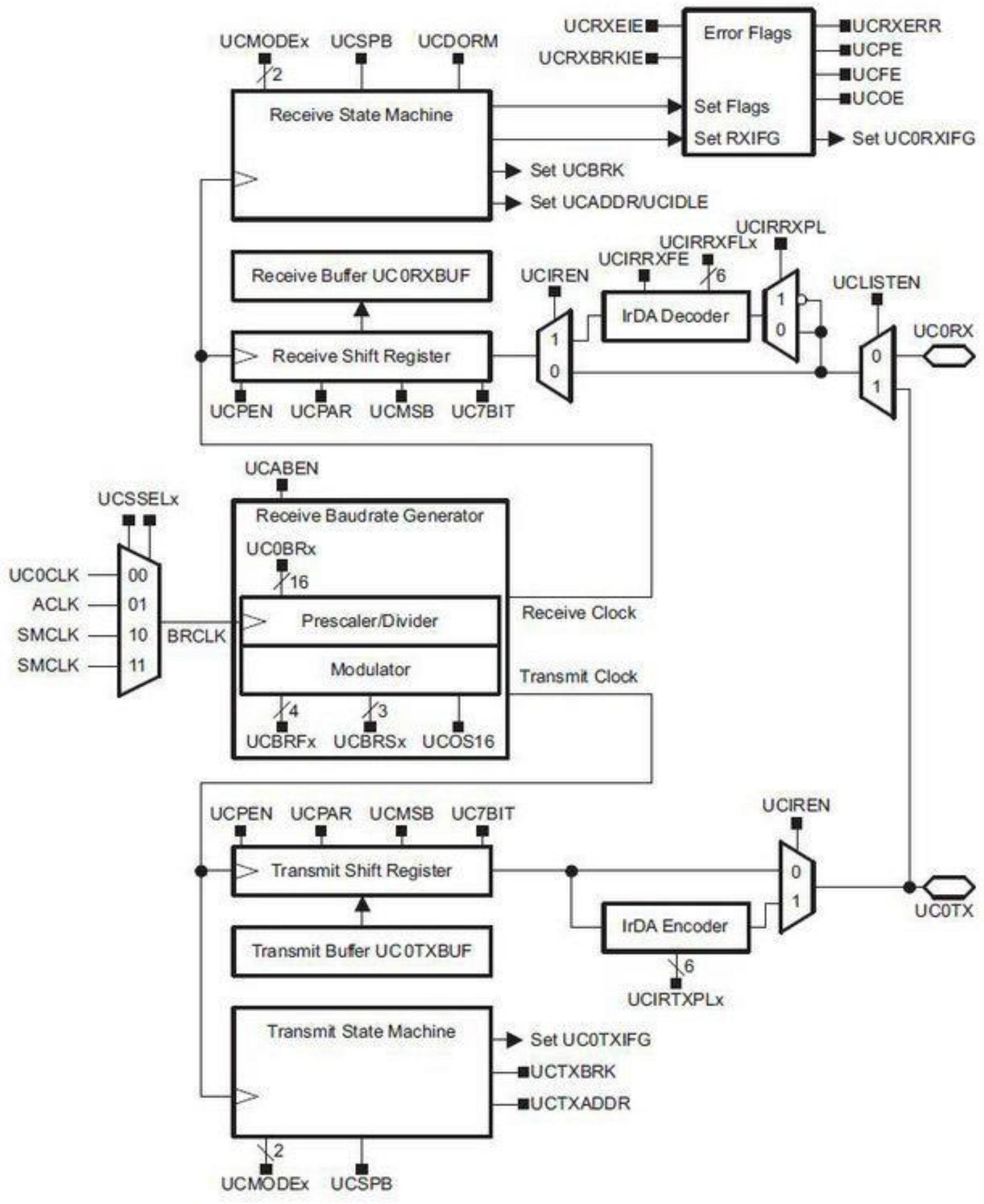
The purpose of this Project is taking the kind of data we want from the server and display it on the LCD screen. It is necessary to use communication protocols in accordance with their usage areas. It was used for data transfer at short distance more than one at a time with units such as sensor, LCD, memory element. In other words, the transmitter and the receiver communicate simultaneously. The UCSYNC bit on the UCAxCTL0 recorder is set to zero to access UART mode.

In UART mode, the MSP430 controller communicates with peripherals via the UCAxRXD and UCAxTXD pins. The x in the expressions represents the number of the USCI_Ax unit. There may be more than one USCI_A unit on different controllers. Since the MSP430G2553 controller used is an USCI_A unit, zero must be used instead of x.

Features of USCI_A0 unit:

a. 7 or 8 bit data transfer single, double or no parity support
b. Independent transmit and receive shift loggers
c. Separate receive and transmit buffer recorders
d. LSB or MSB first transfer selection
e. Multiprocessor communication support
f. Automatic wake up from saving modes when data arrives
g. Receiving and transmitting feature that can create independent interrupting

USB-UART converter unit on MSP430G2553 launchpad board is used. In this way, both debug / program operations were performed without the need for any additional hardware and connection.

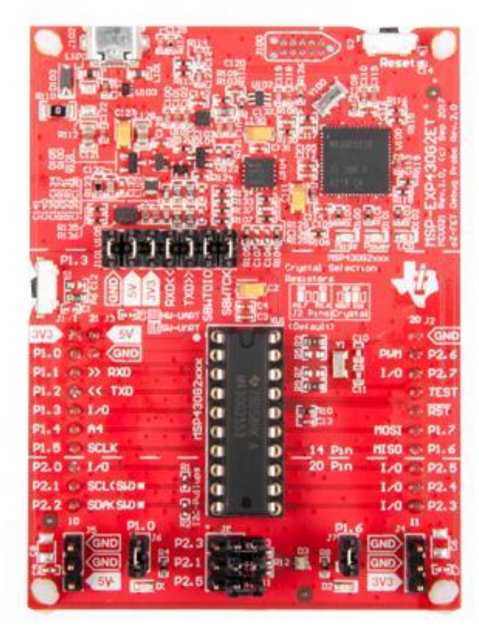


USCI Unit UART Mode Block Diagram

In addition the communication between the computer and the controller is provided UART. The CCS program was used to process the data from ESP8266 in MSP430G2553.

HARDWARE DOCUMENTATION

1) MSP430G2553



MSP430G2553 LAUNCHPAD

In the project, the MSP430G2553 launchpad, which provides 20 pin support, has been set using the CCS program.

Technicial Specifications :

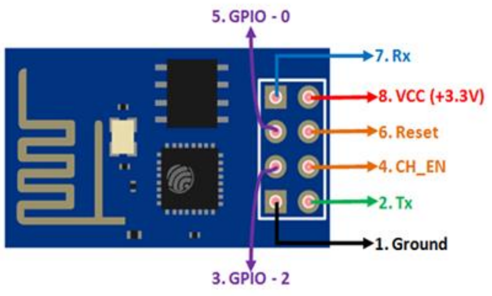
a. Low Supply-Voltage Range: 1.8 V to 3.6 V
b. Ultra-Low Power Consumption
    i. Active Mode: 230 µA at 1 MHz, 2.2 V
    ii. Standby Mode: 0.5 µA
    iii. Off Mode (RAM Retention): 0.1 µA
c. Five Power-Saving Modes
d. Ultra-Fast Wake-Up From Standby Mode in Less Than 1 µs
e. 16-Bit RISC Architecture, 62.5-ns Instruction Cycle Time
f. Basic Clock Module Configurations

    i. Internal Frequencies up to 16 MHz With Four Calibrated Frequency
    ii. Internal Very-Low-Power Low-Frequency (LF) Oscillator
    iii. 32-kHz Crystal
    iv. External Digital Clock Source

g. Two 16-Bit Timer_A With Three Capture/Compare Registers
h. Up to 24 Capacitive-Touch Enabled I/O Pins
i. Universal Serial Communication Interface (USCI)
    i. Enhanced UART Supporting Auto Baudrate Detection (LIN)
    ii. IrDA Encoder and Decoder
    iii. Synchronous SPI
    iv. I^2C™
j. On-Chip Comparator for Analog Signal Compare Function or Slope Analog-to-Digital (A/D) Conversion
k. 10-Bit 200-ksps Analog-to-Digital (A/D) Converter With Internal Reference, Sample-and-Hold, and Autoscan
l. Brownout Detector
m. Serial Onboard Programming, No External Programming Voltage Needed, Programmable Code Protection by Security Fuse
n. On-Chip Emulation Logic With Spy-Bi-Wire Interface
o. Package Options
    i. TSSOP: 20 Pin, 28 Pin
    ii. PDIP: 20 Pin
    iii. QFN: 32 Pin

2) ESP8266-01

ESP8266-01



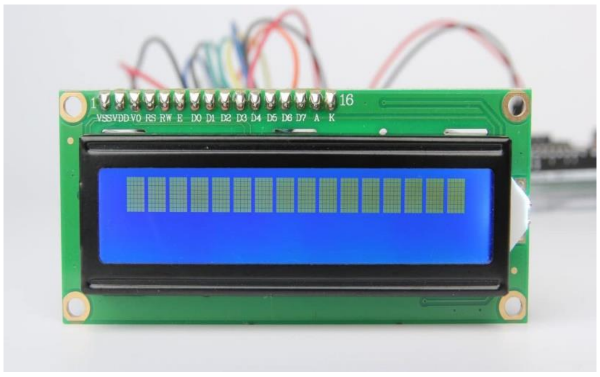
ESP8266-01 PINS

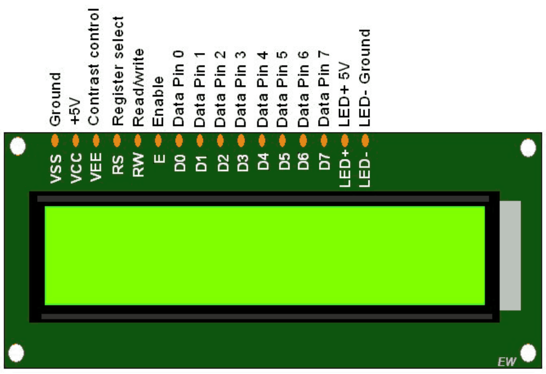
The wifi connection required to establish connection with the server in the project was provided with the ESP8266-01 chip.

Technicial Specifications :

a. Support 802.11 b / g / n
b. Internal TCP / IP protocol stack
c. + 19.5dBm output power (in 802.11b mode)
d. Leakage current <10uA
e. 32-bit processor with built-in low power consumption
f. SDIO 1.1 / 2.0, SPI and UART support
g. STBC, 1x1 MIMO, 2x1 MIMO
h. Wake up time and data packet time <2ms
i. Power consumption in stand-by <1mW

3) 2x16 HD44780 LCD


2x16 HD44780 LCD



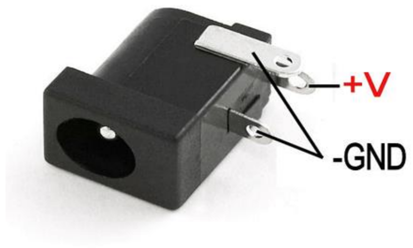
2x16 HD44780 LCD PINS

After the data was received from the server, it was converted into a meaningful expression on the microcontroller, it was printed on the 2x16 HD44780 LCD.

Technicial Specifications :

a. Works with + 5V.
b. It has Back Lighting feature.
c. It draws 4mA current without LCD backlight.
d. Its dimensions are 80x36x9.4mm
e. Working temperature is between -20 and +70 degrees.

4) VOLTAGE-JACK DCJ0303



DCJ0303

The voltage supply for the board prepared in the project was made with the DCJ0303 socket.

Technicial Specifications :

a. Pin diameter : 2.1mm

b. Hole Diameter : 5.5mm

c. Material : Plastic

d. Color : Black

e. Operating Voltage : 0-24VDC

f. Working Current : 5A

g. Number of Pins : 3

5) CAPACITORS

0.33uF Capacitor

1uF Capacitor

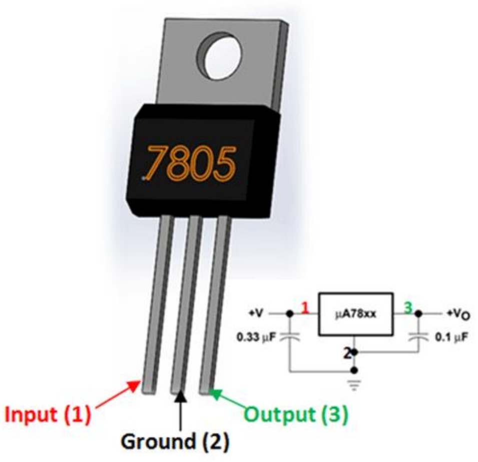
0.1uF Capacitor

10n F Capacitor

Capacitors used to store loads in some parts of the circuit.

6) L7805 VOLTAGE REGULATOR

L7805



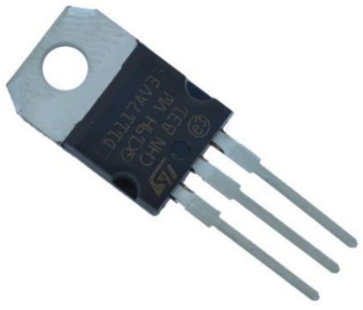
L7805 PINS

L7805 regulator was used to convert 12V source from voltage source to 5V.

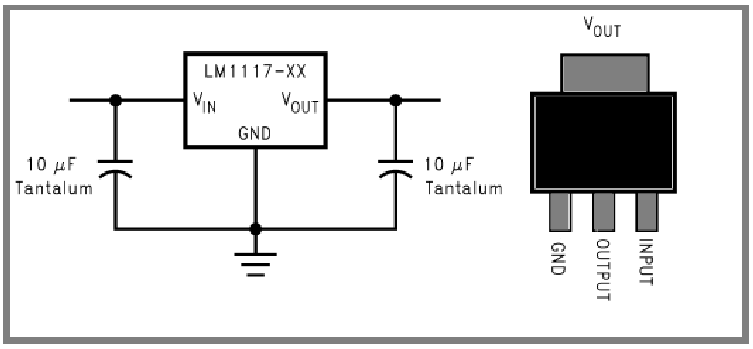
Technicial Specifications :

a. Number 1: Input, 7V-35V
b. Number 2: GND (Ground)
c. Number 3: Output, 5V

7) L1117 VOLTAGE REGULATOR



L1117



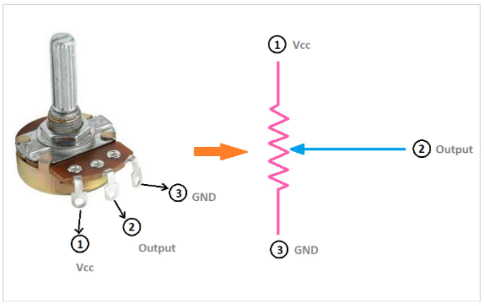
L1117 Pins

The L1117 regulator was used to reduce the 5V supply from the L7805 regulator to 3.3V.

Technicial Specifications :

a. Fixed 2.5V, 2.85V, 3.3V, 5V or adjustable output voltage
b. Low dropout voltage: 1.2V typical at up to 1A
c. Low ground current
d. Fast transient response
e. Current & thermal limiting
f. Line regulation: 0.5% typical
g. Load regulation: 0.5% typical
h. SOT-223, TO-220, TO-252, TO-263 package

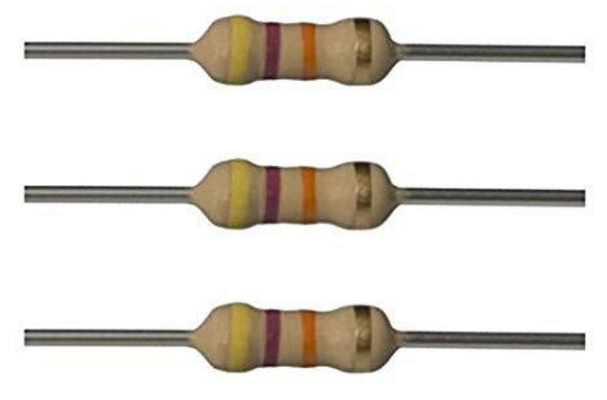
8) 10K POTENTIOMETER



10k POTENTİOMETER

In the project, 10k potentiometer was used to adjust the contrast of the LCD screen.

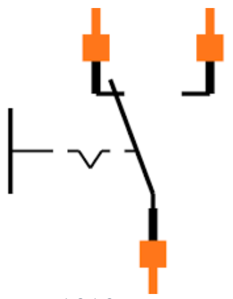
9) RESISTOR


47k RESISTOR

47k ohm resistor is connected between 3.3V to draw current to the reset pin.

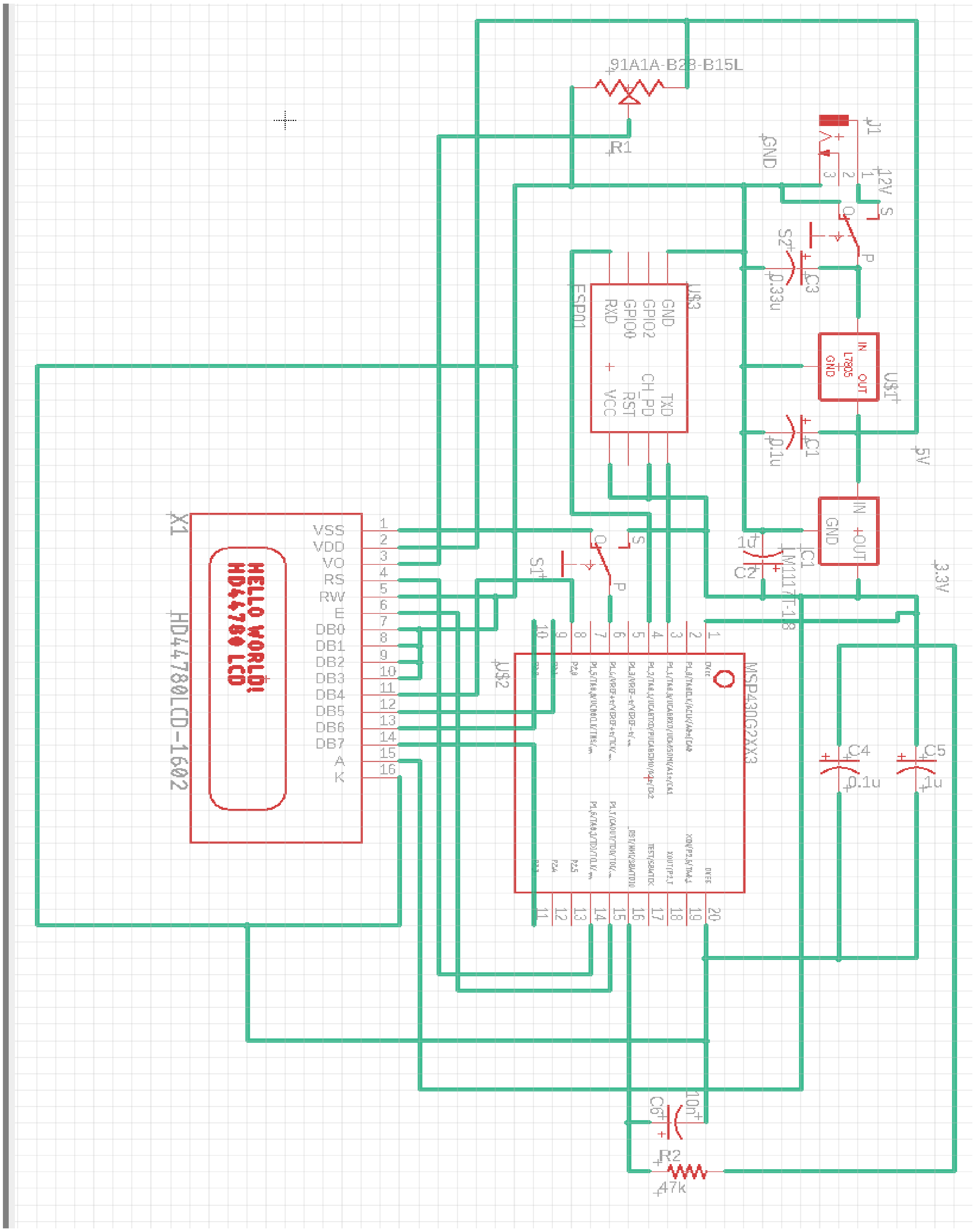
10) EG1218 SWITCHES

EG1218


EG1218 PINS

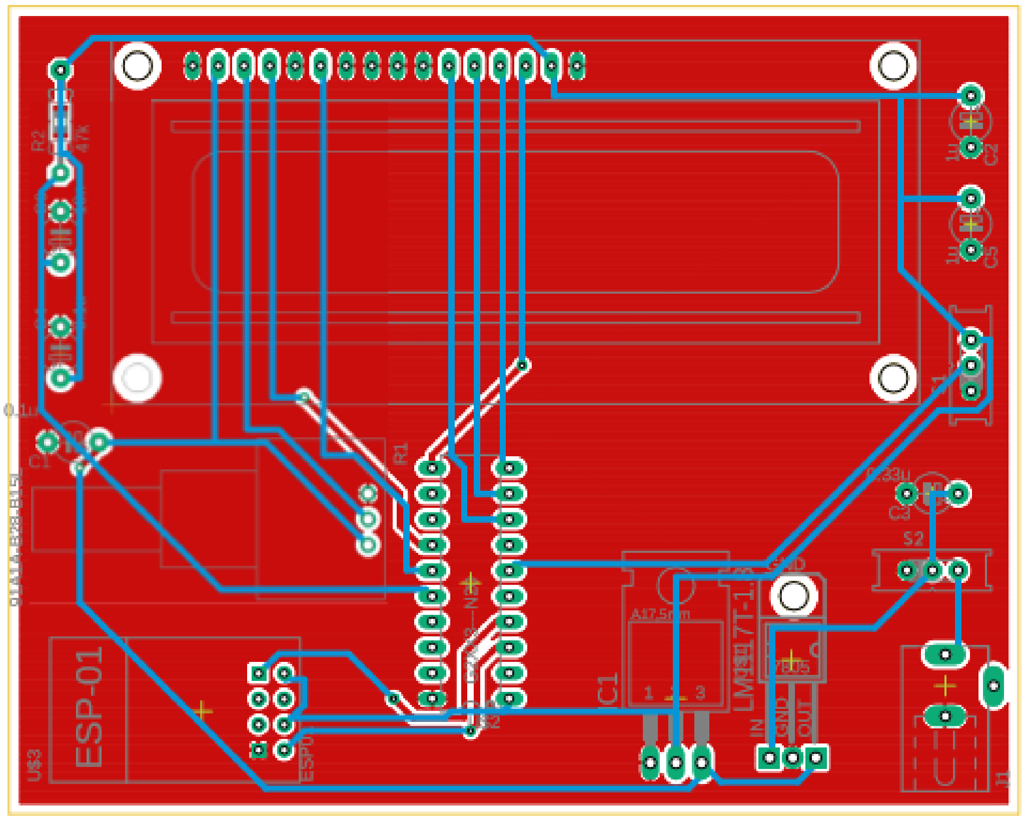
In the project, switches were used in two places. The first one was placed on the voltage source of the circuit and an on-off button was created. The second one was connected to a pin and the type of data received by the interrupt sent from the pin was changed.

11) PROJECT SCHEMATIC

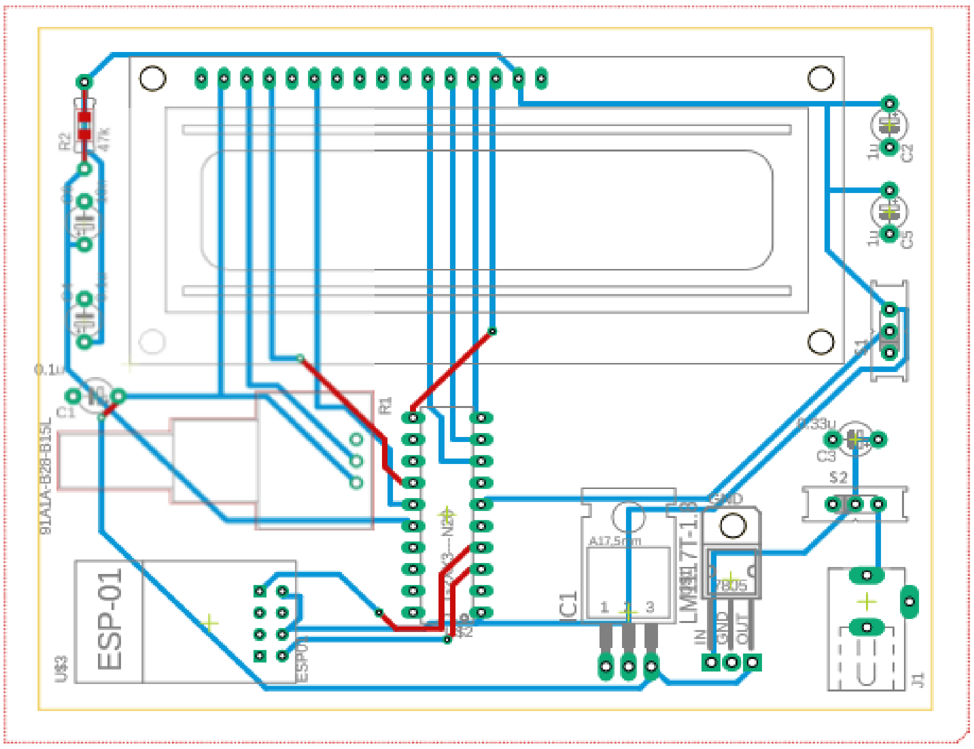


Project Schema

12) PROJECT BOARD



Project Board 1



Project Board 2

13) HARDWARE PROBLEMS

ESP8266-01 Wi-Fi module is produced by a Chinese company.

Since modules of well-known companies such as TI, Microchip and ST are expensive, ESP8266-01 module of Espressif Systems was used in the project. It was noticed that the purchased ESP module is out of date and the module was updated via USB port with ESP8266-Flasher, a Chinese application for updating.

To start updating, the GPIO_0 pin is connected to GND, the device is de-energized and re-supplied; so the module went into update mode. The update files were uploaded to the module via the application.

SOFTWARE DOCUMENTATION

1) CODE DOCUMENTATION

```
#include <msp430g2553.h>
#include <msp430.h>
#include "lcd_4bit.h"
#include "uart.h"
char myarray[200];
unsigned int l0;
#define IP_S "\"10.50.106.10\""
unsigned volatile int t0,t1,t2;

#define WPA_KEY "\"EmbeddeD\""
#define AP_SSID "\"ELE417\""
#define C_UDP "\"TCP\""
int unsigned m=0;//variable for selecting mode
void delay(void); //delay function
void my_putc(char c); // put character on lcd
void lcd_clear(void); //clear lcd
void lcd_puts(const char *); //puts string data on lcd
void cleararray(char marr[]); // erasing all array items
void uart_send_data(unsigned char * data, unsigned char length);
//Put string on TX
```

```
void crc(char data[]);  //CRC Checking bit

void crc0(char data[]); //CRC Checking bit

void crc2(char data[]);//CRC Checking bit

void uart_send_char(unsigned char val);//Put char on TX

unsigned int CRC16=0xFFFF; //using in crc

unsigned int tempcrc=0x0000;    //using in crc

int main(void)

{



P1SEL = 0x00;

P1OUT = 0x00;

P1DIR |=BIT6+BIT7;

P2SEL = 0x00;

P2OUT = 0x00;

P2DIR = 0xFF;


WDTCTL = WDTPW + WDTHOLD;                  // Stop watchdog timer

DCOCTL = 0;                                // Select lowest DCOx and
MODx settings

BCSCTL1 = CALBC1_1MHZ;                     // Set DCO

DCOCTL = CALDCO_1MHZ;

P1SEL = BIT1 + BIT2 ;                      // P1.1 = RXD, P1.2=TXD

P1SEL2 = BIT1 + BIT2;

UCA0CTL1 |= UCSSEL_2;                      // SMCLK

UCA0BR0 = 8;                               // 1MHz 115200

UCA0BR1 = 0;                               // 1MHz 115200

UCA0MCTL = UCBRS2 + UCBRS0;                // Modulation UCBRSx = 5

UCA0CTL1 &= ~UCSWRST;    // **Initialize USCI state machine**


P1DIR &=~0x10;               // All P1 pins as out but P1.4

P1REN |= 0x10; // P1.4 Resistor enabled

P1OUT |= 0x10; // P1.4 Resistor as pull-up
```

```
P1IES &= 0x10; // P1.4 Hi->Lo edge selected


P1IE |= 0x10;

P1IFG &=~0x10;


IE2 &= ~UCA0RXIE;

IE2 &= ~UCA0TXIE;

__bis_SR_register(GIE);


lcd_init();

lcd_clear();

delay();

lcd_goto(1,1);

lcd_puts("Connecting");

lcd_goto(2,1);

delay();

lcd_puts("Server...");

delay();





uart_send_data((unsigned char *)"AT", 2);

delay();

uart_send_data((unsigned char *)"\r\n  ", 2);

delay();

uart_send_data((unsigned char *)"AT+CWMODE=3", 11);

uart_send_data((unsigned char *)"\r\n  ", 4);

delay();

uart_send_data((unsigned char *)"AT+CWJAP=", 9);

delay();

uart_send_data((unsigned char *)AP_SSID, 8);

delay();
```

```
uart_send_data((unsigned char *)",", 1);
delay();
uart_send_data((unsigned char *)WPA_KEY, 10);
delay();
uart_send_data((unsigned char *)"\r\n  ", 4);
delay();
delay();
delay();
delay();
delay();
delay();
delay();
uart_send_data((unsigned char *)"AT+CIPSTART=", 12);
delay();
uart_send_data((unsigned char *)C_UDP, 5);
uart_send_data((unsigned char *)",", 1);
delay();
IE2 |= UCA0RXIE;
uart_send_data((unsigned char *)IP_S, 14);
uart_send_data((unsigned char *)",", 1);
uart_send_data((unsigned char *)"10000", 5);
uart_send_data((unsigned char *)"\r\n  ",4);
delay();
delay();
delay();
delay();
delay();
delay();
delay();
if(myarray[37]=='O');{
if(myarray[38]=='K'){
lcd_clear();
```

```
delay();
lcd_goto(1,1);
lcd_puts("CONNECTED");
delay();
delay();
delay();
lcd_clear();
delay();
lcd_goto(1,1);
delay();
lcd_puts("    HU EE 417");
lcd_goto(2,1);
lcd_puts("    PROJECT");
}
else{
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("ERROR TRY AGAIN");
}
}
delay();
delay();
delay();
while(1){







if(m==0){
```

```
delay();
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("SENDING HI");
delay();
delay();
cleararray(myarray);

uart_send_data((unsigned char *)"AT+CIPSEND=2\r\nhi  ", 18);
delay();
lcd_init();
lcd_clear();
delay();
lcd_goto(1,1);
t0=41;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;

while(t0<=t2){
lcd_putch(myarray[t0]);
t0++;
}
cleararray(myarray);
}
```

```
if(m==1){
tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("TEMP REQUEST");
delay();
delay();
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("TEMPERATURE=");
delay();
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x01  ", 18);
lcd_goto(2,1);
delay();
crc(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=41;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){


if(myarray[t0+t1-2]==tempcrc){
```

```
while(t0<=t2){


lcd_putch(myarray[t0]);

t0++;

}

lcd_goto(2,6);

lcd_puts("Degree");

}}

else{


lcd_clear();

delay();

lcd_goto(1,1);

lcd_puts("CRC ERROR");

}

cleararray(myarray);

}

if(m==2){


tempcrc= 0x0000;

CRC16 = 0xFFFF;

lcd_clear();

delay();

lcd_goto(1,1);
```

```
lcd_puts("AIR PRESSURE=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x02  ", 18);
delay();
delay();
crc(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=41;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
}}
else{

lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}
if(m==3){









tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("HUMIDITY=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x03  ", 18);
delay();
crc(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=41;
```

```
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
}}
else{
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}
if(m==4){









tempcrc= 0x0000;
CRC16 = 0xFFFF;
```

```
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("WIND SPEED=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x04  ", 18);
delay();
delay();
crc(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=41;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
}}
else{
```

```
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}
if(m==5){








tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CLOUDINESS=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x05  ", 18);
delay();
delay();
crc(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
```

```
t0=41;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
}}
else{

lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}









if(m==6){
tempcrc= 0x0000;
```

```
CRC16 = 0xFFFF;

lcd_clear();

delay();

lcd_goto(1,1);

lcd_puts("WEATHER STATUS");

delay();

delay();

delay();

lcd_goto(2,1);

uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x06  ", 18);

delay();

delay();

crc(myarray);

tempcrc = CRC16 ;

CRC16 &= 0b0000000011111111;

tempcrc &= 0b1111111100000000;

tempcrc >>= 8 ;

t0=41;

t1=myarray[t0+1]; //length of message

t2=t0+t1-3;

t0=t0+2;

if(myarray[t0+t1-1]==CRC16){


if(myarray[t0+t1-2]==tempcrc){


while(t0<=t2){


lcd_putch(myarray[t0]);

t0++;

}

}}

else{
```

```
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}








if(m==7){
tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CURRENT TIME=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x07  ", 18);
delay();
delay();
crc2(myarray);
tempcrc = CRC16 ;
```

```
CRC16 &= 0b0000000011111111;

tempcrc &= 0b1111111100000000;

tempcrc >>= 8 ;

t0=42;

t1=myarray[t0+1]; //length of message

t2=t0+t1-3;

t0=t0+2;

if(myarray[t0+t1-1]==CRC16){


if(myarray[t0+t1-2]==tempcrc){


while(t0<=t2){


lcd_putch(myarray[t0]);

t0++;

}

}}

else{


lcd_clear();

delay();

lcd_goto(1,1);

lcd_puts("CRC ERROR");

}

cleararray(myarray);

}
```

```
if(m==8){
tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CURRENT DATE=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x08  ", 18);
delay();
delay();
crc2(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=42;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){
```

```
lcd_putch(myarray[t0]);
t0++;
}
}}
else{

lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}














if(m==9){
tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("SERVER UPTIME=");
delay();
delay();
delay();
```

```
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x09  ", 18);
delay();
delay();
crc2(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=42;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
}}
else{

lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}
```

```
if(m==10){
tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("SYSTEM TEMP=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x0A  ", 18);
delay();
delay();
crc(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=41;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){
```

```
if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
lcd_goto(2,6);
lcd_puts("Degree");
}}
else{
lcd_clear();
delay();
lcd_goto(1,1);
lcd_puts("CRC ERROR");
}
cleararray(myarray);
}









if(m==11){
tempcrc= 0x0000;
CRC16 = 0xFFFF;
lcd_clear();
```

```
delay();
lcd_goto(1,1);
lcd_puts("CONNECTED USERS=");
delay();
delay();
delay();
lcd_goto(2,1);
uart_send_data((unsigned char *)"AT+CIPSEND=2\r\n\xEE\x0B  ", 18);
delay();
crc0(myarray);
tempcrc = CRC16 ;
CRC16 &= 0b0000000011111111;
tempcrc &= 0b1111111100000000;
tempcrc >>= 8 ;
t0=40;
t1=myarray[t0+1]; //length of message
t2=t0+t1-3;
t0=t0+2;
if(myarray[t0+t1-1]==CRC16){

if(myarray[t0+t1-2]==tempcrc){

while(t0<=t2){

lcd_putch(myarray[t0]);
t0++;
}
}}
else{

lcd_clear();
delay();
lcd_goto(1,1);

lcd_puts("CRC ERROR");

}

cleararray(myarray);

}




__bis_SR_register(LPM4_bits+GIE);

}

}




#pragma vector=USCIAB0RX_VECTOR

__interrupt void USCI0RX_ISR(void)

{




if (UCA0RXBUF != '\0'){

myarray[l0]=UCA0RXBUF;

l0++;

}




}

#pragma vector=PORT1_VECTOR

__interrupt void Port_1(void)

{

delay();


if(m<12){

m++;
```

```
}
else{ m=0;}

__low_power_mode_off_on_exit();
P1IFG = ~BIT4;
}
void delay(void){
__delay_cycles(250000);
__delay_cycles(250000);
__delay_cycles(250000);
__delay_cycles(250000);
}
void cleararray(char marr[])
{
unsigned volatile int i = 200;
for(i;i<201;i--)
{
marr[i] = '\0';
}
}
void uart_send_data(unsigned char * data, unsigned char length){
unsigned int i ;
for(i = 0 ; i < length ; i ++){
uart_send_char(data[i]);
}
}
void uart_send_char(unsigned char val){

while (!(IFG2 & UCA0TXIFG));
UCA0TXBUF = val ;
__delay_cycles(600);
}
```

```
void crc(char data[])
{
unsigned int lengthdata=data[42]-0x04;
int n=2,i,j;
for(i=lengthdata;i>0;i--)
{
CRC16^=data[41+n];
for (j=0;j<8;j++)
{
CRC16 >>= 1;        // işaretli yer
if (__get_SR_register() & 0x0001 != 0)
{
CRC16 ^= 0xA001;
}

}
n++;
}
}
void crc0(char data[])
{

unsigned int lengthdata=data[41]-0x04;
int n=2,i,j;
for(i=lengthdata;i>0;i--)
{
CRC16^=data[40+n];
for (j=0;j<8;j++)
{
CRC16 >>= 1;        // işaretli yer
if (__get_SR_register() & 0x0001 != 0)
{
```

```
CRC16 ^= 0xA001;
}
}
n++;
}
}

void crc2(char data[])
{

unsigned int lengthdata=data[43]-0x04;
int n=2,i,j;
for(i=lengthdata;i>0;i--)
{
CRC16^=data[42+n];
for (j=0;j<8;j++)
{
CRC16 >>= 1;         // işaretli yer
if (__get_SR_register() & 0x0001 != 0)
{
CRC16 ^= 0xA001;
}
}
n++;
}
}
```

2) EXPLANATION OF CODE

While writing the code, the focus was on saving power. To achive this saving, CPU is used in active mode only in processing interrupts. All the other time MSP430G2553 waits to interrupt with in the Low Power Mode. Before the main function starts, the program transmit datas for connecting server by using ESP8266 module. If connection becomes succesful and MSP430G2553 gets "OK" message in RX buffer, MSP430G2553 writes "CONNECTED" on LCD. If connection is failed with the server, MSP430 writes "ERROR" on the LCD screen. Then if the connection is succesful with MSP430G2553, "hi" message is sent to start the communication between the server and MSP430G2553. If the answer "hi" is received again, MSP430G2553 prints "hi" on the lcd and if not, it writes "ERROR".

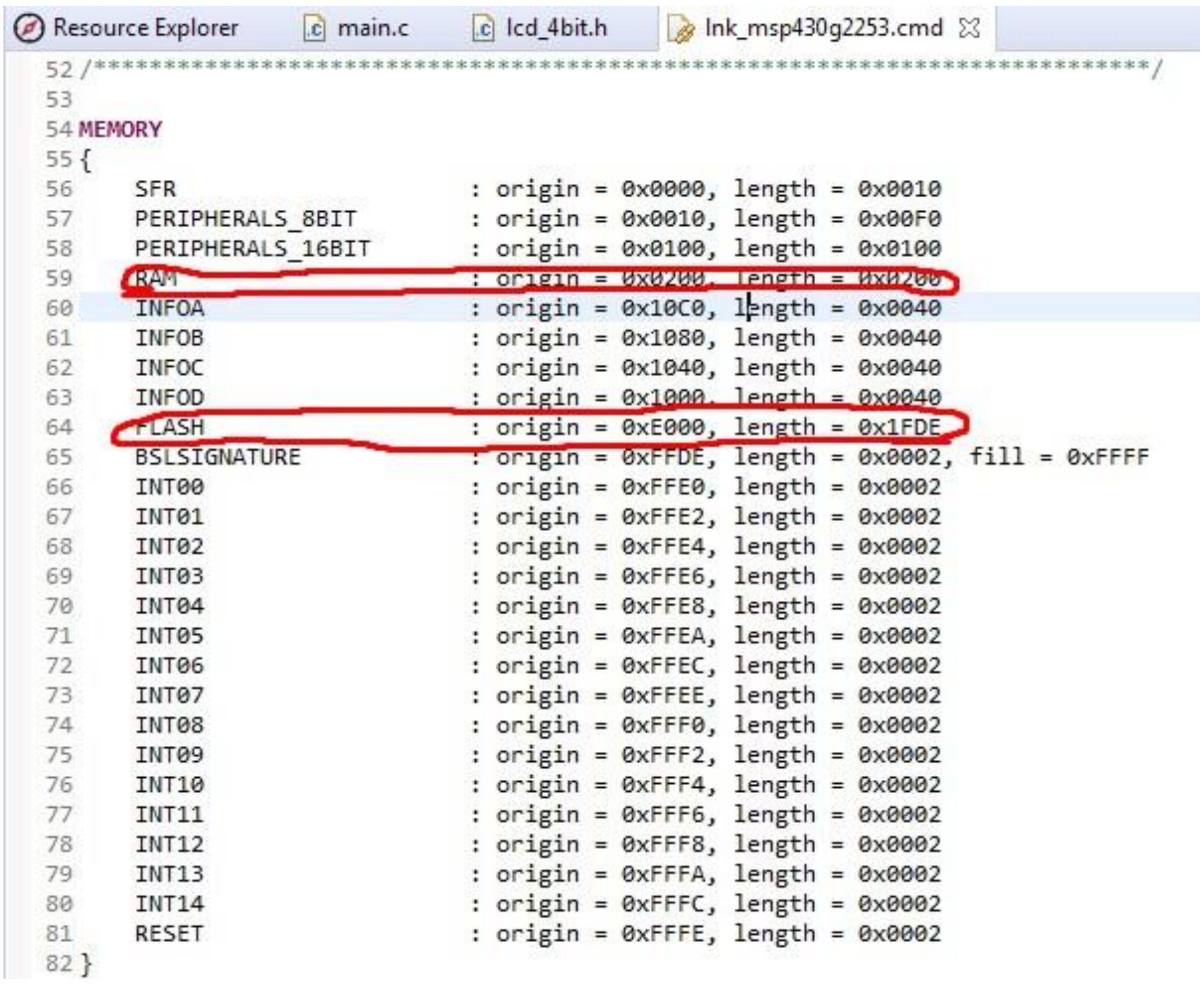
Basically main function starts with the command of "Enter Low Power Mode 4" and it starts to wait an interrupt. The variable "m", which is used as a transition between data request types, is connected to MSP430G2553 in Port 1.4 with the analog switch component. MSP430G2553 gets interrupt from P1.4 port interrupt when switch is turned. This port interrupt does two things. First increases "m" variable to process new request. Second and the most important exits are the low power mode on exit. So MSP430G2553 runs at full power when coming back to main function. With activation of SMCLK and increased "m" value, MSP430G2553 goes in the proper if command.

In if commands MSP430G2553 sends specific request messages on UART for getting specific datas. If messages are sent without error MSP430 gets response datas from RX buffer and saves in "myarray" array. After this point crc function checks data integrity according to CRC16 protocol. If recieved data meets the appropiate conditions, MSP430 writes this data on LCD screen. After the operation is succesful, program loop starts from beginnig and puts MSP430 in low power mode 4. For each value of "m", MSP430 follows the same path. When "m" equals to eleven, next turn on the switch makes "m" equal to zero and all process starts from the beginning.

If the printing of the LCD process is explained in detail, the 41st value of "myarray" is the beginning of the incoming message. So when increased 41 by one, will get total message length. It is also known that the last two bytes of the message are CRC16 bytes. So with this information, private data can be selected from all data of the message.

3) SOFTWARE PROBLEMS

One of the problems encountered is the memory problem.
Since the main focus of the program was power saving, the stack part had to be used as little as possible. As a result, all parts of the program are executed in the main function. Flash memory of MSP430G2553 is insufficient because array used is quite large. To solve this problem, the memory start and end addresses in the MSP430G2553 cmd file; stack was changed to use less and increase flash memory. Thus, the memory of the microprocessor has been rearrangeed according to the specific process to be processed.

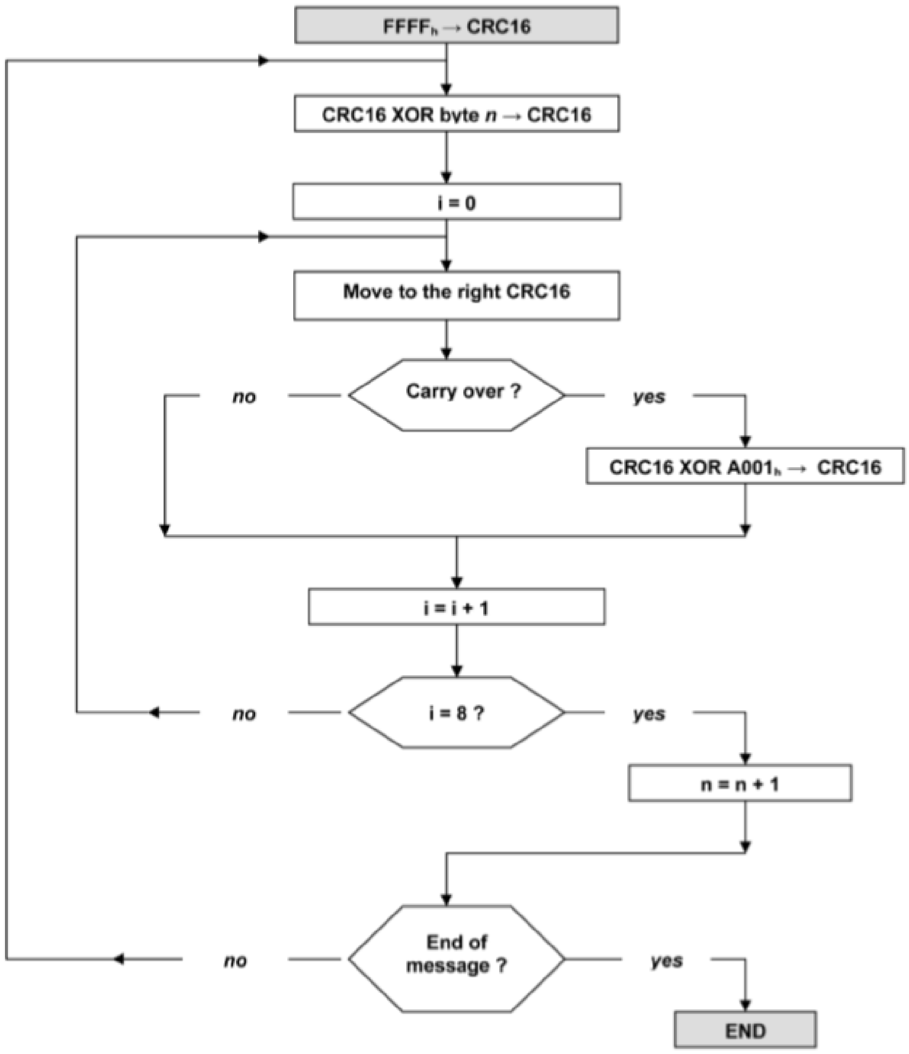


```
/******************************************************************************/

MEMORY
{
    SFR                     : origin = 0x0000, length = 0x0010
    PERIPHERALS_8BIT        : origin = 0x0010, length = 0x00F0
    PERIPHERALS_16BIT       : origin = 0x0100, length = 0x0100
    RAM                     : origin = 0x0200, length = 0x0200
    INFOA                   : origin = 0x10C0, length = 0x0040
    INFOB                   : origin = 0x1080, length = 0x0040
    INFOC                   : origin = 0x1040, length = 0x0040
    INFOD                   : origin = 0x1000, length = 0x0040
    FLASH                   : origin = 0xE000, length = 0x1FDE
    BSLSIGNATURE            : origin = 0xFFDE, length = 0x0002, fill = 0xFFFF
    INT00                   : origin = 0xFFE0, length = 0x0002
    INT01                   : origin = 0xFFE2, length = 0x0002
    INT02                   : origin = 0xFFE4, length = 0x0002
    INT03                   : origin = 0xFFE6, length = 0x0002
    INT04                   : origin = 0xFFE8, length = 0x0002
    INT05                   : origin = 0xFFEA, length = 0x0002
    INT06                   : origin = 0xFFEC, length = 0x0002
    INT07                   : origin = 0xFFEE, length = 0x0002
    INT08                   : origin = 0xFFF0, length = 0x0002
    INT09                   : origin = 0xFFF2, length = 0x0002
    INT10                   : origin = 0xFFF4, length = 0x0002
    INT11                   : origin = 0xFFF6, length = 0x0002
    INT12                   : origin = 0xFFF8, length = 0x0002
    INT13                   : origin = 0xFFFA, length = 0x0002
    INT14                   : origin = 0xFFFC, length = 0x0002
    RESET                   : origin = 0xFFFE, length = 0x0002
}
```

Memory Arrange of MSP430G2553

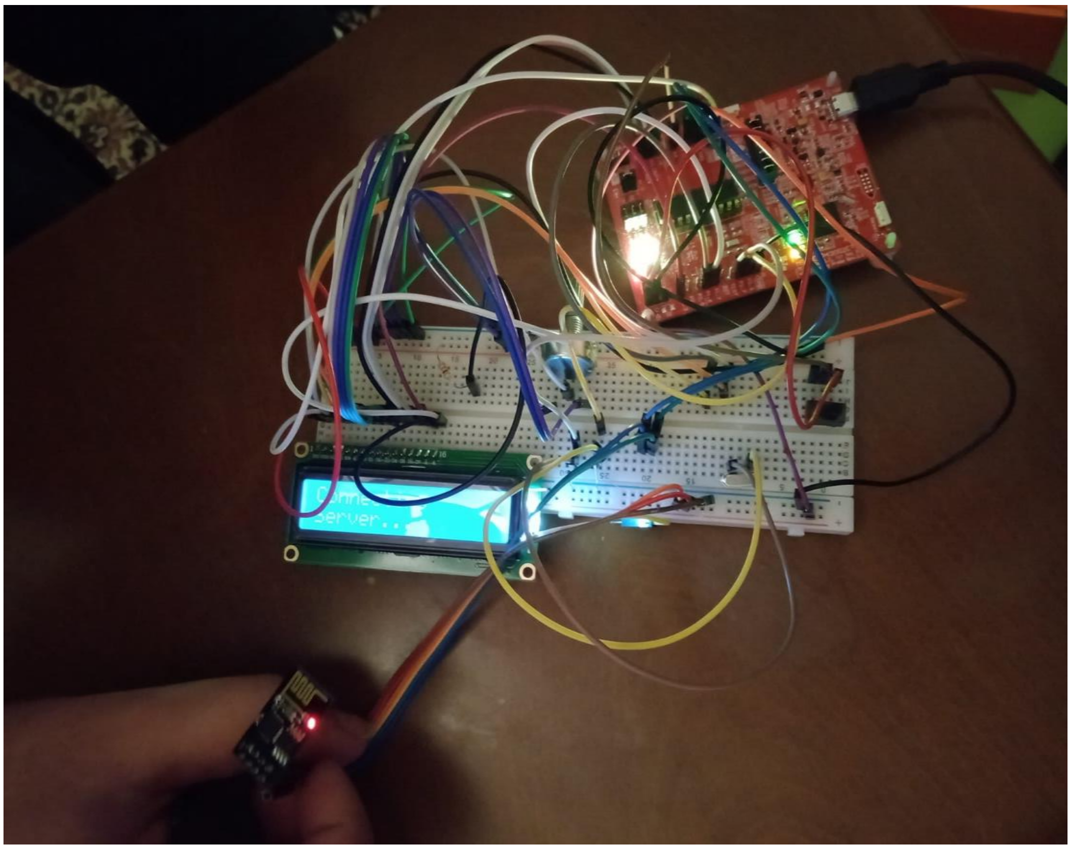
4) CRC16 FLOWCHART



CRC16 Flowchart Diagram

INITIAL SOLUTION of the PROJECT

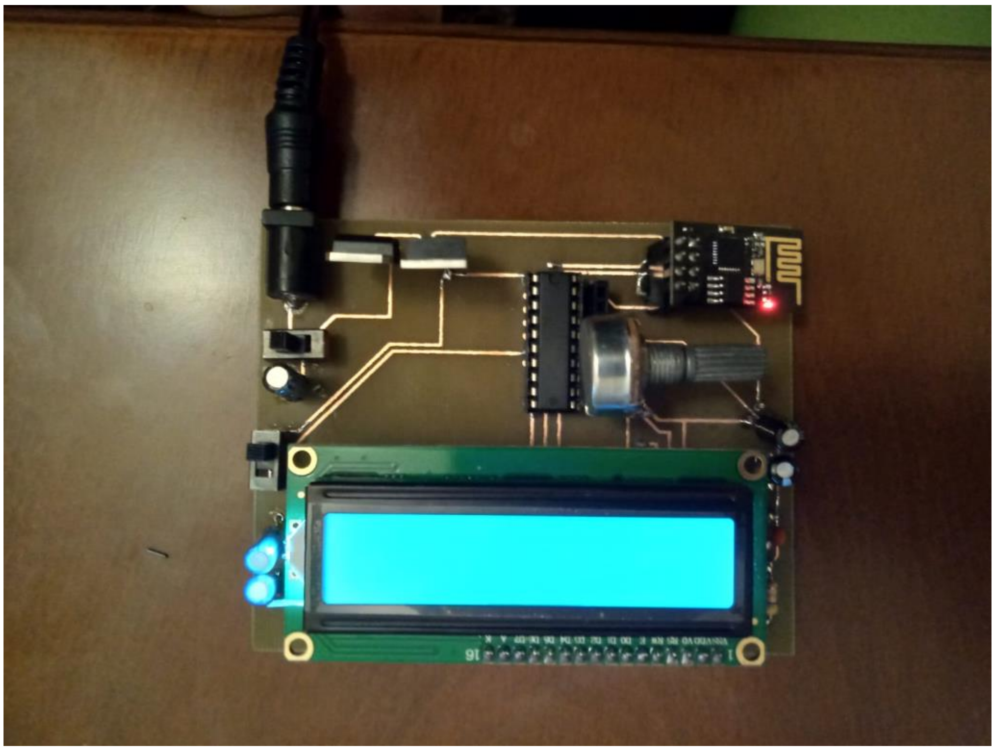
The circuit design was tested on the breadboard after the components to be used were determined and the connections to be made were determined. After understanding that the modules are working, the code writing process was developed in the circuit set up on the breadboard.
It contributed to the PCB board design, which will create a permanent solution.



Breadboard Circuit Design

FINAL SOLUTION of the PROJECT

As a result of the tests performed on the breadboard, the PCB board designed on the EAGLE application was pressed and placed on the components. However, errors were encountered during board printing. Board lines and pins have been reworked; errors fixed.


PCB Board Design