Aparate Electronice de Masura si Control

Proiect

Grupa 5302:

Studenti : Lambru Eusebiu-Vasilica

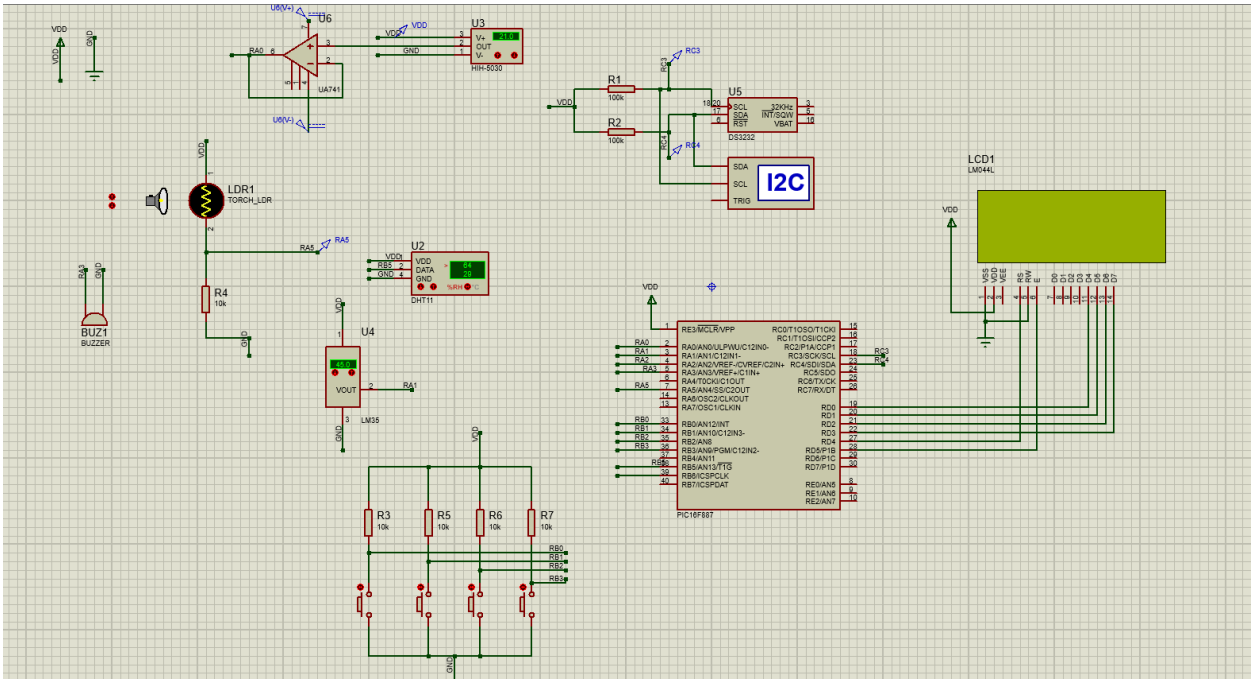
Haisan Ciprian-Constantin

Iacob Andrei

Nechita Razvan-Sebastian

Cerinte:

1) contine microcontrolerul PIC16F887.
2) contine LCD alfanumeric 2-4 linii. Template disponibil (Proteus ver. 8.0): http://ep.etc.tuiasi.ro/site/Aparate%20Electrice%20de%20Masura%20si%20Control/proiect/2020_02_13-Lab_LCD.zip
3) minim 2 senzori de temperatura: dintre care unul sa fie analogic si altul digital.
4) minim 2 senzori de umiditate: dintre care unul sa fie analogic si altul digital.
exemplu senzor de umiditate analogic:
https://www.optimusdigital.ro/ro/senzori-senzori-de-umiditate/590-senzor-rezistiv-de-umiditate-hr202l.html
5) contine un senzor de lumina (fotorezistenta/fotodioda/modul_lumina).
6) Se va proiecta un circuit cu A.O. pentru conditionarea semnalului de la senzorul de umiditate analogic.
7) Se va face o simulare a proiectului AEMC folosind programul Proteus (senzorii vor fi emulati, acolo unde este posibil).
8) Se va realiza un circuit electronic si PCB-ul. Pe partea PCB bottom se va scrie numele studentilor din echipa.
9) Se va implementa software un ceas cu data si 1 alarma. Numarul de butoane nu este impus. Se va folosi un buzzer audio (activ sau pasiv).



Aceasta este schema folosita pentru simularea functionalitatilor. Acestea sunt testate pe rand deoarece memoria microcontrollerului nu permite incarcarea in intregime a codului. Componentele folosite sunt urmatoarele:

- Microcontrollerul PIC16F887
- LCD alfanumeric 4x20 LM044L
- Senzor de temperatura analogic: LM35
- Senzor de temeratura digital: DHT11
- Senzor de umiditate analogic: HIH5030
- Senzor de umiditate digital: DHT11
- Fotorezistor (LDR)
- Amplificator operational uA741
- Real-time Clock DS3232

MICROCHIP

PIC16F882/883/884/886/887

28/40/44-Pin Flash-Based, 8-Bit CMOS Microcontrollers with nanoWatt Technology

High-Performance RISC CPU:

- Only 35 instructions to learn:
  - All single-cycle instructions except branches
- Operating speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%
  - Software selectable frequency range of 8 MHz to 31 kHz
  - Software tunable
  - Two-Speed Start-up mode
  - Crystal fail detect for critical applications
  - Clock mode switching during operation for power savings
- Power-Saving Sleep mode
- Wide operating voltage range (2.0V-5.5V)
- Industrial and Extended Temperature range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with software control option
- Enhanced low-current Watchdog Timer (WDT) with on-chip oscillator (software selectable nominal 268 seconds with full prescaler) with software enable
- Multiplexed Master Clear with pull-up/input pin
- Programmable code protection
- High Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years
- Program memory Read/Write during run time
- In-Circuit Debugger (on board)

Low-Power Features:

- Standby Current:
  - 50 nA @ 2.0V, typical
- Operating Current:
  - 11 µA @ 32 kHz, 2.0V, typical
  - 220 µA @ 4 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 µA @ 2.0V, typical

Peripheral Features:

- 24/35 I/O pins with individual direction control:
  - High current source/sink for direct LED drive
  - Interrupt-on-Change pin
  - Individually programmable weak pull-ups
  - Ultra Low-Power Wake-up (ULPWU)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (CV_{REF}) module (% of V_{DD})
  - Fixed voltage reference (0.6V)
  - Comparator inputs and outputs externally accessible
  - SR Latch mode
  - External Timer1 Gate (count enable)
- A/D Converter:
  - 10-bit resolution and 11/14 channels
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Dedicated low-power 32 kHz oscillator
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Enhanced Capture, Compare, PWM+ module:
  - 16-bit Capture, max. resolution 12.5 ns
  - Compare, max. resolution 200 ns
  - 10-bit PWM with 1, 2 or 4 output channels, programmable "dead time", max. frequency 20 kHz
  - PWM output steering control
- Capture, Compare, PWM module:
  - 16-bit Capture, max. resolution 12.5 ns
  - 16-bit Compare, max. resolution 200 ns
  - 10-bit PWM, max. frequency 20 kHz
- Enhanced USART module:
  - Supports RS-485, RS-232, and LIN 2.0
  - Auto-Baud Detect
  - Auto-Wake-Up on Start bit
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I^2C™ Master and Slave Modes with I^2C address mask

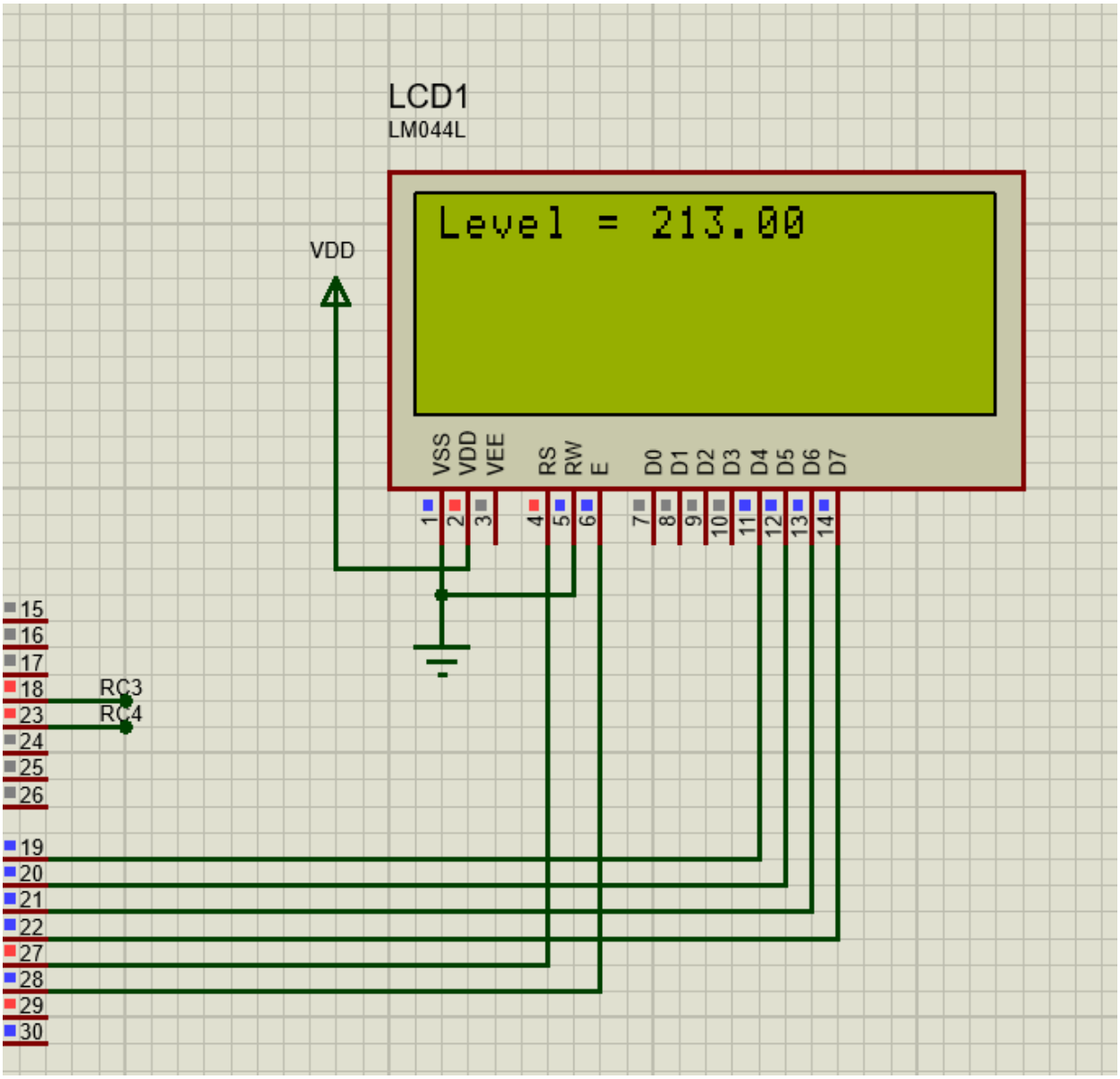
```
void init_uC(void)
{
  OSCCON = 0x71; // setez Osc. intern uC de 8MHz // pag. 64
  TRISA = 0b11111111;
  TRISB = 0xFF;       // tot Portul B este de iesire
  TRISD = 0b00000000; // tot Portul B este de iesire
  ANSEL = 0x02;
  ANSELH = 0x00;
  PORTB = 0b00000000;      // initializez PORTB cu valori 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)

}
```

LCD Alfanumeric 20x4

LM044L

INTERNAL PIN CONNECTION

Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		




```
#include <xc.h>


#define LCD_RS RD4

#define LCD_EN RD5

#define LCD_DATA PORTD

#define LCD_STROBE()   ((LCD_EN = 1), delay_LCD(1), (LCD_EN=0))
```

```
char GRADE = 0xDF;              //codul ASCII pentru simbolul "°" (grade)
char MICRO = 0xE4;              //codul ASCII pentru simbolul "µ" (micro)
char ALPHA = 0xE0;              //codul ASCII pentru simbolul "a" (alfa)
char BETA = 0xE2;               //codul ASCII pentru simbolul "ß" (beta)
char EPSILON = 0xE3;    //codul ASCII pentru simbolul "e" (epsilon)
char THETA = 0xF2;              //codul ASCII pentru simbolul "? " (theta)
char MIU = 0xE4;                //codul ASCII pentru simbolul "µ" (miu)
char OMEGA = 0xF4;              //codul ASCII pentru simbolul "O" (omega)
char SIGMAm = 0xE5;     //codul ASCII pentru simbolul "s" (sigma mic)
char RO = 0xE6;                 //codul ASCII pentru simbolul "?" (ro)
char SIGMAM = 0xF6;     //codul ASCII pentru simbolul "S" (sigma mare)
char PI = 0xF7;         //codul ASCII pentru simbolul "p" (pi)
char RADICAL = 0xE8;    //codul ASCII pentru simbolul "radical"
char MINUS1 = 0xE9;     //codul ASCII pentru simbolul "-1" (putere -1)
char INFINIT = 0xF3;    //codul ASCII pentru simbolul "infinit"
char NEGRU = 0xFF;              //codul ASCII pentru simbolul "celula neagra"
char ALB = 0xFE;                //codul ASCII pentru simbolul "celula alba"
char ms1[8]={0x00,0x00,0x0F,0x10,0x0E,0x01,0x1E,0x04}; //litera "s,"
char ma1[8]={0x0A,0x04,0x0E,0x01,0x0F,0x11,0x0F,0x00}; //litera "a~"
char ma2[8]={0x04,0x0A,0x0E,0x01,0x0F,0x11,0x0F,0x00}; //litera "a^"
char mi1[8]={0x04,0x0A,0x00,0x0C,0x04,0x04,0x0E,0x00}; //litera "i^"
char mt1[8]={0x08,0x08,0x1C,0x08,0x08,0x09,0x06,0x04}; //litera "t,"
char mL1[8]={0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01}; //caracter " |"
char mL2[8]={0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x03}; //caracter " ||"
char mL3[8]={0x07,0x07,0x07,0x07,0x07,0x07,0x07,0x07}; //caracter " |||"
char mL4[8]={0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F,0x0F}; //caracter "||||"
unsigned char a1, a2, i1, s1, t1, L1, L2, L3, L4;       //denumirile acestor variabile se aleg

        //pentru a descrie caracterul diacritic sau grafic definit de
```

```
        //utilizator in cazul de fata este vorba de diacriticile "a~", "a^", "i^",

        //"s,","t," si de simbolurile grafice" |"," ||"," ||",“ |||";
void init_LCD(void);
void lcd_write(unsigned char c);
void lcd_clear(void);
void lcd_goto(unsigned char pos);
void lcd_puts(char * s);
void lcd_putch(char c);
void delay_LCD(unsigned long t);
void initializare_diacritice(void);
void scrie_diacritice_in_CGRAM(char matrice[8], unsigned char pozitie_DDRAM,unsigned char
*diacritic);
void afisare_diacritice(unsigned char diacritic_afisat, unsigned char linia);


void init_LCD(void)
{
   delay_LCD(2000);
//    lcd_write(0x2C);
//    delay_LCD(100);
   lcd_clear();
        delay_LCD(3000);
   lcd_write(0x02);
   delay_LCD(2000);
   lcd_write(0x06);
   delay_LCD(100);
   lcd_write(0x0C);
   delay_LCD(100);
   lcd_write(0x10);
```

```
   delay_LCD(100);
   lcd_write(0x2C);
   delay_LCD(100);

        initializare_diacritice();
}

void lcd_write(unsigned char c)
{
   LCD_DATA = (LCD_DATA & 0xF0) | (c >> 4);
        LCD_STROBE();
   LCD_DATA = (LCD_DATA & 0xF0) | ( c & 0x0F );
        LCD_STROBE();
}
void lcd_clear(void)
{
        LCD_RS = 0;
        lcd_write(0x01);
}
void lcd_puts( char * s)
{
        LCD_RS = 1;       // write characters
        while(*s)
                lcd_write(*s++);
}
void lcd_putch(char c)
{
        LCD_RS = 1;       // write characters
        lcd_write( c );
```

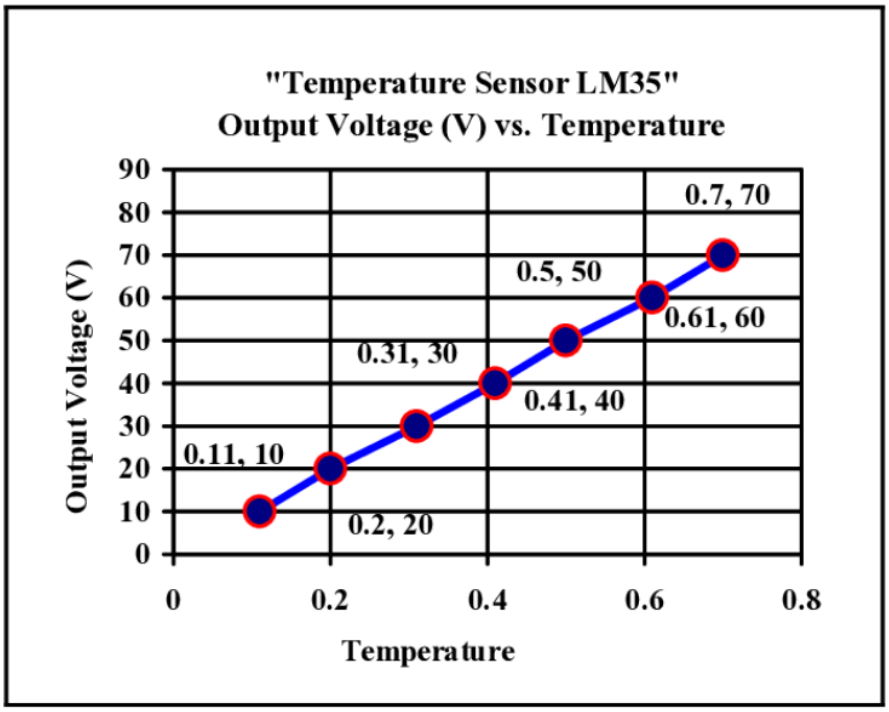
```
}
void lcd_goto(unsigned char pos)
{
        LCD_RS = 0;
        lcd_write(pos);
}
void delay_LCD(unsigned long t)
{
   unsigned long var;
   for(var=0; var < t>>6; var++);
}


void initializare_diacritice(void)
{
        scrie_diacritice_in_CGRAM(ma1,0,&a1); //litera "a~"
        scrie_diacritice_in_CGRAM(ma2,1,&a2); //litera "a^"
        scrie_diacritice_in_CGRAM(mi1,2,&i1); //litera "i^"
        scrie_diacritice_in_CGRAM(ms1,3,&s1); //litera "s,"
        scrie_diacritice_in_CGRAM(mt1,4,&t1); //litera "t,"
        scrie_diacritice_in_CGRAM(mL1,5,&L1); //simbol grafic " |"
        scrie_diacritice_in_CGRAM(mL2,6,&L2); //simbol grafic " ||"
        scrie_diacritice_in_CGRAM(mL3,7,&L3); //simbol grafic " |||"
}
void scrie_diacritice_in_CGRAM(char matrice[8], unsigned char pozitie_DDRAM, unsigned char
*diacritic)
{
        char i;
        *diacritic = pozitie_DDRAM;
        for (i=0; i<8; i++)
```

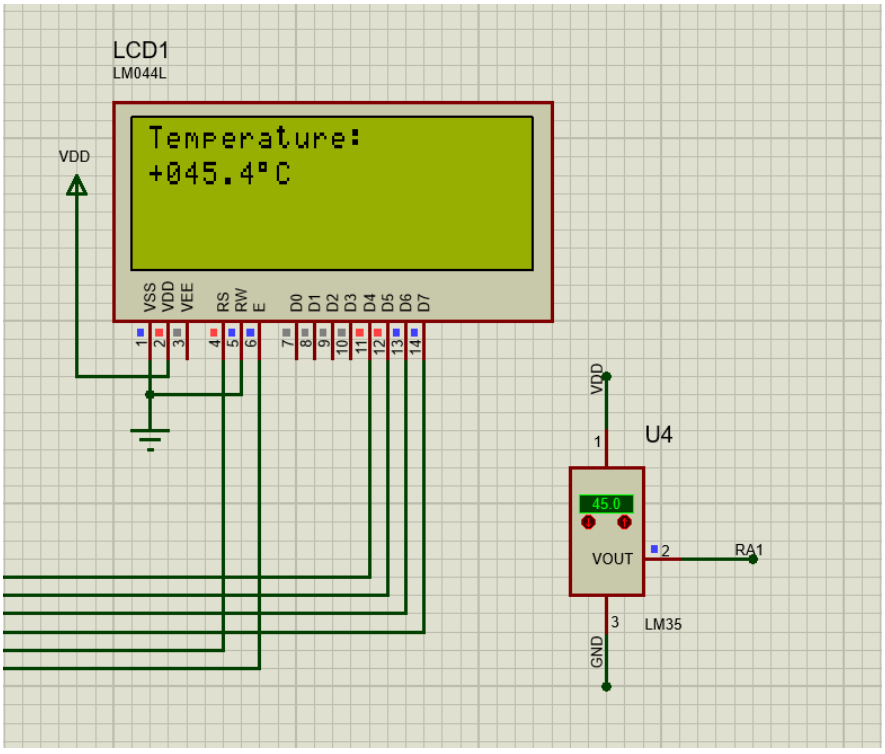
```
        {
                lcd_goto(0x40+(pozitie_DDRAM*8)+i);
                delay_LCD(100);
                lcd_putch(matrice[i]);
                delay_LCD(100);
        }
}
void afisare_diacritice(unsigned char diacritic_afisat, unsigned char linia)
{
        lcd_goto(linia);
        lcd_putch(diacritic_afisat);
}
```

Senzor de temperatura analogic: LM35

Pin Functions

PIN					TYPE	DESCRIPTION
NAME	TO46	TO92	TO220	SO8		
V_{OUT}	2	2	3	1	O	Temperature Sensor Analog Output
N.C.	—	—	—	2	—	No Connection
	—	—	—	3		
GND	3	3	2	4	GROUND	Device ground pin, connect to power supply negative terminal
N.C.	—	—	—	5	—	No Connection
	—	—	—	6		
	—	—	—	7		
$+V_S$	1	1	1	8	POWER	Positive power supply pin


"Temperature Sensor LM35"
Output Voltage (V) vs. Temperature
Output Voltage (V)
90
80
70
60
50
40
30
20
10
0
0
0.2
0.4
0.6
0.8
Temperature
0.11, 10
0.2, 20
0.31, 30
0.41, 40
0.5, 50
0.61, 60
0.7, 70


LCD1
LM044L
Temperature:
+045.4°C
VDD
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
U4
45.0
VOUT
RA1
LM35
GND

```
#include <xc.h>
#include <pic.h>
#include <pic16f887.h>
#include <math.h>
#include <stdio.h>

__PROG_CONFIG(1, 0x20D4); // config. uC
__PROG_CONFIG(2, 0x0000); // config. uC

#define _XTAL_FREQ 8000000



unsigned int ADC_Read(char channel);
void init_ADC(void);
void init_uC(void);
void init_LCD(void);
void lcd_clear(void);
void lcd_goto(unsigned char pos);
void lcd_puts(char *s);
void lcd_putch(char c);
void lcd_write(unsigned char c);
void delay_LCD(unsigned long t);
void initializare_diacritice(void);
void scrie_diacritice_in_CGRAM(char matrice[8], unsigned char pozitie_DDRAM, unsigned char
*diacritic);
void afisare_diacritice(unsigned char diacritic_afisat, unsigned char linia);
void dht11_init();
void find_response();
char read_dht11();
```

```
unsigned int read_adc();
void sterge_ecran(void);
void DHT_11_summation(void);
void read_HIH_5030(void);
void read_LM_35(void);
void Update_Current_Date_Time();
void main_RTC(void);
void Set_Time_Date();
void I2C_Initialize(const unsigned long feq_K);
void alarm(int hour, int minute, int second);
void main_LDR(void);

void read_LM_35(void)
{
 float temp = ADC_Read(1) * 5000.0 / 1024.0;
 temp = temp / 10.0;
 int s1 = (int)temp;
 int s2 = (int)((temp - (float)s1) * 10);
 lcd_goto(0x80);
 delay_LCD(100);
 char *temperature = (char *)"Temperature:";
 lcd_puts(temperature);
 char *pozitiv = (char *)"+";
 char *negativ = (char *)"-";
 lcd_goto(0xC0);
 delay_LCD(100);
 if (s1 >= 0)
 {
  lcd_puts(pozitiv);
```

```
}
if (s1 < 0)
{
  lcd_puts(negativ);
}
char str1 = (s1 / 100) + '0';
lcd_goto(0xC1);
delay_LCD(100);
lcd_putch(str1);
if (s1 >= 100)
{
  s1 = s1 % 100;
}
str1 = (s1 / 10) + '0';
lcd_goto(0xC2);
delay_LCD(100);
lcd_putch(str1);
s1 = s1 % 10;
str1 = (s1) + '0';
lcd_goto(0xC3);
delay_LCD(100);
lcd_putch(str1);
str1 = s2 + '0';
lcd_goto(0xC4);
delay_LCD(100);
lcd_putch('.');
lcd_goto(0xC5);
delay_LCD(100);
lcd_putch(str1);
```

```
  lcd_goto(0xC6);
  delay_LCD(100);
  afisare_diacritice(0xDF, 0xC6);
  lcd_goto(0xC7);
  lcd_putch('C');
}
```

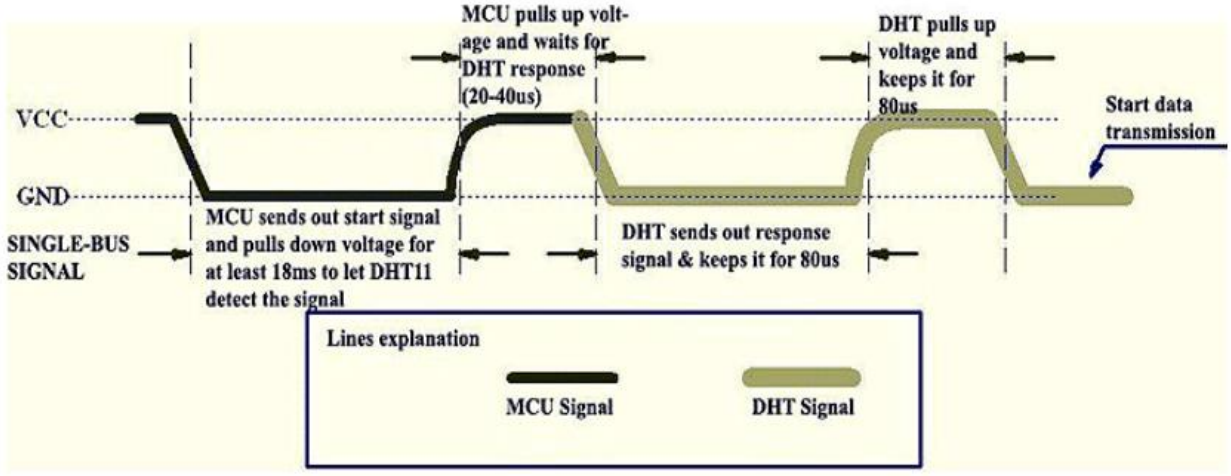
Senzor de temperatura si umiditate digital: DHT11

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

\

Tabelul de mai sus specifica acuratetea senzorului pt masurarea temperaturii si a umiditatii .

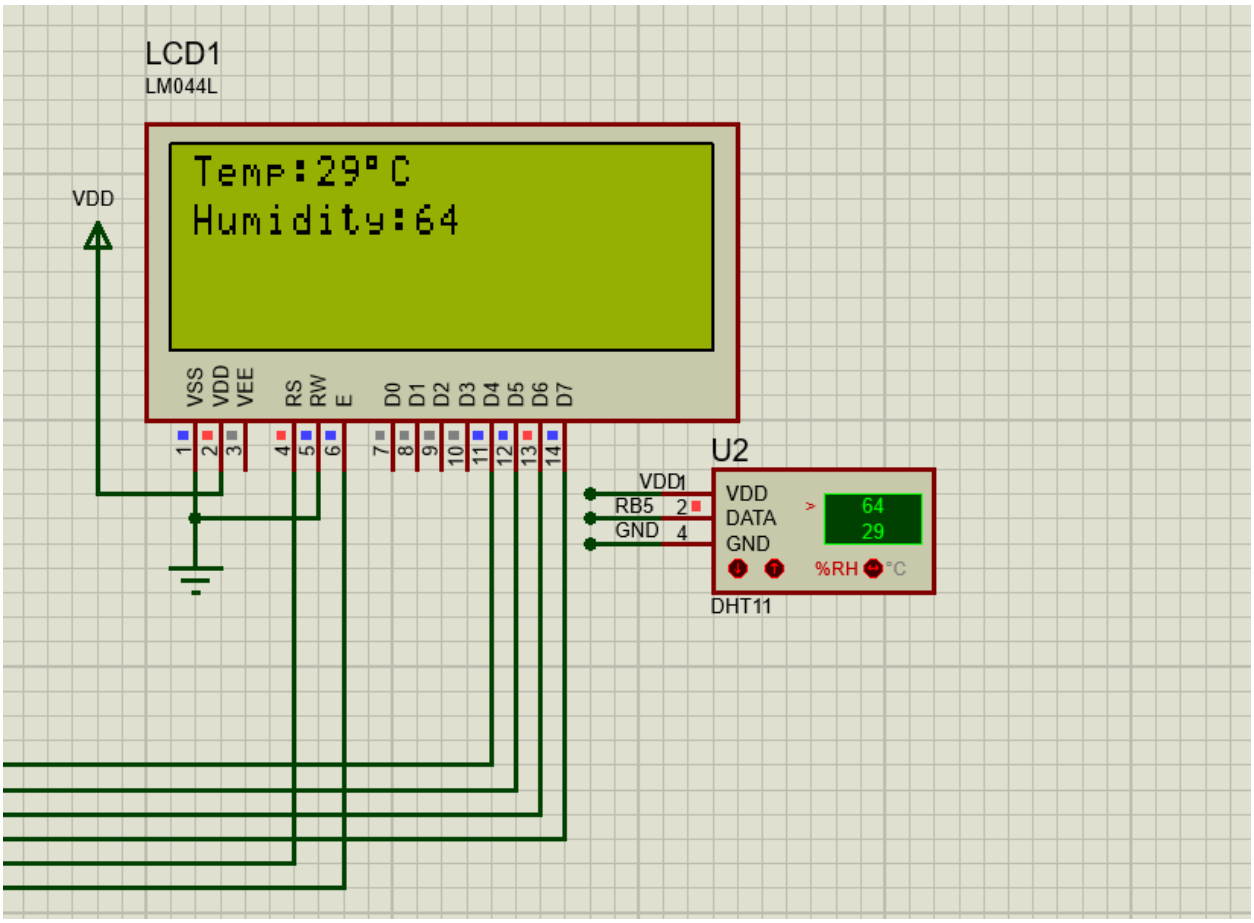
RH -> acuratete 20-90% +/- 5%RH.



DHT11 are nevoie de un timp de tranzitie de minim 18ms pentru trecerea de HIGH in LOW.

Senzorul detecteaza perioada de start a semnalului si data line-ul il face HIGH pentru 20-40us.

8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit checksum.




```
#include <xc.h>

#include <pic.h>

#include <pic16f887.h>

#include <math.h>

#include <stdio.h>


__PROG_CONFIG(1, 0x20D4); // config. uC

__PROG_CONFIG(2, 0x0000); // config. uC
```

```
#define LED1 RB0
#define LED2 RB1
#define _XTAL_FREQ 8000000
#define output_humidity RB5 //eticheta output senzor DHT11
#define analog_input RA2

void init_LCD(void);
void lcd_clear(void);
void lcd_goto(unsigned char pos);
void lcd_puts(char *s);
void lcd_putch(char c);
void lcd_write(unsigned char c);
void delay_LCD(unsigned long t);
void initializare_diacritice(void);
void dht11_init();
void find_response();
char read_dht11();
unsigned int read_adc();
void sterge_ecran(void);
void DHT_11_summation(void);
//var. pt prelucrarea datelor de la senzorul DHT11
unsigned char Check_bit, Temp_byte_1, Temp_byte_2, RH_byte_1, RH_byte_2;
unsigned char Himudity, RH, Sumation;

void DHT_11_summation()
{
  dht11_init();
  find_response();
```

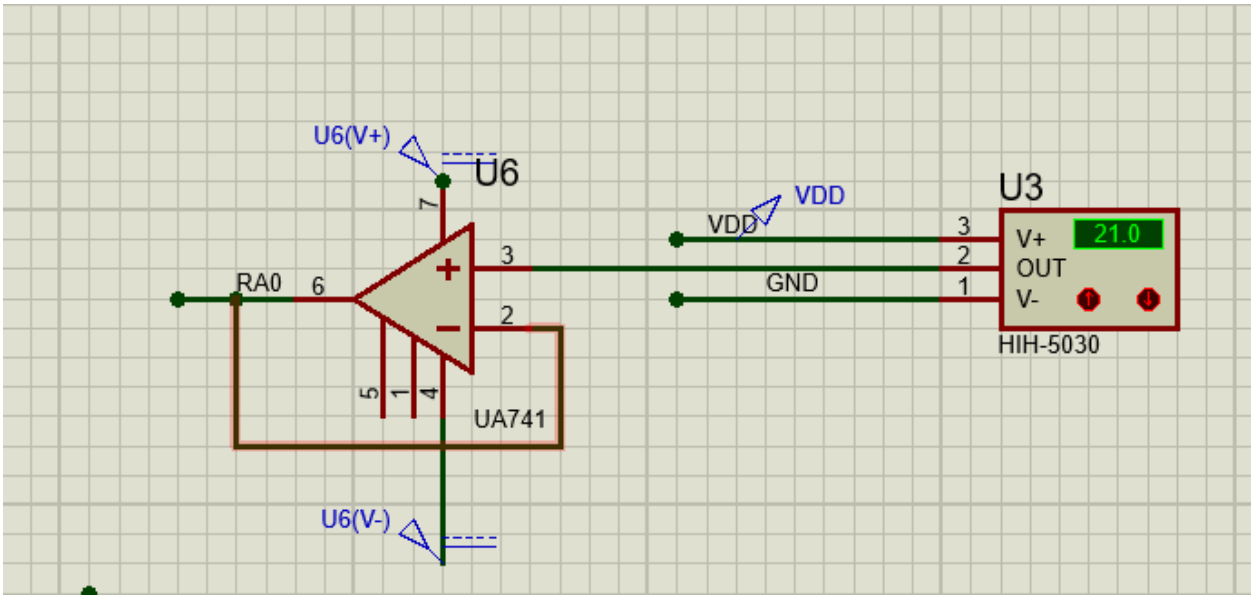
```
if (Check_bit == 1)
{
 RH_byte_1 = read_dht11();
 RH_byte_2 = read_dht11();
 Temp_byte_1 = read_dht11();
 Temp_byte_2 = read_dht11();
 Sumation = read_dht11();
 if (Sumation == ((RH_byte_1 + RH_byte_2 + Temp_byte_1 + Temp_byte_2) & 0XFF))
 {
  Himudity = Temp_byte_1;
  RH = RH_byte_1;
  lcd_goto(0x80);
  char *temp_mesaj = (char *)"Temp:";
  lcd_puts(temp_mesaj);
  lcd_putch(48 + ((Himudity / 10) % 10));
  lcd_putch(48 + (Himudity % 10));
  lcd_putch(0xDF);
  lcd_putch(0x43);
  lcd_goto(0xC0); //valoare pentru a doua linie din LCD
  char *umiditate_afisare = (char *)"Humidity:";
  lcd_puts(umiditate_afisare);
  lcd_putch(48 + ((RH / 10) % 10));
  lcd_putch(48 + (RH % 10));

 }
 else
 {
  char *sum_error = (char *)"Check sum error";
  lcd_puts(sum_error);
```

```
   }
  }
  else
  {
   char *error = (char *)"Error ";
   char *no_response = (char *)"No response ";
   sterge_ecran();
   lcd_goto(0x80);
   lcd_puts(error);
   lcd_goto(0xC0);
   lcd_puts(no_response);
  }
  __delay_ms(1000);
}

//functie initializare timpi intarziere +citire DHT11
//functie output ->output_humidity
void dht11_init()
{
  TRISB = 0b00000000;
  output_humidity = 0; //RE0 trimite 0 catre senzor
  __delay_ms(18);
  output_humidity = 1; //RE0 trimitr 1 catre senzor
  __delay_us(30);
  TRISB = 0b00100000; //RE0 configurat ca input
}

/*
 * Aceasta functie verifica daca DHT11 primeste raspuns
```

```
 */

void find_response()
{
  Check_bit = 0;
  __delay_us(40);
  if (output_humidity == 0)
  {
    __delay_us(80);
    if (output_humidity == 1)
    {
      Check_bit = 1;
    }
    __delay_us(50);
  }
}

/*
  Aceasta functie este pentru citirea senzorului DHT11
 */

char read_dht11()
{
  char data, for_count;
  for (for_count = 0; for_count < 8; for_count++)
  {
    while (!output_humidity)
      ;
    __delay_us(30);
```

```
    if (output_humidity == 0)
    {
      data &= ~(1 << (7 - for_count)); //Clear bit (7-b)
    }
    else
    {
      data |= (1 << (7 - for_count)); //Set bit (7-b)
      while (output_humidity)
        ;
    } //Asteapta pana cand PORT0 ajunge low
  }
  return data;
}

void sterge_ecran(void)
{
  char *space = (char *)"              ";
  delay_LCD(100);
  lcd_goto(0x80);
  lcd_puts(space);
  delay_LCD(100);
  lcd_goto(0xC0); //linia 2
  lcd_puts(space);
}
```

Senzor de umiditate analogic: HIH5030



S-a folosit amplificatorul operational pentru conditionarea impedantei semnalului de intrare.

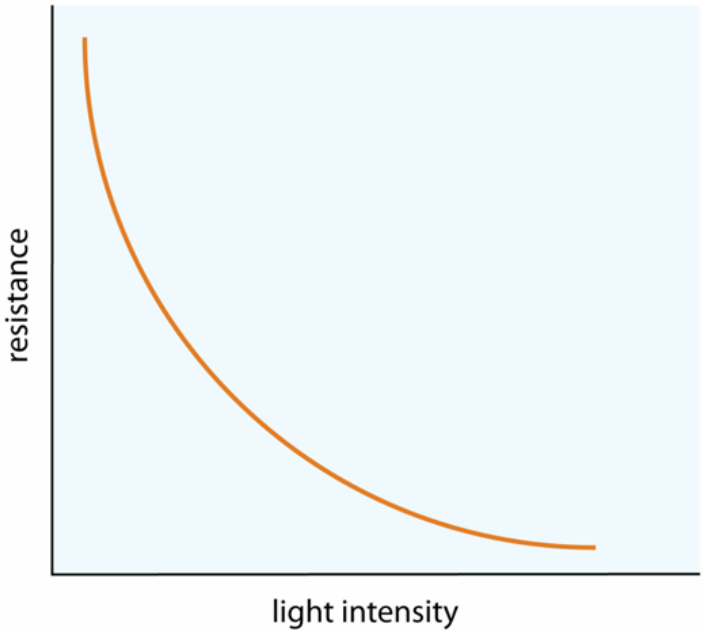
Din datasheet:

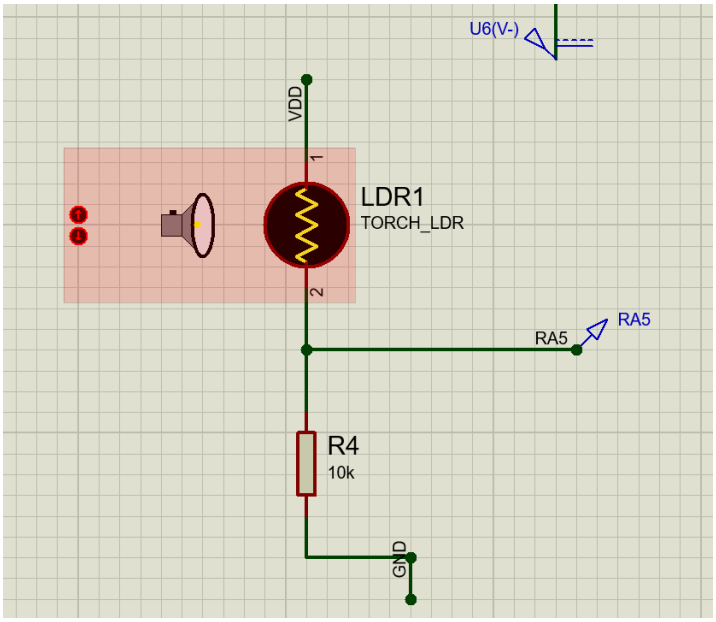
```
//#Honeywell HIH-5030 Formula Sensor Voltage Output (1st order curve fit)
//#Vout = Vsupply(0.00636(Sensor RH%)+ 0.1515), (Typical at 25 'C)

Pentru aplicatia noastra, aceasta relatie se transorma in:

Sensor RH = (voltage + 0.0332) * 32.25 - 25.81;
```

```
#include <xc.h>
#include <pic.h>
#include <pic16f887.h>
#include <math.h>
#include <stdio.h>

__PROG_CONFIG(1, 0x20D4); // config. uC
__PROG_CONFIG(2, 0x0000); // config. uC

#define _XTAL_FREQ 8000000


unsigned int ADC_Read(char channel);
void init_ADC(void);
void init_uC(void);
void interrupt etti(void); // functie de intreruperi globala ptr. TOATE intreruperile de pe un
void init_LCD(void);
void lcd_clear(void);
```

```
void lcd_goto(unsigned char pos);
void lcd_puts(char *s);
void lcd_putch(char c);
void lcd_write(unsigned char c);
void delay_LCD(unsigned long t);
void initializare_diacritice(void);
void scrie_diacritice_in_CGRAM(char matrice[8], unsigned char pozitie_DDRAM, unsigned char
*diacritic);
void afisare_diacritice(unsigned char diacritic_afisat, unsigned char linia);
void dht11_init();
void find_response();
char read_dht11();
unsigned int read_adc();
void sterge_ecran(void);
void DHT_11_summation(void);
void read_HIH_5030(void);
void read_LM_35(void);
void Update_Current_Date_Time();
void main_RTC(void);
void Set_Time_Date();
void I2C_Initialize(const unsigned long feq_K);
void alarm(int hour, int minute, int second);
void main_LDR(void);

void read_HIH_5030(void)
{
  ANSEL = 0xFF;
  ADCON0 = 0b11000001;
  ADCON1 = 0;
  TRISA = 0xFF; //toti pinii sunt de intrare
  char voltage_response[10];
  char adc_response[10];
  float adcValue = 0;  //ia valoarea de la ADC
  float voltage = 0;   //valoarea ADC converted to voltage
  float percentRH = 0; //din voltage in procent
  char RH_response[10];
  adcValue = read_adc();
  voltage = 5 * adcValue / 255; //transormam din ADC in tensiune
  percentRH = (voltage + 0.0332) * 32.25 - 25.81;
  sprintf(RH_response, " RH = %.2f", percentRH);
  delay_LCD(800);
  lcd_goto(0x80);
  lcd_puts(RH_response);
}
```

Fotorezistor (LDR)





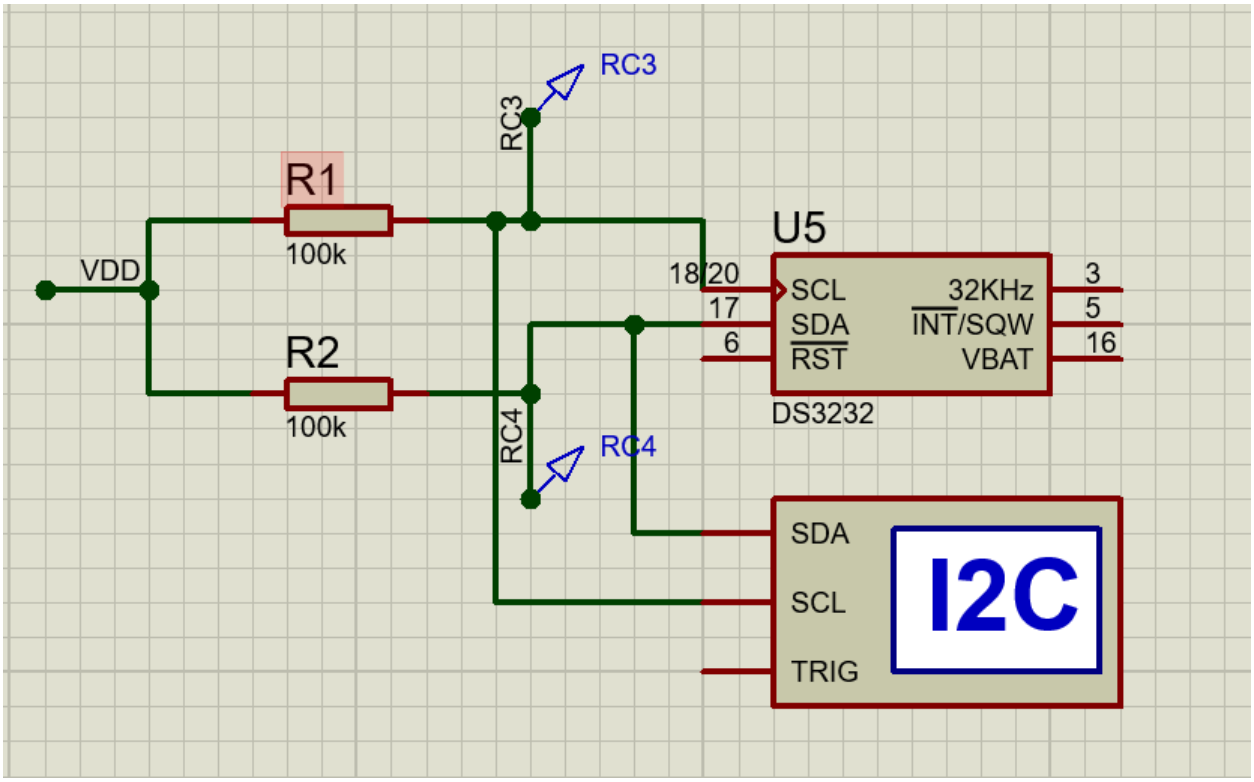

```
#include <xc.h>
#include <pic.h>
#include <pic16f887.h>
#include <math.h>
#include <stdio.h>
```

```
__PROG_CONFIG(1, 0x20D4); // config. uC
__PROG_CONFIG(2, 0x0000); // config. uC

#define _XTAL_FREQ 8000000



unsigned int ADC_Read(char channel);
void init_ADC(void);
void init_uC(void);
void init_LCD(void);
void lcd_clear(void);
void lcd_goto(unsigned char pos);
void lcd_puts(char *s);
void lcd_putch(char c);
unsigned int read_adc();
void main_LDR(void);


void main_LDR(void)
{
  ANSEL = 0xFF;
  ADCON0 = 0b11010001;
  ADCON1 = 0;
  TRISA = 0xFF; //toti pinii sunt de intrare
  float lightLevel;
  lightLevel = read_adc();
  char s[20];
  sprintf(s, "Level = %.2f", lightLevel);
  lcd_goto(0x80);
```

```
  lcd_puts(s);
}
```

Real-time Clock DS3232

Pentru alarma am folosit un RTC DS3232 ce comunica cu microcontrollerul prin intermediul protocolului I2C.




```
/*Template Proiect AEMC - ETTI Iasi*/
#include <xc.h>
#include <pic.h>
#include <pic16f887.h>
#include <math.h>
#include <stdio.h>
#define LEFT RB0
#define RIGHT RB1
```

```
#define SET RB2
__PROG_CONFIG(1, 0x20D4); // config. uC
__PROG_CONFIG(2, 0x0000); // config. uC

#define _XTAL_FREQ 8000000
void lcd_clear(void);
void lcd_goto(unsigned char pos);
void lcd_puts(char *s);
void lcd_putch(char c);
void lcd_write(unsigned char c);
void delay_LCD(unsigned long t);
void Set_Time_Date();
int BCD_2_DEC(int to_convert);
int DEC_2_BCD(int to_convert);
void Update_Current_Date_Time();
void I2C_Wait();
void I2C_Stop();
void I2C_Start();
void I2C_Write(unsigned data);
unsigned short I2C_Read(unsigned short ack);
void I2C_Repeated_Start();
void I2C_Initialize(const unsigned long feq_K);
void I2C_Begin();
void I2C_Hold();
void I2C_End();
void alarm(int hour, int second, int minute);

/*Setam valorile curente ale datei si orei*/
int sec = 0;
```

```
int min = 3;
int hour = 2;
int date = 06;
int month = 05;
int year = 18;
/*Setam data si ora*/

void main_RTC(void)
{


  Update_Current_Date_Time(); //Citim data si ora curenta de pe RTC

  //Le separam in caractere pentru a le afisa pe LCD
  char sec_0 = sec % 10;
  char sec_1 = (sec / 10);
  char min_0 = min % 10;
  char min_1 = min / 10;
  char hour_0 = hour % 10;
  char hour_1 = hour / 10;
  char date_0 = date % 10;
  char date_1 = date / 10;
  char month_0 = month % 10;
  char month_1 = month / 10;
  char year_0 = year % 10;
  char year_1 = year / 10;

  //Afisam ora pe LCD
  lcd_goto(0x80);
```

```
  lcd_puts(" TIME: ");
  lcd_putch(hour_1 + '0');
  lcd_putch(hour_0 + '0');
  lcd_putch(':');
  lcd_putch(min_1 + '0');
  lcd_putch(min_0 + '0');
  lcd_putch(':');
  lcd_putch(sec_1 + '0');
  lcd_putch(sec_0 + '0');

  //Afisam data pe LCD
  lcd_goto(0xC0);
  lcd_puts("DATE: ");
  lcd_putch(date_1 + '0');
  lcd_putch(date_0 + '0');
  lcd_putch(':');
  lcd_putch(month_1 + '0');
  lcd_putch(month_0 + '0');
  lcd_putch(':');
  lcd_putch(year_1 + '0');
  lcd_putch(year_0 + '0');
  __delay_ms(500); //Refresh la fiecare 0.5 secunde
}

void Set_Time_Date()
{
  I2C_Begin();
  I2C_Write(0xD0);
  I2C_Write(0);
```

```
  I2C_Write(DEC_2_BCD(sec));
  I2C_Write(DEC_2_BCD(min));
  I2C_Write(DEC_2_BCD(hour));
  I2C_Write(1);
  I2C_Write(DEC_2_BCD(date));
  I2C_Write(DEC_2_BCD(month));
  I2C_Write(DEC_2_BCD(year));
  I2C_End();
}

int BCD_2_DEC(int to_convert)
{
  return (to_convert >> 4) * 10 + (to_convert & 0x0F);
}

int DEC_2_BCD(int to_convert)
{
  return ((to_convert / 10) << 4) + (to_convert % 10);
}

void Update_Current_Date_Time()
{
  //Incepe citirea
  I2C_Begin();
  I2C_Write(0xD0);
  I2C_Write(0);
  I2C_End();

  //Citireste
```

```
  I2C_Begin();
  I2C_Write(0xD1);
  sec = BCD_2_DEC(I2C_Read(1));
  min = BCD_2_DEC(I2C_Read(1));
  hour = BCD_2_DEC(I2C_Read(1));
  I2C_Read(1);
  date = BCD_2_DEC(I2C_Read(1));
  month = BCD_2_DEC(I2C_Read(1));
  year = BCD_2_DEC(I2C_Read(1));
  I2C_End();

  //Opreste citirea
  I2C_Begin();
  I2C_Write(0xD1);
  I2C_Read(1);
  I2C_End();
}

void I2C_Repeated_Start()
{
  while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F))
    ;
  RSEN = 1;
}

void I2C_Init(int i2c_clk_freq)
{
  SSPCON = 0x28;
  SSPADD = (_XTAL_FREQ / (4 * i2c_clk_freq)) - 1;
```

```
  SSPSTAT = 0;
}

void I2C_Initialize(const unsigned long feq_K) //Initializeza I2C ca master
{
  ANS3 = 0;
  TRISA3 = 0;
  RA3 = 0;
  TRISC3 = 1;
  TRISC4 = 1; //Setam SDA si SCL ca input

  SSPCON = 0b00101000;
  SSPCON2 = 0b00000000;

  SSPADD = (_XTAL_FREQ / (4 * feq_K * 100)) - 1; //Setam Clock Speed pg99/234
  SSPSTAT = 0b00000000;                    //pg83/234
}

void I2C_Hold()
{
  while ((SSPCON2 & 0b00011111) || (SSPSTAT & 0b00000100))
    ; //verificam registrii ca I2c sa nu comunice deja
}

void I2C_Begin()
{
  I2C_Hold();
  SEN = 1;
}
```

```
void I2C_End()
{
  I2C_Hold();
  PEN = 1;
}

void I2C_Write(unsigned data)
{
  I2C_Hold();
  SSPBUF = data;
}

unsigned short I2C_Read(unsigned short ack)
{
  unsigned short incoming;
  I2C_Hold();
  RCEN = 1;

  I2C_Hold();
  incoming = SSPBUF;

  I2C_Hold();
  ACKDT = (ack) ? 0 : 1; //verifica daca am primit ack
  ACKEN = 1;

  return incoming;
}
```

Functia de alarma:

```
void alarm(int hours, int minute, int second )
{

 int sec, min, hour, year, date, month;
 __delay_ms(200);
 I2C_Begin();
 I2C_Write(0xD0);
 I2C_Write(0);
 I2C_Repeated_Start();
 I2C_Write(0xD1);
 sec = I2C_Read(1);
 min = I2C_Read(1);
 hour = I2C_Read(1);
 I2C_Read(1);
 date = I2C_Read(1);
 month = I2C_Read(1);
 year = I2C_Read(0);
 I2C_End();
 sec = (sec >> 4) * 10 + (sec & 0x0F);
 min = (min >> 4) * 10 + (min & 0x0F);
 hour = (hour & 0b00010000) * 10 + (hour & 0x0F);

 char b[15];
 sprintf(b, "Time: %d%d:%d%d:%d%d", hour / 10, hour % 10, min / 10, min % 10, sec / 10, sec % 10);
 lcd_goto(0x94);
 lcd_puts(b);
```

```
  char c[15];
  sprintf(c, "Alarm: %d%d:%d%d:%d%d", hours / 10, hours % 10, minute / 10, minute % 10, second / 10, second % 10);
  lcd_goto(0xD4);
  if (hours != 99 && minute != 99 && second != 99)
    lcd_puts(c);

  if ((hours == hour) && (minute == min) && (second == sec))
  {
    lcd_clear();
    char d[20];

    sprintf(d, "   TREZIREA!!!!");
    lcd_goto(0x80);
    lcd_puts(d);
    RA3 = 1;
  }
}
```

Codul main final:

In acest cod, fiecare functie va fi decomentata pe rand:

```
/*Template Proiect AEMC - ETTI Iasi*/
#include <xc.h>
#include <pic.h>
#include <pic16f887.h>
#include <math.h>
#include <stdio.h>
```

```
__PROG_CONFIG(1, 0x20D4); // config. uC
__PROG_CONFIG(2, 0x0000); // config. uC

#define _XTAL_FREQ 8000000



unsigned int ADC_Read(char channel);
void init_ADC(void);
void init_uC(void);
void interrupt etti(void); // functie de intreruperi globala ptr. TOATE intreruperile de pe un
void init_LCD(void);
void lcd_clear(void);
void lcd_goto(unsigned char pos);
void lcd_puts(char *s);
void lcd_putch(char c);
void lcd_write(unsigned char c);
void delay_LCD(unsigned long t);
void initializare_diacritice(void);
void scrie_diacritice_in_CGRAM(char matrice[8], unsigned char pozitie_DDRAM, unsigned char
*diacritic);
void afisare_diacritice(unsigned char diacritic_afisat, unsigned char linia);
void dht11_init();
void find_response();
char read_dht11();
unsigned int read_adc();
void sterge_ecran(void);
void DHT_11_summation(void);
void read_HIH_5030(void);
void read_LM_35(void);
```

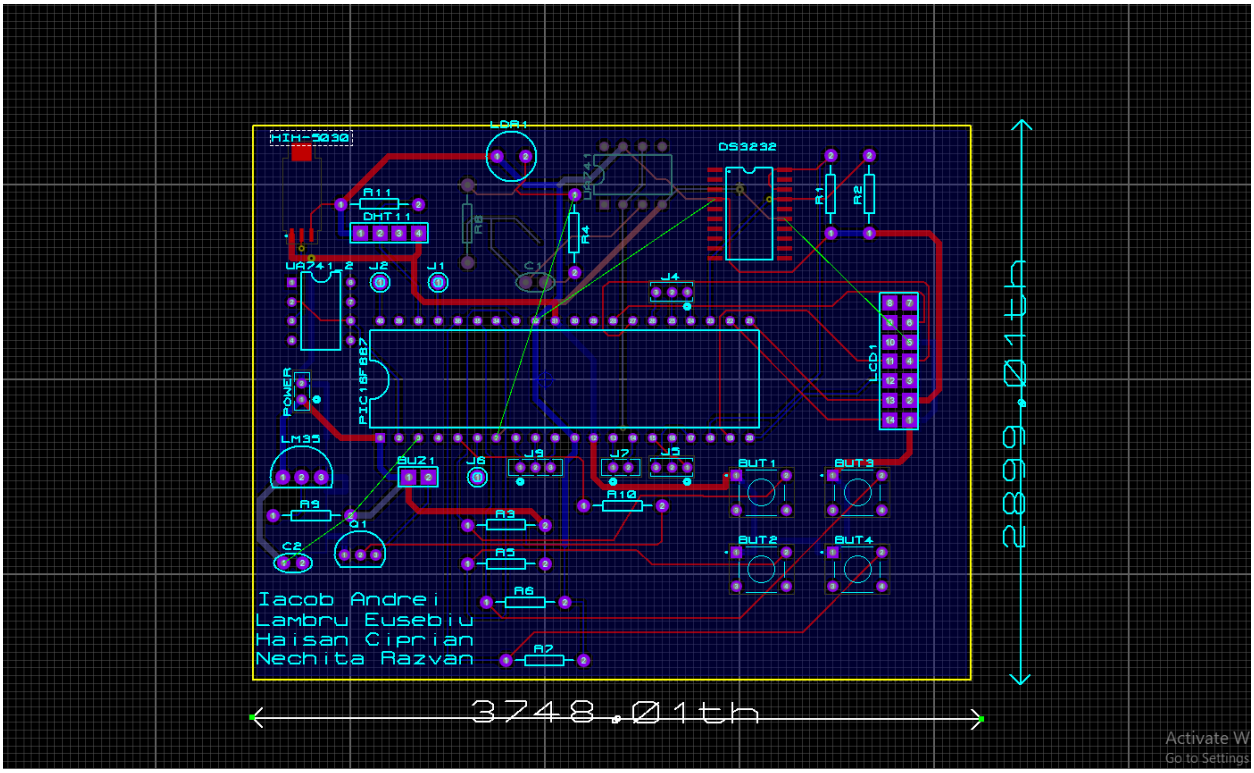
```
void Update_Current_Date_Time();
void main_RTC(void);
void Set_Time_Date();
void I2C_Initialize(const unsigned long feq_K);
void alarm(int hour, int minute, int second);
void main_LDR(void);

int hour, minute, second;

void main(void)
{
  init_uC();
  init_LCD();
  init_ADC();
  //I2C_Initialize(100);
  //Set_Time_Date();


  while (1)
  {
/*
    if (RB0 == 0)
      hour++;
    if (hour == 25)
      hour = 0;
    if (RB1 == 0)
      minute++;
    if (minute == 61)
      minute = 0;
```

```
    if (RB2 == 0)
      second++;
    if (second == 61)
      second = 0;
    if (RB3 == 0)
    {
      lcd_clear();
      RA3 = 0;
      hour = second = minute = 0;
    }
    alarm(hour, minute, second);*/
  //main_LDR(); //decom
    DHT_11_summation(); //anyway
  // read_HIH_5030(); //anyway
 //read_LM_35(); //com

  }
}

void init_ADC()
{
  ADCON1 = 0x80;
  TRISA = 0x02;
  __delay_ms(10);
  //ADCON0 = 0x81;
}

unsigned int ADC_Read(char channel)
{
```

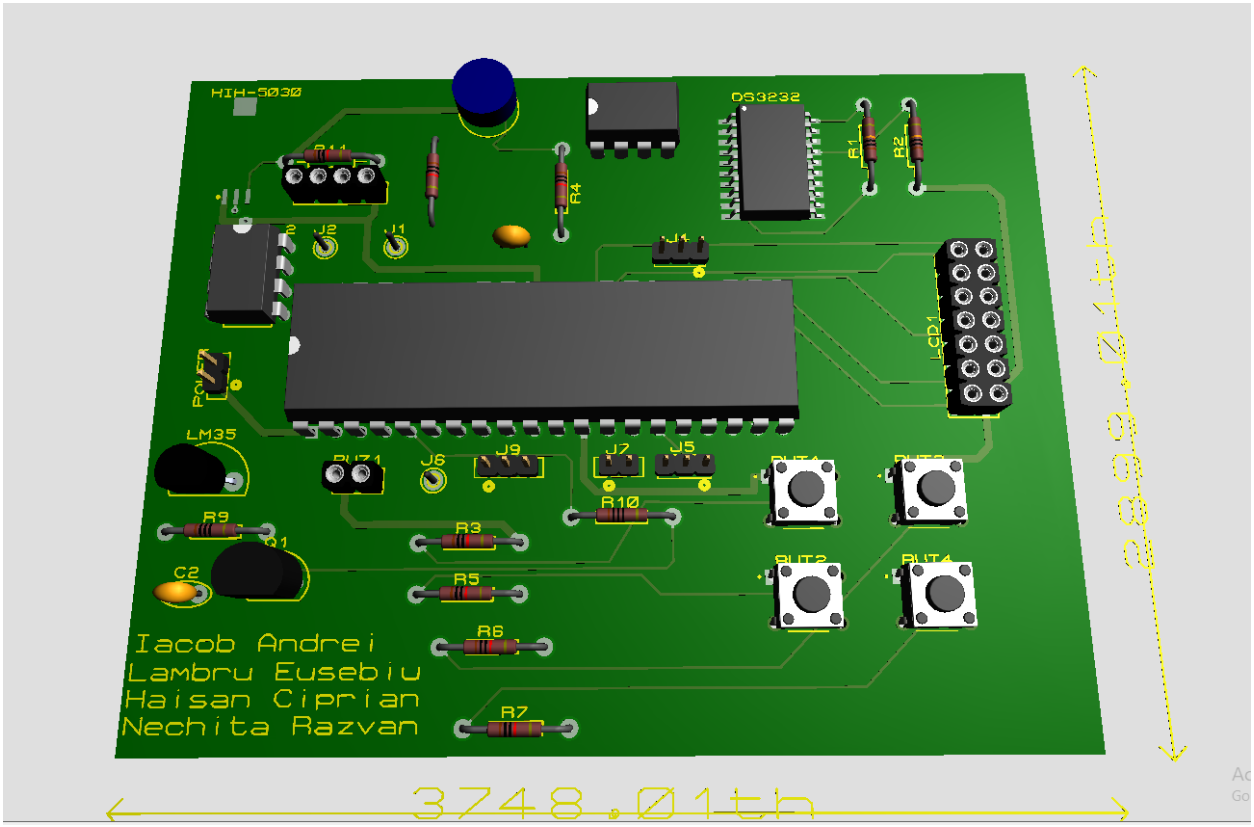
```
  __delay_ms(10);
  ADCON0 &= 0xc3;
  ADCON0 |= (channel << 2);
  ADCON0 |= 0xc5;
  __delay_ms(2);
  GO_nDONE = 1;
  while (GO_nDONE == 1)
    ;

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

unsigned int read_adc()
{
  GO_nDONE = 1;
  while (GO_nDONE)
    ;
  return ADRESH;
}

void init_uC(void)
{
  OSCCON = 0x71; // setez Osc. intern uC de 8MHz // pag. 64
  TRISA = 0b11111111;
  TRISB = 0xFF;       // tot Portul B este de iesire
  TRISD = 0b00000000; // tot Portul B este de iesire
  ANSEL = 0x02;
  ANSELH = 0x00;
  PORTB = 0b00000000;      // initializez PORTB cu valori de 0 logic
```

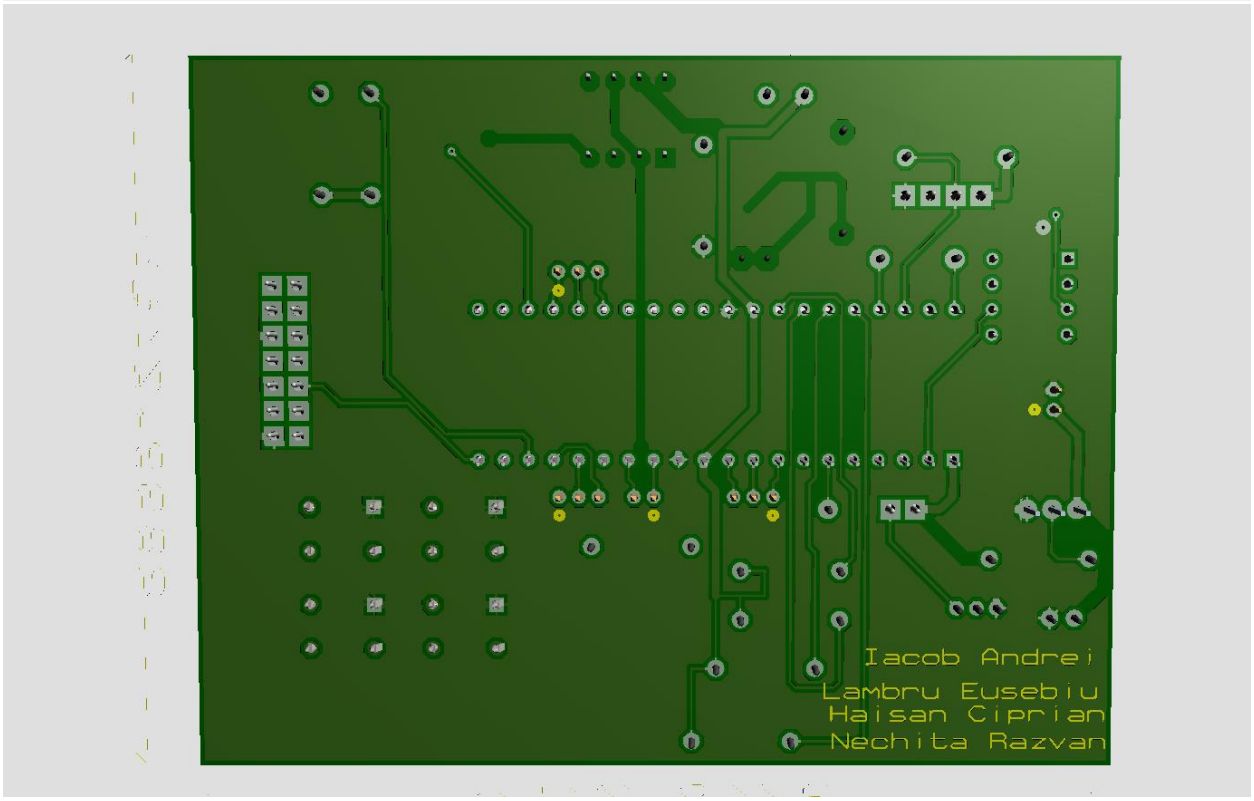
```
OPTION_REG = 0b00000111; // Frecv. intrare T0 = Frecv. Osc./4 (=8MHz/4) = 2MHz
                         // prescaler=256 => Frecv. T0 = 2MHz/256 = KHz (sau T=128us)

}
```

PCB layout



Model 3D:


HIH-5030
DS3232
LM35
R14
R4
R1
R2
R9
R3
R10
R5
R6
R7
C2
Iacob Andrei
Lambru Eusebiu
Haisan Ciprian
Nechita Razvan
2999.01th
3748.01th


Iacob Andrei
Lambru Eusebiu
Haisan Ciprian
Nechita Razvan