**AEMC PROJECT ACTIVITY REPORT**

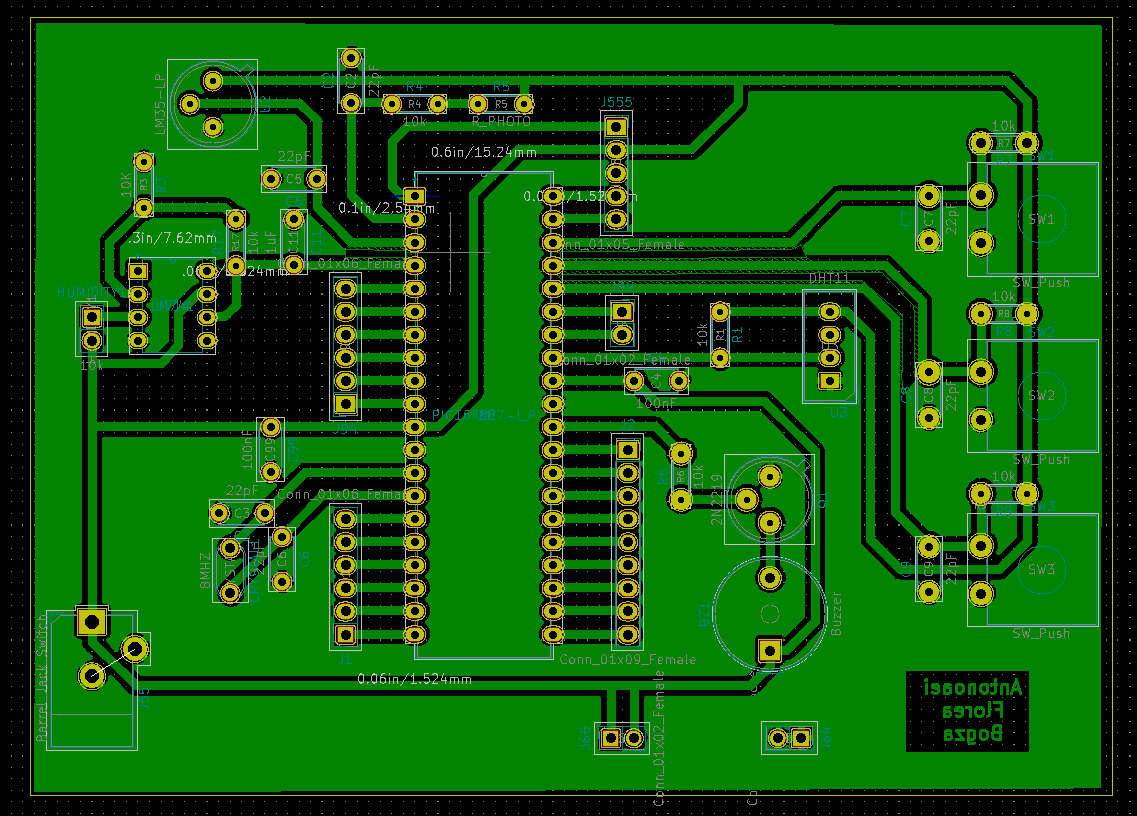
**TEAM1 - Group 5311**

**Bogza Gabriel** - Realization of the PCB

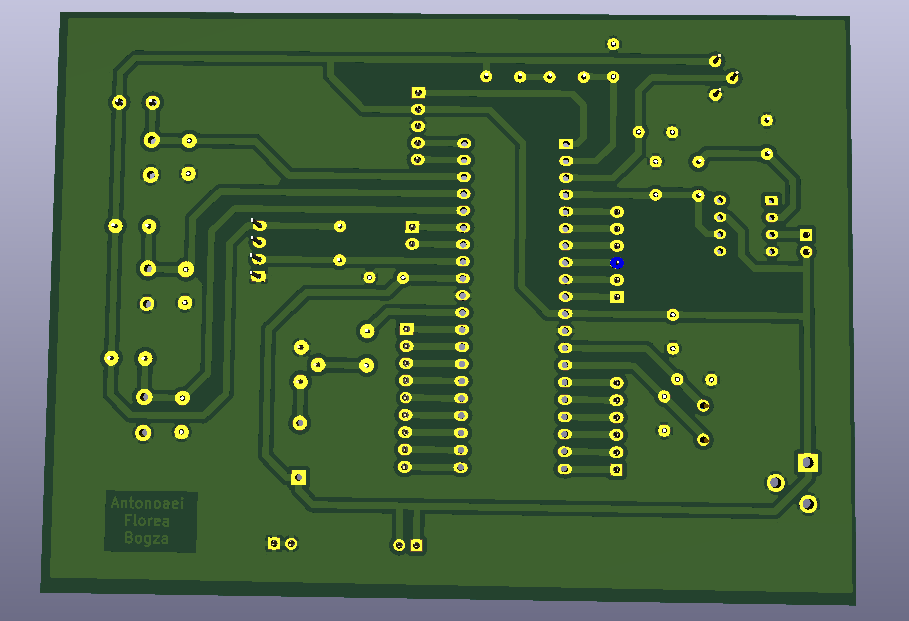
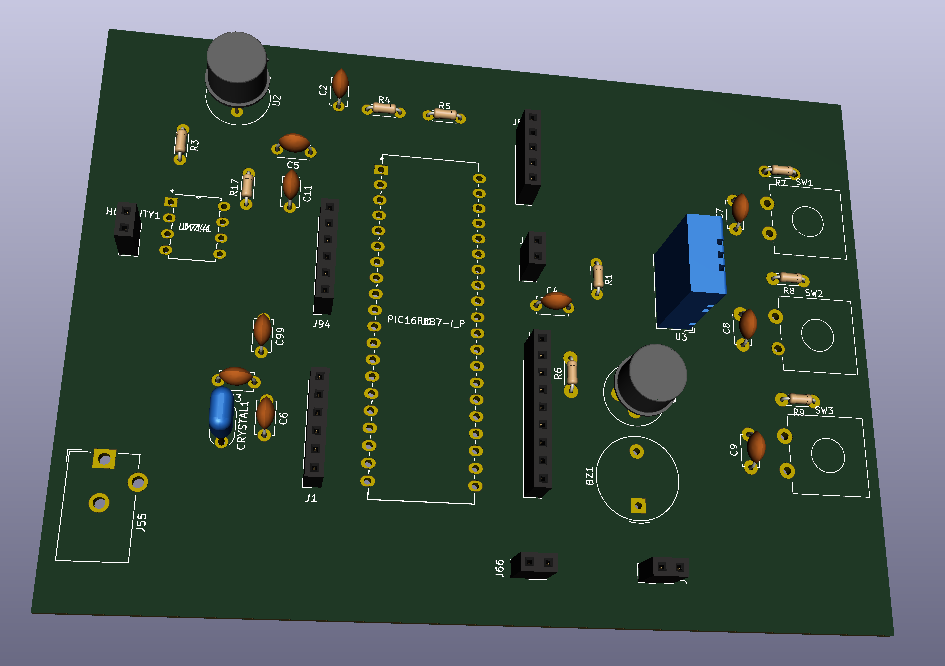
Used KiCad to create the layout for the PCB

Searching for footprints

Design the layout



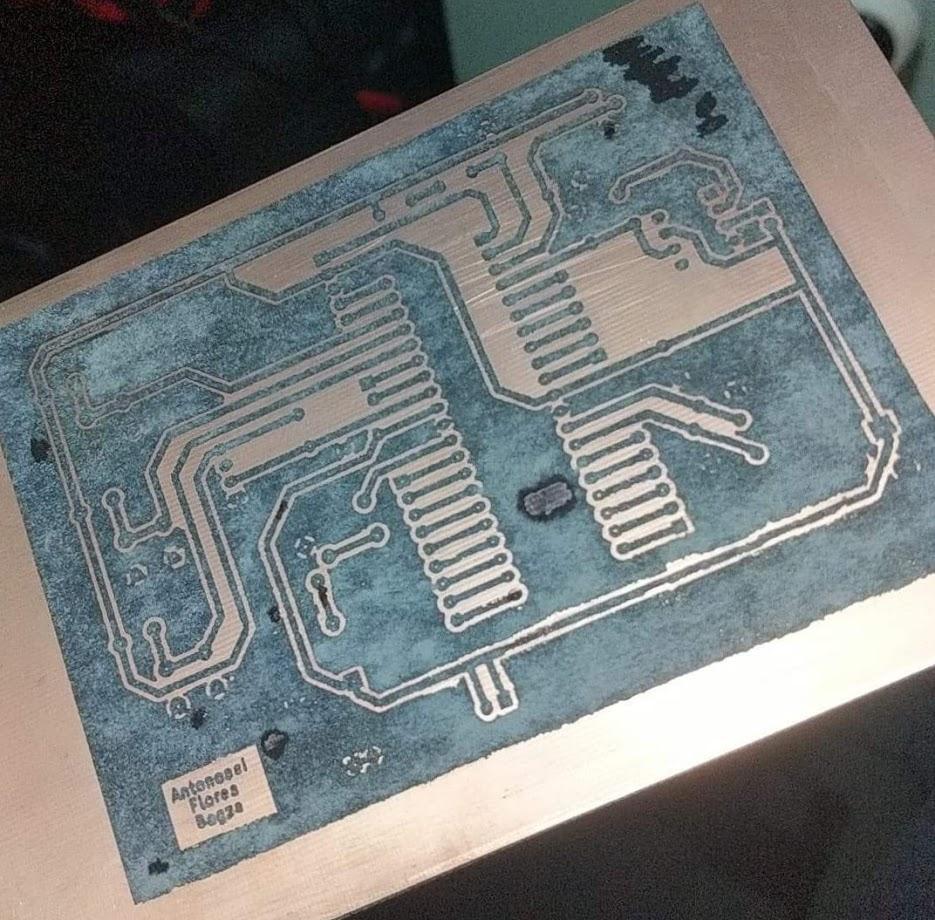
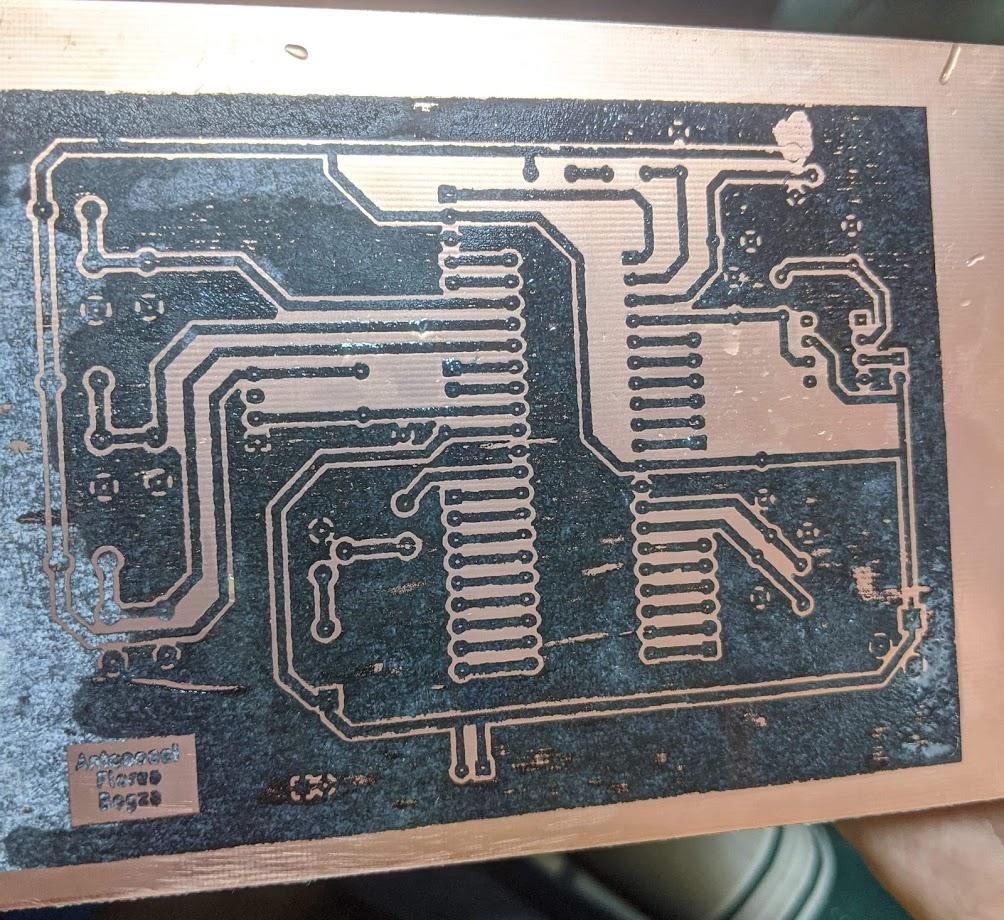
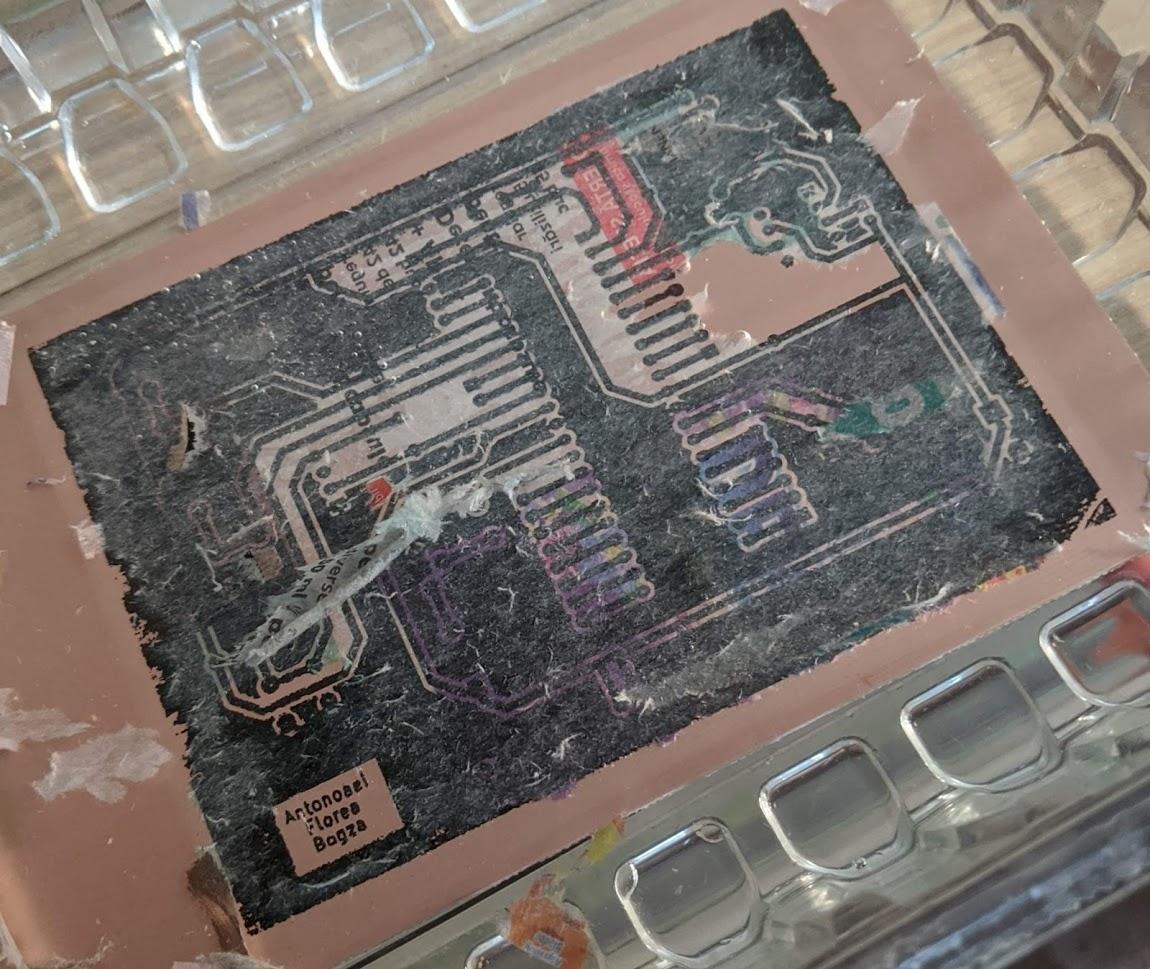
**3D simulation:**



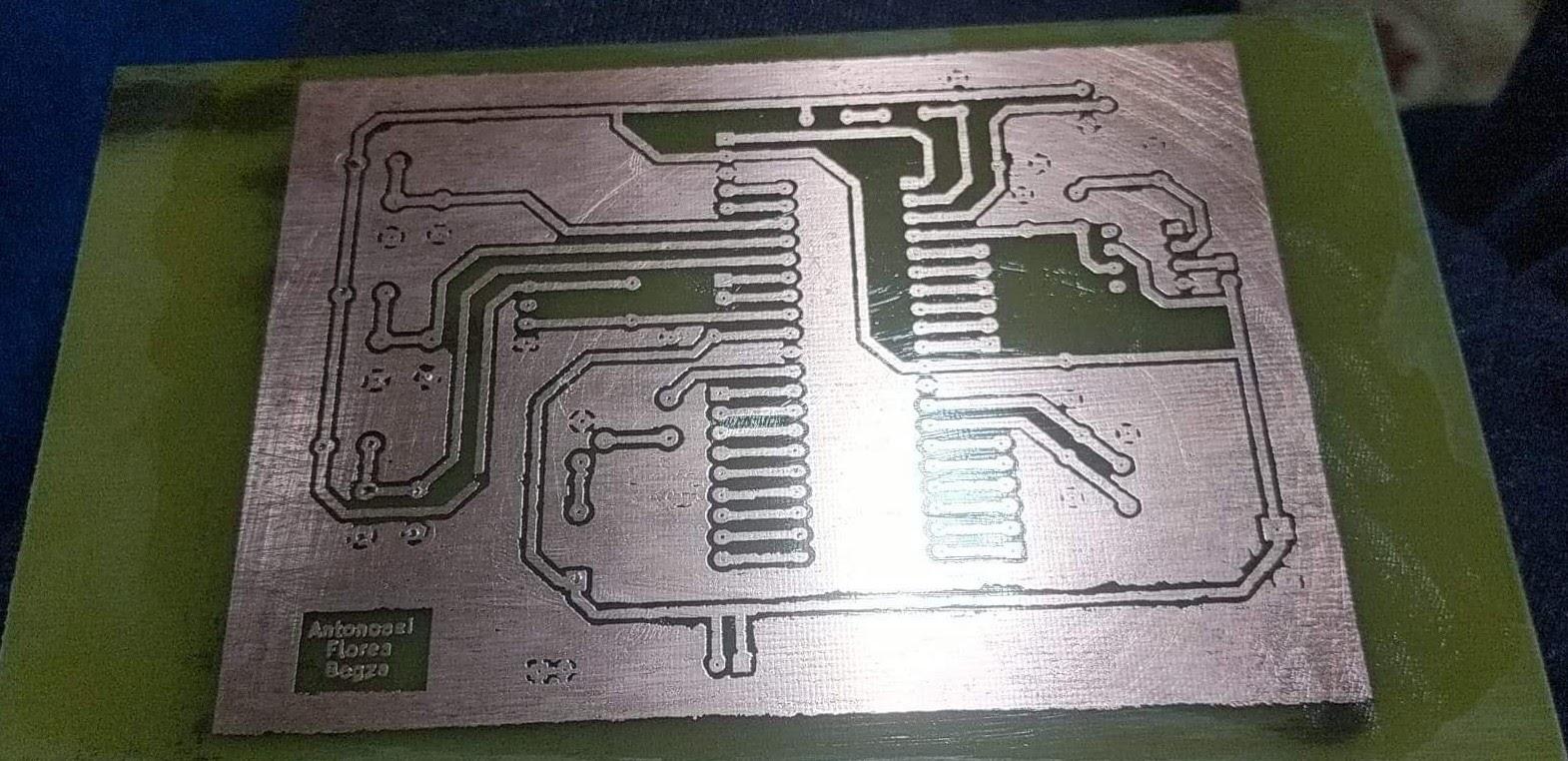
Order the components.

Multiple attempts to print the layout because the printer does not respect the sizes of pins.

Transfer the layout on the copper plate using the acetone and magazine paper method.

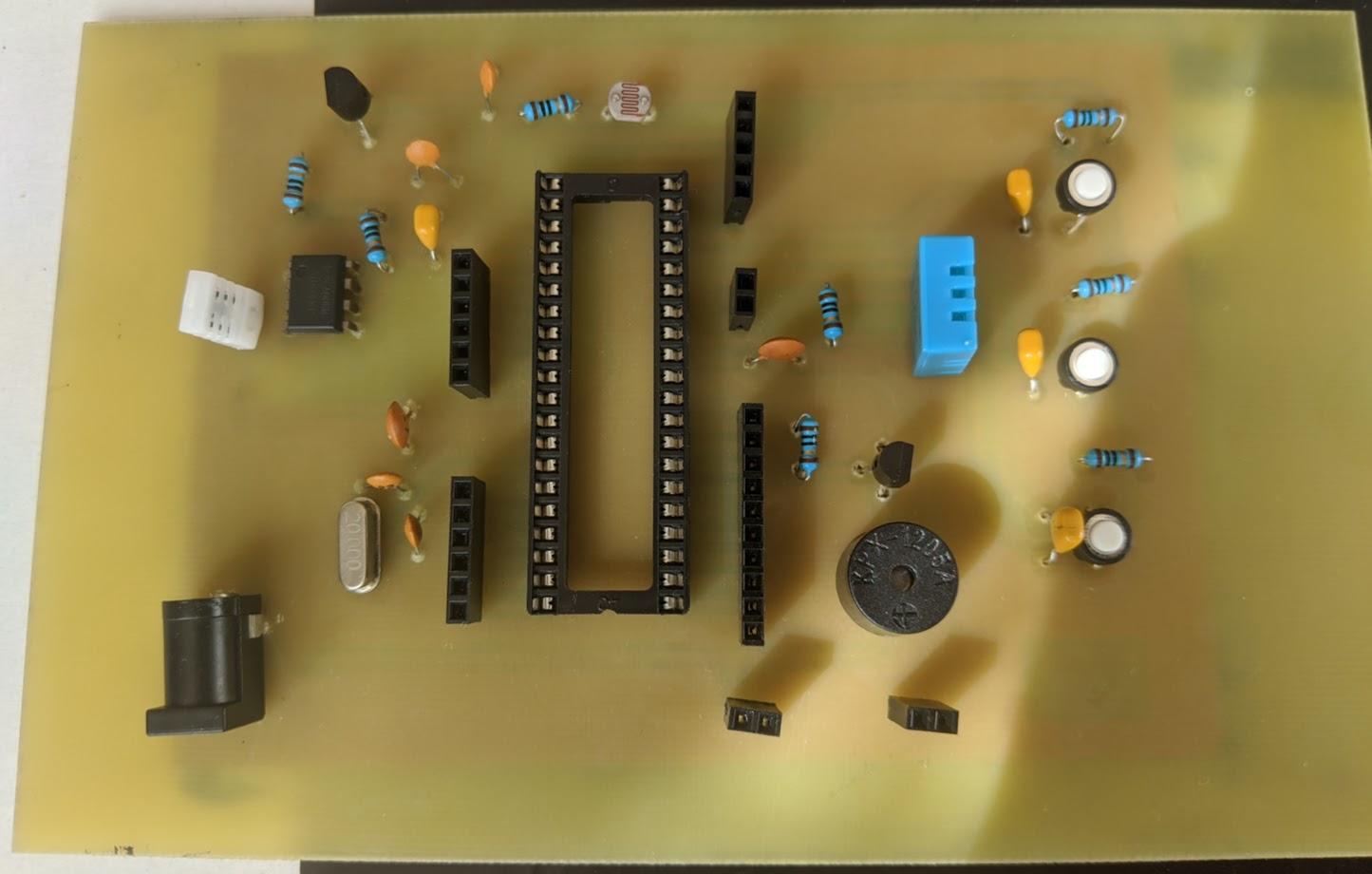


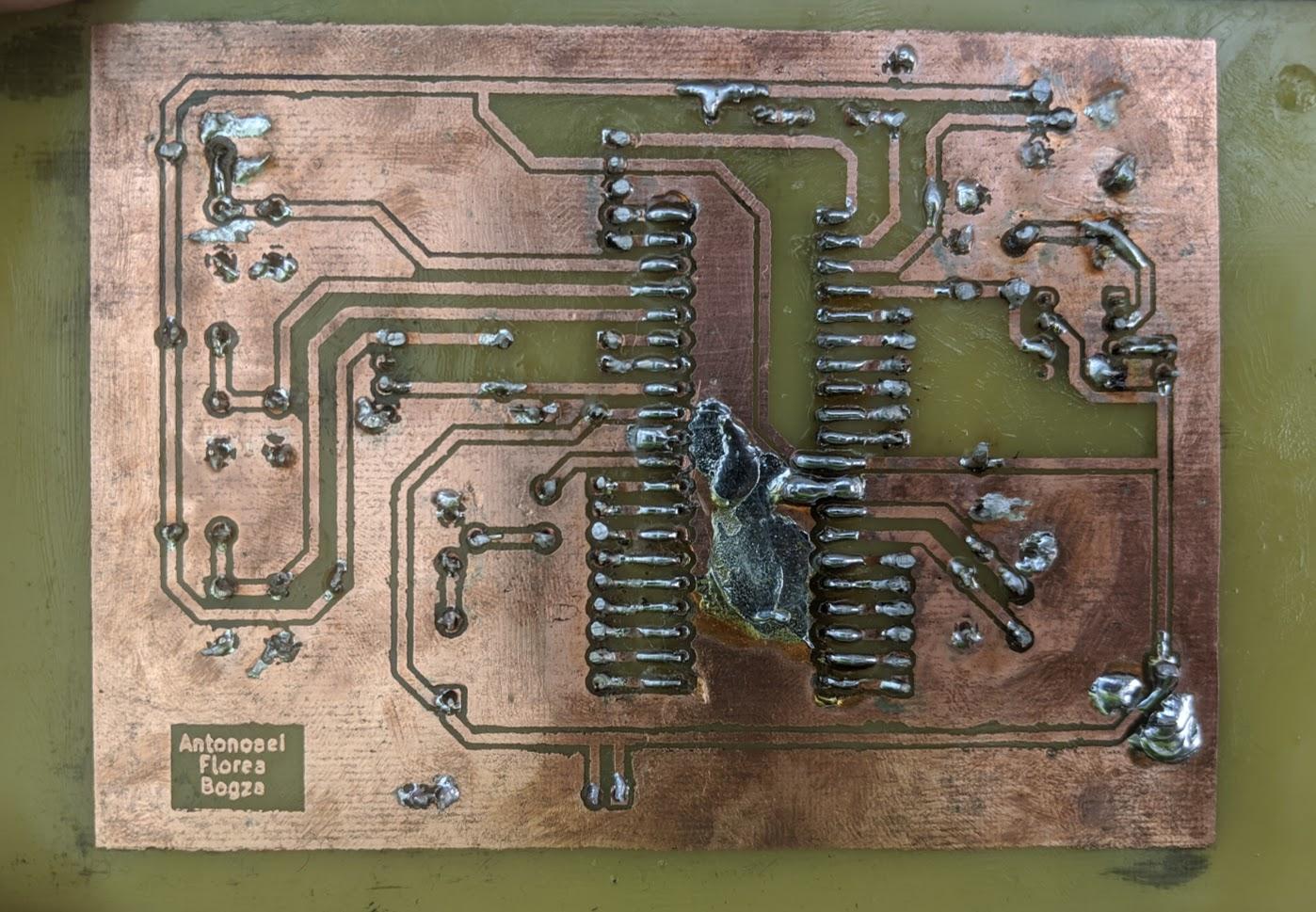
I corroded the PCB using ferric chloride.



I drilled holes in the PCB to be able to add the components.

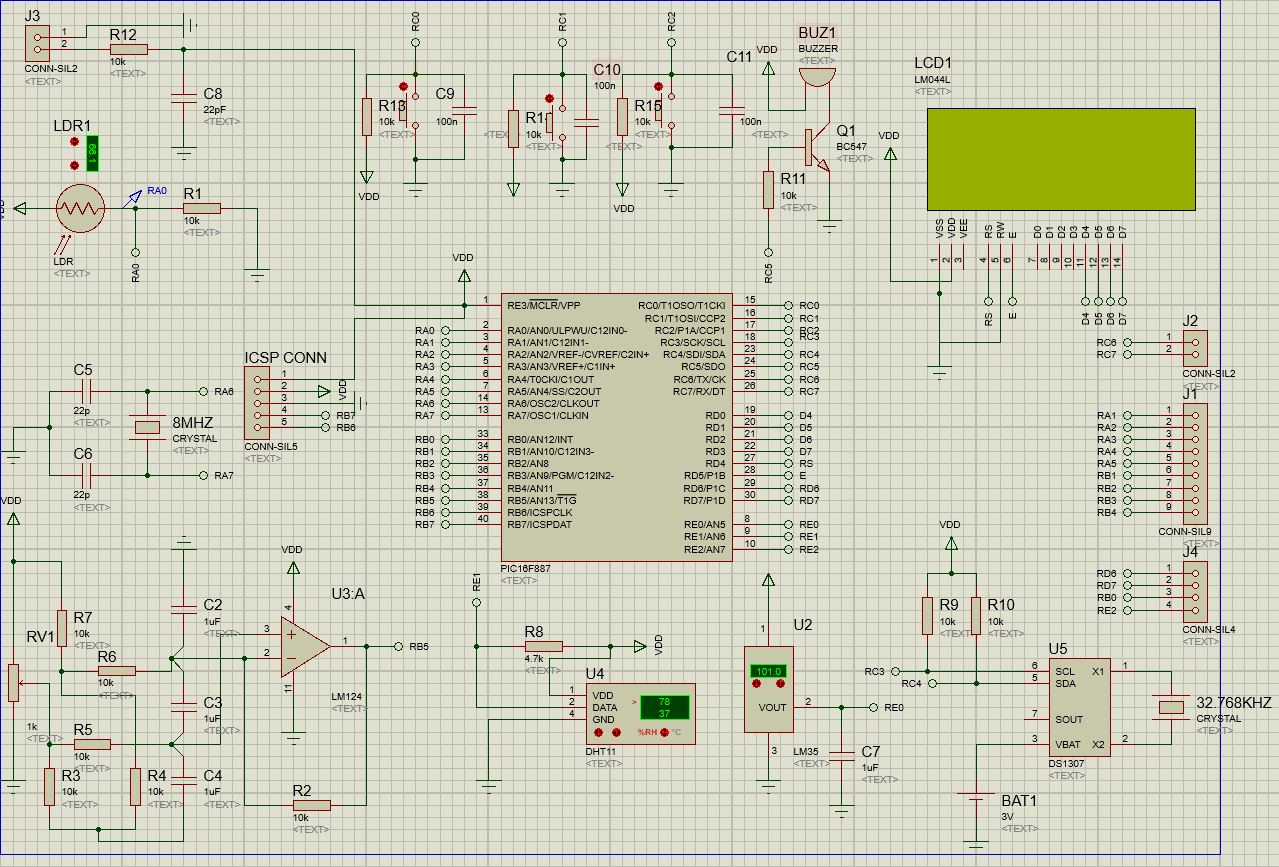
Soldered the components on the PCB



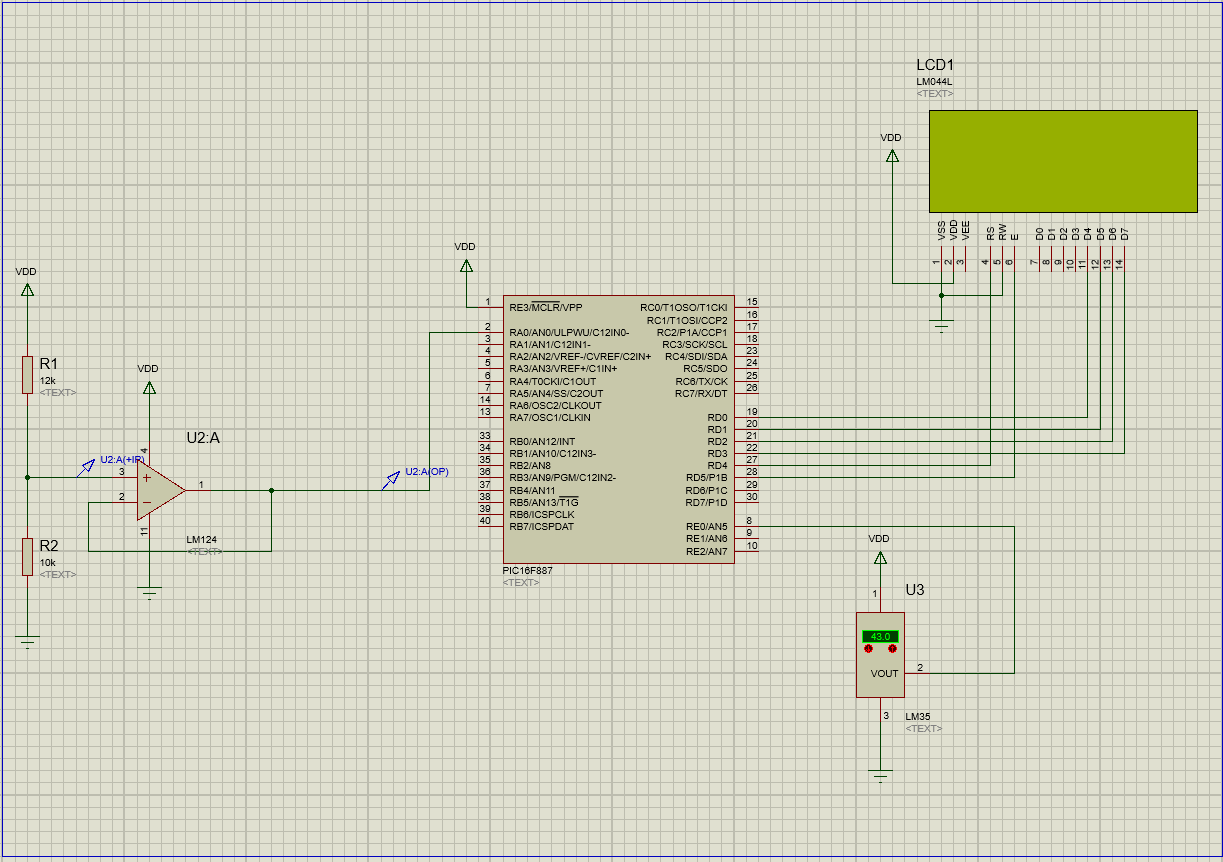


**Antonoaei Claudia-Schematic and code for Real Time clock**

I used Proteus 8.0 for my part. Dht11 can’t be found in Proteus 8.0 so we added it in 8.1.

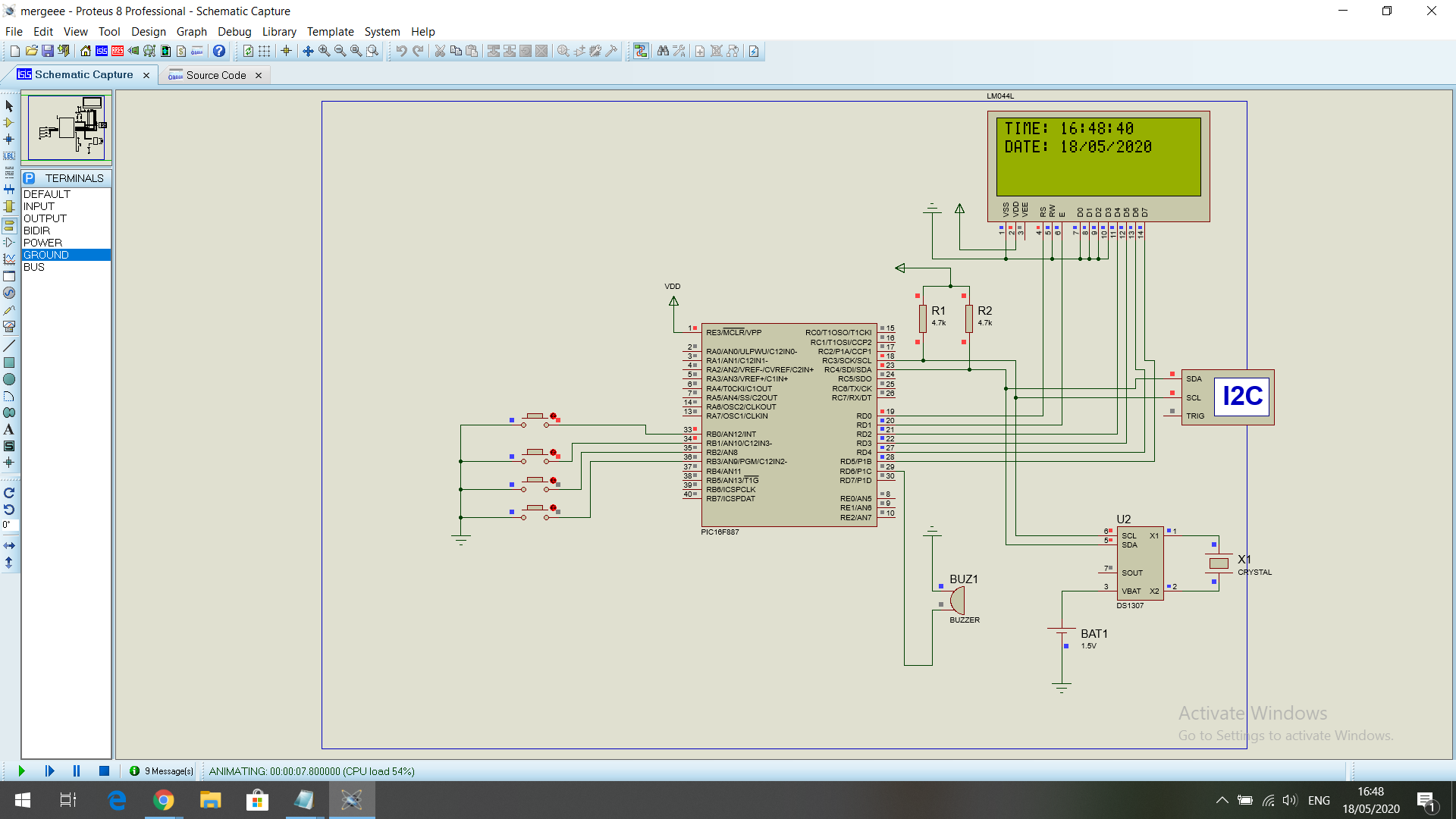
This is the final schematic we used

It should be noted that for the final code for the analog humidity sensor we used this schematic for testing, instead of the version of the amplifier in the last image.



For the schematic, all colleagues made updates as necessary along the way.

For the code part I used Proteus 8.0. I did the code for the real time clock.



As you can see from the image, the real time clock is functional. Below I will list the code that I attached as well in the sent folder. This was not the first attempt at this, in fact it took several tries to make it work.

The used component for this particular simulation was DS1307 REAL TIME CLOCK.

#pragma config CONFIG1 = 0x2CD4

#pragma config CONFIG2 = 0x0700

#define button1 RB0

#define button2 RB1

#define buttonA1 RB2

#define buttonA2 RB3

#define LCD\_RS RD0

#define LCD\_EN RD1

#define LCD\_D4 RD2

#define LCD\_D5 RD3

#define LCD\_D6 RD4

#define LCD\_D7 RD5

#define LCD\_RS\_DIR TRISD0

#define LCD\_EN\_DIR TRISD1

#define LCD\_D4\_DIR TRISD2

#define LCD\_D5\_DIR TRISD3

#define LCD\_D6\_DIR TRISD4

#define LCD\_D7\_DIR TRISD5

#include <xc.h>

#define \_XTAL\_FREQ 8000000

#include <stdint.h>

#include <xc.h>

#include <stdio.h>

#include <string.h>

uint8\_t i, second, minute, hour, m\_day, month, year;

void LCD\_Write\_Nibble(uint8\_t n);

void LCD\_Cmd(uint8\_t Command);

void LCD\_Goto(uint8\_t col, uint8\_t row);

void LCD\_PutC(char LCD\_Char);

void LCD\_Print(char\* LCD\_Str);

void LCD\_Begin();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* I2C functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void I2C\_Init(uint32\_t i2c\_clk\_freq)

{

SSPCON = 0x28;

SSPADD = (\_XTAL\_FREQ/(4 \* i2c\_clk\_freq)) - 1;

SSPSTAT = 0;

}

void I2C\_Start()

{

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

SEN = 1;

}

void I2C\_Repeated\_Start()

{

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

RSEN = 1;

}

void I2C\_Stop()

{

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

PEN = 1;

}

void I2C\_Write(uint8\_t i2c\_data)

{

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

SSPBUF = i2c\_data;

}

uint8\_t I2C\_Read(uint8\_t ack)

{

uint8\_t \_data;

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

RCEN = 1;

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

\_data = SSPBUF;

while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F));

ACKDT = !ack;

ACKEN = 1;

return \_data;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end I2C functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// a small function for button1 (B1) debounce

\_\_bit debounce ()

{

uint8\_t count = 0;

for(uint8\_t i = 0; i < 5; i++) {

if (button1 == 0)

count++;

\_\_delay\_ms(10);

}

if(count > 2) return 1;

else return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* RTC chip functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

uint8\_t bcd\_to\_decimal(uint8\_t number) {

return((number >> 4) \* 10 + (number & 0x0F));

}

uint8\_t decimal\_to\_bcd(uint8\_t number) {

return(((number / 10) << 4) + (number % 10));

}

void RTC\_display()

{

static char Time[] = "TIME: 00:00:00";

static char Date[] = "DATE: 00/00/2000";

second = bcd\_to\_decimal(second);

minute = bcd\_to\_decimal(minute);

hour = bcd\_to\_decimal(hour);

m\_day = bcd\_to\_decimal(m\_day);

month = bcd\_to\_decimal(month);

year = bcd\_to\_decimal(year);

Time[6] = hour / 10 + '0';

Time[7] = hour % 10 + '0';

Time[9] = minute / 10 + '0';

Time[10] = minute % 10 + '0';

Time[12] = second / 10 + '0';

Time[13] = second % 10 + '0';

Date[6] = m\_day / 10 + '0';

Date[7] = m\_day % 10 + '0';

Date[9] = month / 10 + '0';

Date[10] = month % 10 + '0';

Date[14] = year / 10 + '0';

Date[15] = year % 10 + '0';

LCD\_Goto(1, 1);

LCD\_Print(Time);

LCD\_Goto(1, 2);

LCD\_Print(Date);

}

void blink()

{

uint8\_t j = 0;

while(j < 100 && button1 && button2) {

j++;

\_\_delay\_ms(5);

}

}

uint8\_t edit(uint8\_t x, uint8\_t y, uint8\_t parameter)

{

while(debounce());

while(1) {

while(!button2)

{

parameter++;

if(i == 0 && parameter > 23)

parameter = 0;

if(i == 1 && parameter > 59)

parameter = 0;

if(i == 2 && parameter > 31)

parameter = 1;

if(i == 3 && parameter > 12)

parameter = 1;

if(i == 4 && parameter > 99)

parameter = 0;

LCD\_Goto(x, y);

LCD\_PutC(parameter / 10 + '0');

LCD\_PutC(parameter % 10 + '0');

\_\_delay\_ms(200);

}

LCD\_Goto(x, y);

blink();

LCD\_Goto(x, y);

LCD\_PutC(parameter / 10 + '0');

LCD\_PutC(parameter % 10 + '0');

blink();

if(!button1)

if(debounce())

{

i++;

return parameter;

}

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* end RTC chip functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* main function \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main(void)

{

OSCCON = 0X70;

ANSELH = 0;

nRBPU = 0;

WPUB = 0x03;

\_\_delay\_ms(1000);

I2C\_Init(100000);

LCD\_Begin();

while(1) {

if(!button1)

if(debounce())

{

i = 0;

hour = edit(7, 1, hour);

minute = edit(10, 1, minute);

m\_day = edit(7, 2, m\_day);

month = edit(10, 2, month);

year = edit(15, 2, year);

while(debounce());

minute = decimal\_to\_bcd(minute);

hour = decimal\_to\_bcd(hour);

m\_day = decimal\_to\_bcd(m\_day);

month = decimal\_to\_bcd(month);

year = decimal\_to\_bcd(year);

I2C\_Start();

I2C\_Write(0xD0);

I2C\_Write(0);

I2C\_Write(0);

I2C\_Write(minute);

I2C\_Write(hour);

I2C\_Write(1);

I2C\_Write(m\_day);

I2C\_Write(month);

I2C\_Write(year);

I2C\_Stop();

\_\_delay\_ms(200);

}

I2C\_Start();

I2C\_Write(0xD0);

I2C\_Write(0);

I2C\_Repeated\_Start();

I2C\_Write(0xD1);

second = I2C\_Read(1);

minute = I2C\_Read(1);

hour = I2C\_Read(1);

I2C\_Read(1);

m\_day = I2C\_Read(1);

month = I2C\_Read(1);

year = I2C\_Read(0);

I2C\_Stop();

RTC\_display();

\_\_delay\_ms(50);

}

}

Below, I will list the modified functions for the alarm as well

void alarms\_read\_display() // read and display alarm1 and alarm2 data function

{

I2C\_Start();

I2C\_Write(0xD0);

I2C\_Write(0x08);

I2C\_Repeated\_Start();

I2C\_Write(0xD1);

alarm1\_minute = I2C\_Read(1);

alarm1\_hour = I2C\_Read(1);

I2C\_Read(1);

I2C\_Read(1);

I2C\_Read(1);

control\_reg = I2C\_Read(1);

status\_reg = I2C\_Read(1);

I2C\_Read(1);

I2C\_Read(1);

I2C\_Read(1);

I2C\_Stop();

alarm1\_minute = bcd\_to\_decimal(alarm1\_minute);

alarm1\_hour = bcd\_to\_decimal(alarm1\_hour);

Alarm1[8] = alarm1\_minute % 10 + '0';

Alarm1[7] = alarm1\_minute / 10 + '0';

Alarm1[5] = alarm1\_hour % 10 + '0';

Alarm1[4] = alarm1\_hour / 10 + '0';

}

\_\_bit debounce (uint8\_t button)

{

uint8\_t count = 0;

for(uint8\_t i = 0; i < 5; i++)

{

switch (button)

{

case 1: if (button1 == 0)

count++;

break;

case 3: if (buttonA1 == 0)

count++;

break;

default: break;

}

\_\_delay\_ms(10);

}

if(count > 2) return 1;

else return 0;

}

void alarm\_check(){

if((alarm1\_minute == ((minute & 0x0F) + (minute >> 4) \* 10)) &&

(alarm1\_hour == ((hour & 0x0F) + (hour >> 4) \* 10)) && (second == 0))

Buzzer=1;

}

**Alexandru Florea-code**

Code for humidity sensor:

/\*Template Proiect AEMC - ETTI Iasi\*/

#include <xc.h>

\_\_PROG\_CONFIG(1,0x20D4); // config. uC

\_\_PROG\_CONFIG(2,0x0000); // config. uC

#define LED1 RB0

#define LED2 RB1

#define \_XTAL\_FREQ 8000000

char \*stringumid = (char \*) "umiditate analogica";

void init\_uC(void);

void interrupt etti(void); // functie de intreruperi globala ptr. TOATE intreruperile de pe un

void init\_LCD(void);

void lcd\_goto(unsigned char pos);

void lcd\_puts(char \* s);

void lcd\_putch(char c);

void senzumid(void);

unsigned int Citeste\_ADC(unsigned char canal);

int temperatura[] = {0,5,10,15,20,25,30,35,40,45,50};

const double matricetabel[11][15] = {

{0, 0, 12000, 5200, 2800, 720, 384, 200, 108, 64, 38, 23, 16 ,10.2,6.9},

{0, 19800, 9800, 4700, 2000, 510, 271, 149, 82, 48, 29, 18, 12, 8.2, 5.4},

{0, 16000, 7200, 3200, 1400, 386, 211, 118, 64, 38, 24, 15, 10.2, 6.9, 4.7},

{21000, 10500, 5100, 2350, 1050, 287, 159, 91, 51, 31, 19, 12, 8.1, 5.5, 4.1},

{13500, 6700, 3300, 1800, 840, 216, 123, 70, 40, 25, 16, 10, 7.2, 4.7, 3.2},

{9800, 4803, 2500, 1300, 630, 166, 95, 55, 31, 20, 13, 8.5, 5.7, 4.0, 2.8},

{8000, 3900, 2000, 980, 470, 131, 77, 44, 25, 17, 10.5, 7.2, 5.0, 3.6, 2.5},

{6300, 3100, 1500, 750, 385, 104, 63, 38, 21, 13, 9, 6.4, 4.4, 3.2, 2.3},

{4600, 2300, 1100, 575, 282, 80, 52, 32, 17, 11, 8.2, 5.8, 4.0, 2.9, 2.1},

{3800, 1850, 900, 430, 210, 66, 45, 30, 14, 9, 7.1, 5.0, 3.3, 2.4, 1.8},

{3200, 1550, 750, 350, 170, 51, 38, 24, 12, 8, 6.0, 4.1, 2.9, 2.0, 1.5},

};

int umid[]={20,25,30,35,40,45,50,55,60,65,70,75,80,85,90};

void main(void)

{

init\_uC();

init\_LCD();

while (1)

{

senzumid();

}

}

void init\_uC (void)

{

OSCCON = 0x71; // setez Osc. intern uC de 8MHz // pag. 64

ANSELH = 0x00;

ADCON0 = 0b10000001;

ADCON1 = 0x80;

T1CON=0x10;

TMR1H=0;

TMR1L=0;

TRISA = 0x01;

TRISB = 0x00;

TRISC = 0x01;

TRISD = 0x00;

TRISE = 0b00000011;

ANSEL = 0b00000001;

PORTB = 0b00000000; // initializez PORTB cu valori de 0 logic

OPTION\_REG = 0b00000111;// Frecv. intrare T0 = Frecv. Osc./4 (=8MHz/4) = 2MHz

// prescaler=256 => Frecv. T0 = 2MHz/256 = KHz (sau T=128us)

TMR0IF = 0; //

TMR0 = 178; // porneste numararea de la valoarea 178; pana la maxim (255)

TMR0IE = 1; // activez intreruperea ptr. T0

GIE = 1; // activez Global intreruperile

}

void interrupt etti(void) // ajung aici la fiecare 78\*128us=~10ms

{

if(TMR0IF) // daca flagul TMR0IF=1

{

TMR0IF = 0; // obligatoriu - sterg flagul Timerului care a generat intreruperea

TMR0 = 178; //

LED2 = !LED2; //

}

}

unsigned int Citeste\_ADC(unsigned char canal)

{

if(canal > 13)

return 0;

ADCON0 &= 0xC3;

ADCON0 |= (canal << 2);

\_\_delay\_ms(100);

GO\_nDONE = 1;

while(GO\_nDONE);

return ((ADRESH<<8)+ADRESL);

}

void senzumid(void)

{

int temptabel = 1, umidtabel = 0;

float templm;

templm = Citeste\_ADC(5);

templm = templm\*4.882;

templm=templm/10.0;

int i;

for(i=0;i<=10;i++)

{ if(templm>=temperatura[i])

{

temptabel++;

}

}

float imp;

imp = Citeste\_ADC(0);

imp=imp\*4.882;

imp=imp/1000.0;

imp = (10.0\*(5-imp)/imp);

if(temptabel==0)

{

umidtabel=2;

}

if(temptabel == 1)

{

umidtabel=1;

}

if(temptabel == 2)

{

umidtabel=1;

}

while(imp<=matricetabel[temptabel][umidtabel])

{

umidtabel++;

}

int val=(int)umid[umidtabel];

\_\_delay\_ms(2);

lcd\_goto(0x80);

\_\_delay\_ms(2);

lcd\_puts(stringumid);

lcd\_goto(0xC0);

lcd\_putch((val/10)%10+'0');

\_\_delay\_ms(2);

lcd\_putch((val)%10+'0');

\_\_delay\_ms(2);

lcd\_putch('%');

}

Code for LM35,photoresistor+DHT11

/\*Template Proiect AEMC - ETTI Iasi\*/

#include <xc.h>

#include<math.h>

\_\_PROG\_CONFIG(1,0x20D4); // config. uC

\_\_PROG\_CONFIG(2,0x0000); // config. uC

#define \_XTAL\_FREQ 8000000

#define LED2 RB1

#define butoninc RC2

#define butonalarm RC1

#define butontimp RC0

#define dhtpin RE1

#define dhtpin\_dir TRISE1

char \*tempanalog = (char \*) " Temperatura LM35";

char \*luminanalog = (char \*) "luminozitate";

char \*tempdht = (char \*) "temp dht";

char \*umiddht = (char \*) "umid dht";

char \*pag1 = (char \*) "pagina 1";

char \*pag2 = (char \*) "pagina 2";

char \*pag3 = (char \*) "pagina 3";

char \*lum = (char \*) "lumeni/m^2";

void init\_uC(void);

void interrupt etti(void); // functie de intreruperi globala ptr. TOATE intreruperile de pe un

void init\_LCD(void);

void lcd\_goto(unsigned char pos);

void lcd\_puts(char \* s);

void lcd\_putch(char c);

void lcd\_clear(void);

void lm35(void);

void luminozitate(void);

void initializare\_dht(void);

unsigned short raspuns\_dht(void);

unsigned short citire\_dht(unsigned short \*);

void afisare\_dht(void);

void afisarepag(void);

unsigned int Citeste\_ADC(unsigned char canal);

char temperatura[]="temperatura = 00.0C";

char umiditate[]="umiditate = 00.0%";

unsigned short tempbyte1, tempbyte2, umidbyte1, umidbyte2, suma;

void main(void)

{

init\_uC();

init\_LCD();

while(1)

{

afisarepag();

}

}

void init\_uC (void)

{

OSCCON = 0x71; // setez Osc. intern uC de 8MHz // pag. 64

ADCON0 = 0b10000001;

ADCON1 = 0x80;

T1CON=0x10;

TMR1H=0;

TMR1L=0;

TRISA = 0x01;

TRISB = 0x00;

TRISC = 0x00;

TRISD = 0x00;

TRISE = 0b00000011;

ANSEL = 0b00000001;

ANSELH = 0x00;

PORTB = 0b00000000; // initializez PORTB cu valori de 0 logic

OPTION\_REG = 0b00000111;// Frecv. intrare T0 = Frecv. Osc./4 (=8MHz/4) = 2MHz

// prescaler=256 => Frecv. T0 = 2MHz/256 = KHz (sau T=128us)

TMR0IF = 0; //

TMR0 = 178; // porneste numararea de la valoarea 178; pana la maxim (255)

TMR0IE = 1; // activez intreruperea ptr. T0

GIE = 1; // activez Global intreruperile

}

unsigned int Citeste\_ADC(unsigned char canal)

{

if(canal > 13)

return 0;

ADCON0 &= 0xC3;

ADCON0 |= (canal << 2);

\_\_delay\_ms(100);

GO\_nDONE = 1;

while(GO\_nDONE);

return ((ADRESH<<8)+ADRESL);

}

void lm35(void)

{

int adc,temp1,c1,c2,c3,c4;

float volt,temp;

adc=Citeste\_ADC(5);

volt=adc\*4.882;

temp=volt;

temp1=temp;

c1=(temp1/1000)%10;

c2=(temp1/100)%10;

c3=(temp1/10)%10;

c4=(temp1/1)%10;

\_\_delay\_ms(2);

lcd\_goto(0xC1);

\_\_delay\_ms(2);

lcd\_puts(tempanalog);

\_\_delay\_ms(2);

lcd\_goto(0x99);

\_\_delay\_ms(2);

lcd\_putch(c1+'0');

\_\_delay\_ms(2);

lcd\_putch(c2+'0');

\_\_delay\_ms(2);

lcd\_putch(c3+'0');

lcd\_puts(".");

\_\_delay\_ms(2);;

lcd\_putch(c4+'0');

\_\_delay\_ms(2);

lcd\_putch(0xDF);

\_\_delay\_ms(2);

lcd\_puts("C");

}

void luminozitate(void)

{

int adc,c1,c2,c3,c4;

float r1=10.0,curent;

float lv,lum,ressenz;

adc=Citeste\_ADC(0);

lv=adc\*4.882;

lv/=1000;

ressenz=(5-lv)/(lv/r1);

ressenz=(pow(10,5.07)\*pow(ressenz,(-0.9)))/10;

int result=(int)ressenz;

c1=(result/1000)%10;

c2=(result/100)%10;

c3=(result/10)%10;

c4=(result/1)%10;

\_\_delay\_ms(2);

lcd\_goto(0xC4);

\_\_delay\_ms(2);

lcd\_puts(luminanalog);

\_\_delay\_ms(2);

lcd\_goto(0x98);

\_\_delay\_ms(2);

lcd\_putch(c1+'0');

\_\_delay\_ms(2);

lcd\_putch(c2+'0');

\_\_delay\_ms(2);

lcd\_puts(".");

\_\_delay\_ms(2);

lcd\_putch(c3+'0');

\_\_delay\_ms(2);

lcd\_putch(c4+'0');

\_\_delay\_ms(2);

lcd\_puts("lumeni/m^2");

}

void initializare\_dht(void)

{

dhtpin\_dir = 0;

dhtpin = 0;

\_\_delay\_ms(25);

dhtpin = 1;

\_\_delay\_us(25);

dhtpin\_dir = 1;

}

unsigned short raspuns\_dht()

{

TMR1H = 0;

TMR1L = 0;

TMR1ON = 1;

while(!dhtpin && TMR1L < 100);

if(TMR1L > 99)

return 0;

else { TMR1H = 0;

TMR1L = 0;

while(dhtpin && TMR1L < 100);

if(TMR1L > 99)

{

return 0;

}

else

{

return 1;

}

}

}

unsigned short citire\_dht(unsigned short\* dhtdata)

{

short i;

\*dhtdata = 0;

for( int i = 0; i < 8; i++)

{

TMR1H=0;

TMR1L=0;

while(!dhtpin)

{

if(TMR1L > 100)

{

return 1;

}

TMR1H = 0;

TMR1L = 0;

}

while(dhtpin)

if(TMR1L > 100)

if(TMR1L > 100)

{

return 1;

}

if(TMR1L > 50)

\*dhtdata |=(1<<(7-i));

}

return 0;

}

void afisare\_dht(void)

{

initializare\_dht();

if(raspuns\_dht())

{

if(citire\_dht(&umidbyte1) || citire\_dht(&umidbyte2) || citire\_dht(&tempbyte1) || citire\_dht(&tempbyte2) || citire\_dht(&suma))

{

lcd\_clear();

lcd\_goto(0x80);

lcd\_puts((char \*) "eroare la citirea datelor");

}

else {

if(suma == ((umidbyte1 + umidbyte2 + tempbyte1 + tempbyte2) & 0xFF))

{

temperatura[14] = (tempbyte1/10 + 48);

temperatura[15] = (tempbyte1%10 + 48);

temperatura[17] = (tempbyte2/10 + 48);

umiditate[12] = (umidbyte1/10 + 48);

umiditate[13] = (umidbyte1%10 + 48);

umiditate[15] = (umidbyte2/10 + 48);

temperatura[18] = 223;

lcd\_goto(0xC2);

lcd\_puts(temperatura);

lcd\_goto(0x96);

lcd\_puts(umiditate);

}

else {

lcd\_clear();

lcd\_goto(0x80);

lcd\_puts((char\*) "verifica suma");

}

}

}

else {

lcd\_goto(0x80);

lcd\_puts((char\*) "Niciun");

lcd\_goto(0xC0);

lcd\_puts((char\*) "raspuns");

}

TMR1ON = 0;

\_\_delay\_ms(1000);

}

void afisarepag(void)

{

unsigned k=0;

do{

lcd\_goto(0x86);

lcd\_puts(pag1);

lm35();

k++;

}while(k<=50);

\_\_delay\_ms(100);

lcd\_clear();

k=0;

do{

\_\_delay\_ms(10);

lcd\_goto(0x86);

lcd\_puts(pag2);

luminozitate();

k++;

}while(k<=50);

lcd\_clear();

\_\_delay\_ms(10);

k=0;

do{

\_\_delay\_ms(10);

lcd\_goto(0x86);

lcd\_puts(pag3);

afisare\_dht();

k++;

}while(k<=50);

lcd\_clear();

\_\_delay\_ms(10);

k=0;

}

void interrupt etti(void) // ajung aici la fiecare 78\*128us=~10ms

{

if(TMR0IF) // daca flagul TMR0IF=1

{

TMR0IF = 0; // obligatoriu - sterg flagul Timerului care a generat intreruperea

TMR0 = 178; //

LED2 = !LED2; //

}

}

The following table illustrates the calculus done for the luminosity sensor.

