# Data acquisition project

Student:Pop Razvan Valentin

Coord. Professor : Patarau Toma

# *Greenhouse monitoring and automation project*

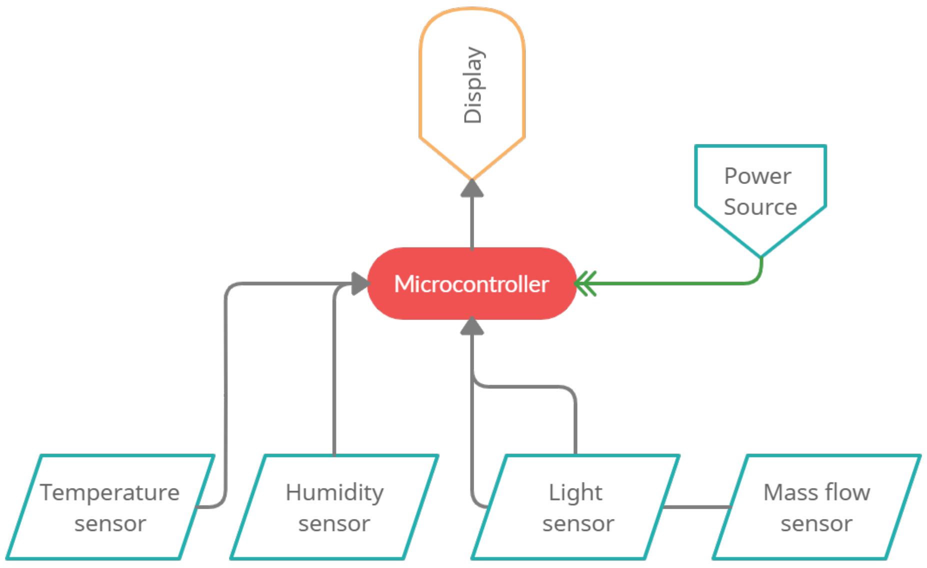
## 1.Description

This project intends to enable monitoring of temperature, humidity and other critical aspects of plant needs; and to alert and partially modify conditions given by the outside environment to suit the plants.

## 2.Generic list of components

2.1 Microcontroller
2.2 Temperature sensor
2.3 Humidity sensor
2.4 Mass flow sensor
2.5 Display
2.6 Light Sensor
2.7 Power source

## 3.Block Scheme



## 4.Temperature measurements methods

### 4.1 **RTD resistance sensors**

They make use of a change of resistance caused by temperature changes. Platinum (Pt) is the most frequently used thermoresistant material, however, we can also buy nickel (Ni) or copper (Cu) thermoresistors. Standard sensors include Pt100, Pt500 and Pt1000 devices. It means that their nominal resistance in 0ºC is 100Ω, 500Ω or 1000Ω, respectively. RTD sensors are considered to be the most accurate.

### 4.2 **Thermistor sensors**

They are a kind of thermoresistors made of sintered materials characterized by high temperature coefficient. They can be divided into two groups: NTC sensors, with negative temperature coefficient, where temperature increase causes reduction of sensor resistance, and PTC sensors, with positive temperature coefficient.

### 4.3 **Thermoelectric sensors**

They make use of **Seebeck effect**: heat is converted directly into electricity at the junction of different kind of wires made of two different kind of alloys. Value of this thermoelectric power depends on the difference in temperature of both connectors and their type.

### 4.4 **Pyrometric sensors (pyrometers)**

These are contactless temperature sensors. This is their characteristic feature; the measurement is non-invasive. In order to make a measurement, a pyrometric sensor does not have to exchange heat with an object being measured.

This way, pyrometers do not interfere with the temperature field during the measurement, and their dynamic characteristics are incomparably better. Pyrometric sensors convert thermal radiation emitted by all objects. The intensity of thermal radiation depends on temperature; it is within the visible and infrared wavelength range.

## 5.Temperature sensor method chosen

The chosen method of measuring temperature is with thermistor sensor, even if the linearity is affected on extended temperature range, that is not a problem because in a greenhouse the temperature is between ~10-30 °C.

## 6.Temperature sensors options based on chosen method

| Name | **102PS1J** | **B57861S0202H040** | **NTCLE100E3472JB0** |
|---|---|---|---|
| Photo | | | |
| Price | 11 RON | 11RON | 1.43 RON |
| Accuracy | ~5% | ~3% | ~5% |
| Type | NTC | NTC | NTC |
| Producer | Little\|fuse | TDK | VISHAY |

**The interfacing** with the microcontroller is **generally** made with a voltage divider for this type of temperature measuring sensors, because the microcontroller cannot read resistance. To be noted that the value of the resistor used in the divider must be as close as possible to the value of the thermistor.

$$V_{out} = V_{in} \times \left(\frac{R2}{R1 + R2}\right)$$

*Figure 1 Voltage divider formula*

$V_{out}:$ *Voltage between thermistor and known resistor*
$V_{in}:$ $V_{cc}$, *i.e.* $5V$
$R1:$ *Known resistor value*
$R2:$ *Resistance of thermistor*

*Figure 2 Legend for Figure 1*

$$R2 = R1 \times \left(\frac{V_{in}}{V_{out}} - 1\right)$$

*Figure 3 Formula for obtaining R2 from the voltage divider*

After that we need to use the Steinhart-Hart equation

$$\frac{1}{T} = A + BlnR + C(lnR)^3 \qquad (1)$$

And find the Steinhart-Hart coefficients to use in the previous equation

$$\begin{bmatrix} 1 & lnR1 & \ln^3 R1 \\ 1 & lnR2 & \ln^3 R2 \\ 1 & lnR3 & \ln^3 R3 \end{bmatrix} * \begin{vmatrix} A \\ B \\ C \end{vmatrix} = \begin{matrix} \frac{1}{T1} \\ \frac{1}{T2} \\ \frac{1}{T3} \end{matrix} \qquad (2)$$

With R1, R2, R3 values of resistance at the temperatures T1, T2, T3 the coefficients are expressed:

$$L_1 = \ln R_1, \quad L_2 = \ln R_2, \quad L_3 = \ln R_3$$
$$Y_1 = \frac{1}{T_1}, \quad Y_2 = \frac{1}{T_2}, \quad Y_3 = \frac{1}{T_3}$$
$$\gamma_2 = \frac{Y_2 - Y_1}{L_2 - L_1}, \quad \gamma_3 = \frac{Y_3 - Y_1}{L_3 - L_1}$$
$$\Rightarrow C = \left(\frac{\gamma_3 - \gamma_2}{L_3 - L_2}\right)(L_1 + L_2 + L_3)^{-1}$$
$$\Rightarrow B = \gamma_2 - C\left(L_1^2 + L_1 L_2 + L_2^2\right)$$
$$\Rightarrow A = Y_1 - \left(B + L_1^2 C\right) L_1$$

**7.The chosen sensor**

Given the availability, accuracy and price the optimal choice is B57861S0202H040(50mm length), main features, that influenced widely the choice are:

7.1 Short response time
7.2 Epoxy resin encapsulation
7.3 PTFE-insulated leads of silver-plated nickel wire
7.4 Relative small resistance (2KOhm)
7.5 Rated temperature 25 degrees Celsius
7.6 High range -55 to 155 degrees

Because the producer of the chosen sensor does not provide the resistance values at certain temperatures, only the B parameter and R(T)/R(25) values, we can use the B equation which essentially is a modified Steinhart-Hard equation we obtain the following:

$$\mathrm{T} = \frac{B}{\ln\left(\frac{R}{R0}\right) + \ln e^{\frac{B}{T0}}} \tag{3}$$

R0 is the resistance at temperature T0 (25°C = 298.12 K)
We have B = 3560 and R0 = 2000 OHM:

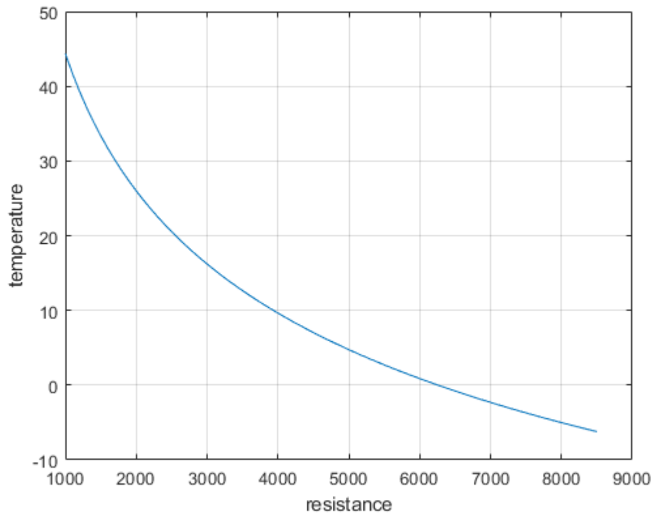


*Figure 4 Temperature computation based on beta equation*

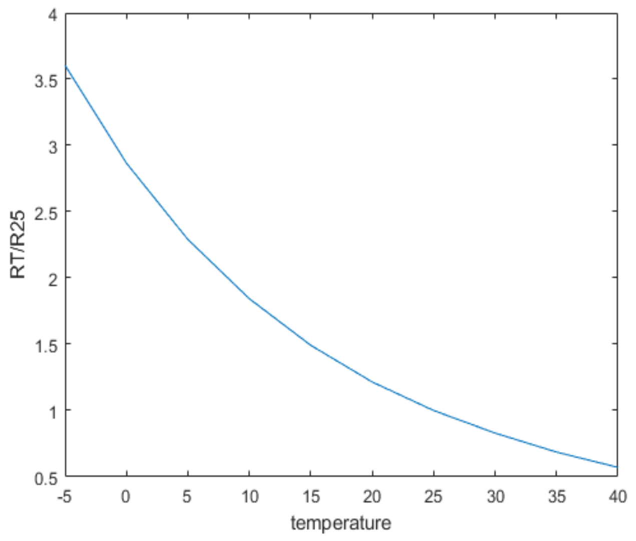


*Figure 5 R/T Characteristics*

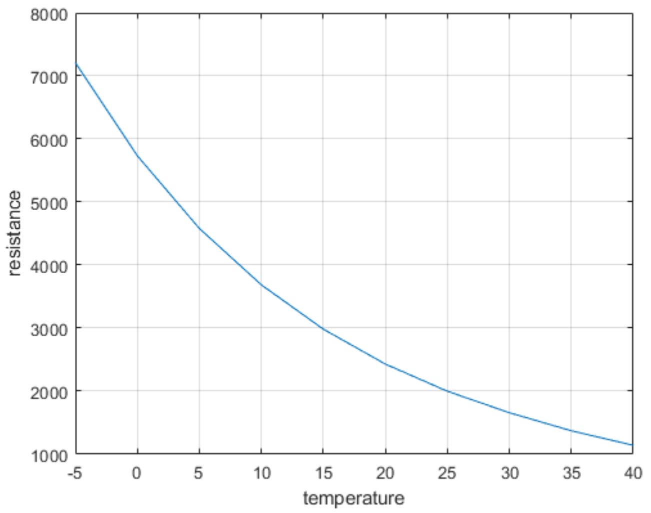


*Figure 6 Resistance computed on known temperature and RT/R25 coeff.*

# *Data acquisition project*

## 0.Specifications

The sensor must be able to determine as close as possible the humidity of the environment it is placed, to have a digital output and to be affordable.

The main working range of the sensor must be 50-70% although for some extremity cases we may need to have a range of 20-90% humidity.

## 1.Description

This project intends to enable monitoring of temperature, humidity and other critical aspects of plant needs; and to alert and partially modify conditions given by the outside environment to suit the plants.

## 2.Generic list of components

2.1 Microcontroller
2.2 Temperature sensor
2.3 Humidity sensor
2.4 Mass flow sensor
2.5 Display
2.6 Light Sensor
2.7 Power source

## 3.Block Scheme

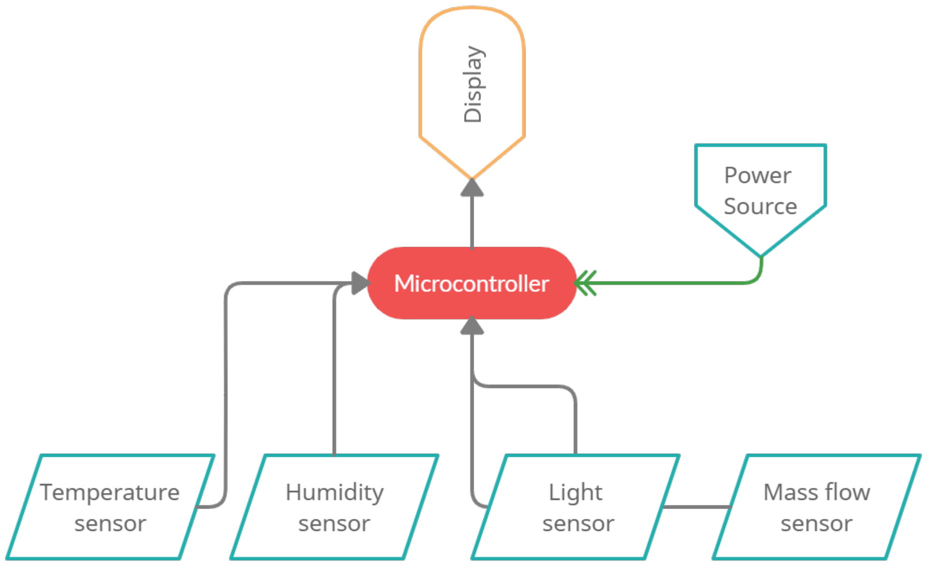


*Figure 1 Block Diagram*

## 4.Humidity measurements methods

### Capacitive Humidity Sensors

Humidity Sensors based on capacitive effect or simply Capacitive Humidity Sensors are one of the basic types of Humidity Sensors available.

They are often used in applications where factors like cost, rigidity and size are s of concern. In Capacitive Relative Humidity (RH) Sensors, the electrical permittivity of the dielectric material changes with change in humidity.

A simple Capacitive RH Sensor can be made from an air filled capacitor as the moisture in the atmosphere changes its permittivity. But for practical applications, air as a dielectric is not feasible.

### Resistive Humidity Sensors (Electrical Conductivity Sensors)

Resistive Humidity Sensors are another important type of Humidity Sensors that measure the resistance (impedance) or electrical conductivity. The principle behind resistive humidity sensors is the fact that the conductivity in non – metallic conductors is dependent on their water content.

The electrodes are placed in interdigitized pattern to increase the contact area. The resistivity between the electrodes changes when the top layer absorbs water and this change can be measured with the help of a simple electric circuit.

### Thermal Conductivity Humidity Sensors

Thermal Conductivity Humidity Sensors are also known as Absolute Humidity (AH) Sensors as they measure the Absolute Humidity. Thermal Conductivity Humidity Sensors measure the thermal conductivity of both dry air as well as air with water vapor. The difference between the individual thermal conductivities can be related to absolute humidity.

#### Working of Thermal Conductivity Humidity Sensors

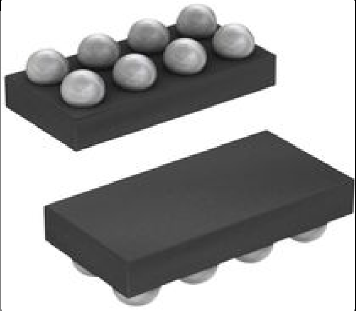
The best component to accomplish thermal conductivity based humidity sensor is thermistor. Hence, two tiny thermistors with negative temperature coefficient are used to for a bridge circuit.

In that, one thermistor is hermetically sealed in a chamber filled with dry Nitrogen while the other is exposed to open environment through small venting holes. When the circuit is powered on, the resistance of the two thermistors are calculated and the difference between those two values is directly proportional to Absolute Humidity (AH).

## 5.Temperature sensor method chosen

Based on the research made, I chose to stick with the thermal conductivity humidity sensor, because they come in small package , digital output done by the ADC and high accuracy.

## 6.Humidity sensors options based on chosen method

| Name | **SI7034-A10-IM** | **SHT21** | **HDC1010YPAT** |
|---|---|---|---|
| Photo | | |  |
| Price | 11.75 RON | 38 RON | 16 RON |
| Accuracy | ± 4% RH | ±2 | ±2 |
| Type | TCH | TCH | TCH |
| Producer | Silicon Labs | Sensirion | Texas Instruments |
| Range | 0-100% | 5-90% | 10-70% |

The interfacing : I2C interface .

**I²C** (**Inter-Integrated Circuit**), pronounced *I-squared-C*, is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial communication bus invented in 1982 by Philips Semiconductor (now NXP Semiconductors).

I²C uses only two bidirectional open collector or open drain lines, Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors. Typical voltages used are +5 V or +3.3 V, although systems with other voltages are permitted.

## 7.The chosen sensor

**SI7034-A10-IM**

Features:
-low price
-great interfacing
-high operating range (-40 -> +125 °C)
-high accuracy , low error
-small integrable package
-factory-calibrated

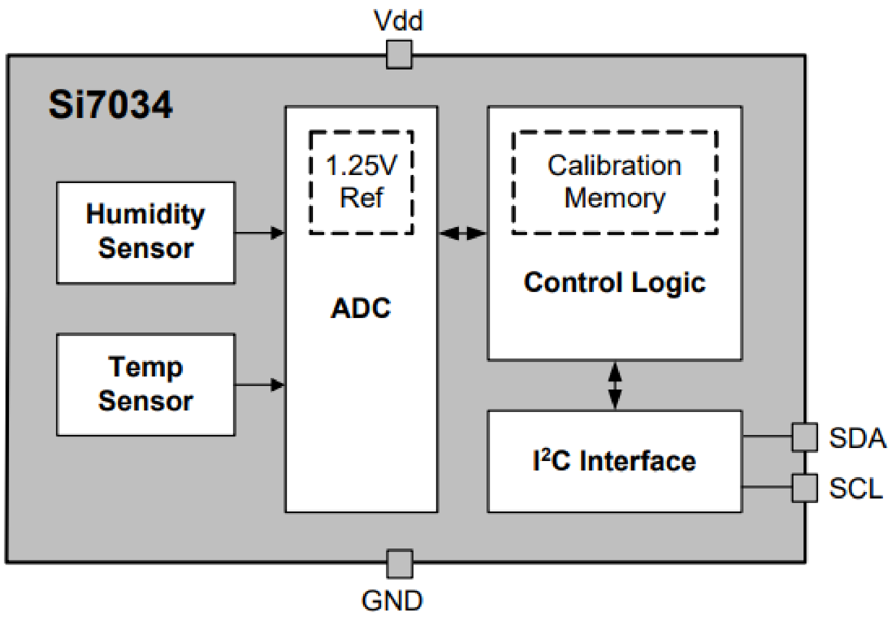


*Figure 2 Internal structural scheme*

# Data acquisition of temperature sensor converted to digital

## 1.Components and circuit

1.ADC 0808
2.OPAMP 1458
3.NTC temperature sensor
4.2x10k Resistors
5.1x2k Resistor

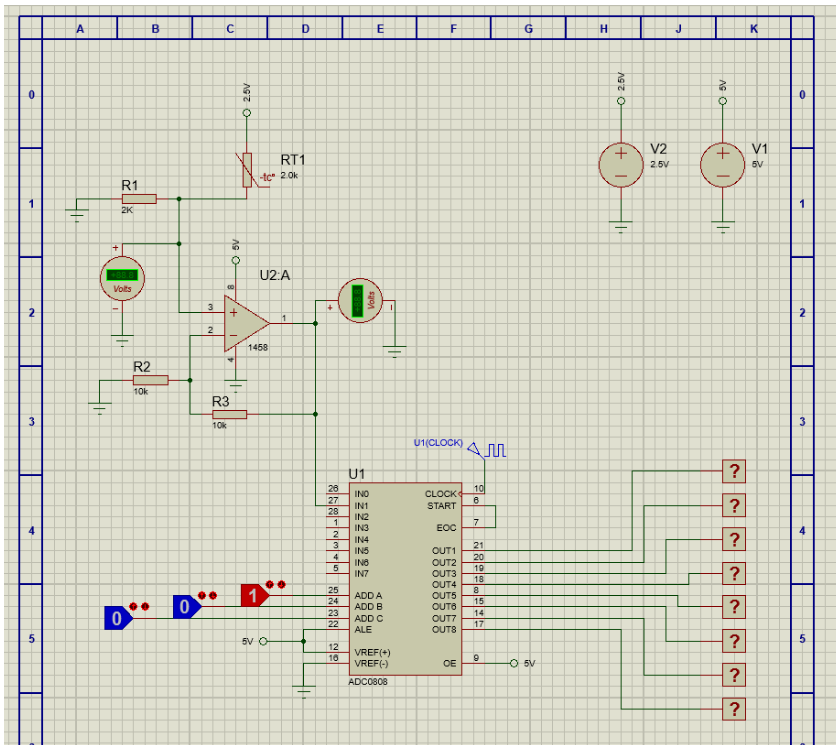


*Figure 1 The circuit without simulation*

The ADD A is set to 1 to select the input 1.

## 2.Range and computations

The optimal range is between 0 and 35 degrees Celsius.

At 0 we obtain at the output of the ADC 01000010 and a voltage of 0.64V supplied at 2.5V.
At 35 we obtain at the output of the ADC 10011000 and a voltage of 1.49 supplied at 2.5V.
At 25(T0) we obtain at the output of the ADC 10000000 and a voltage of 1.26 supplied at 2.5V.

We have a resolution of 256 because we have 8 bits.

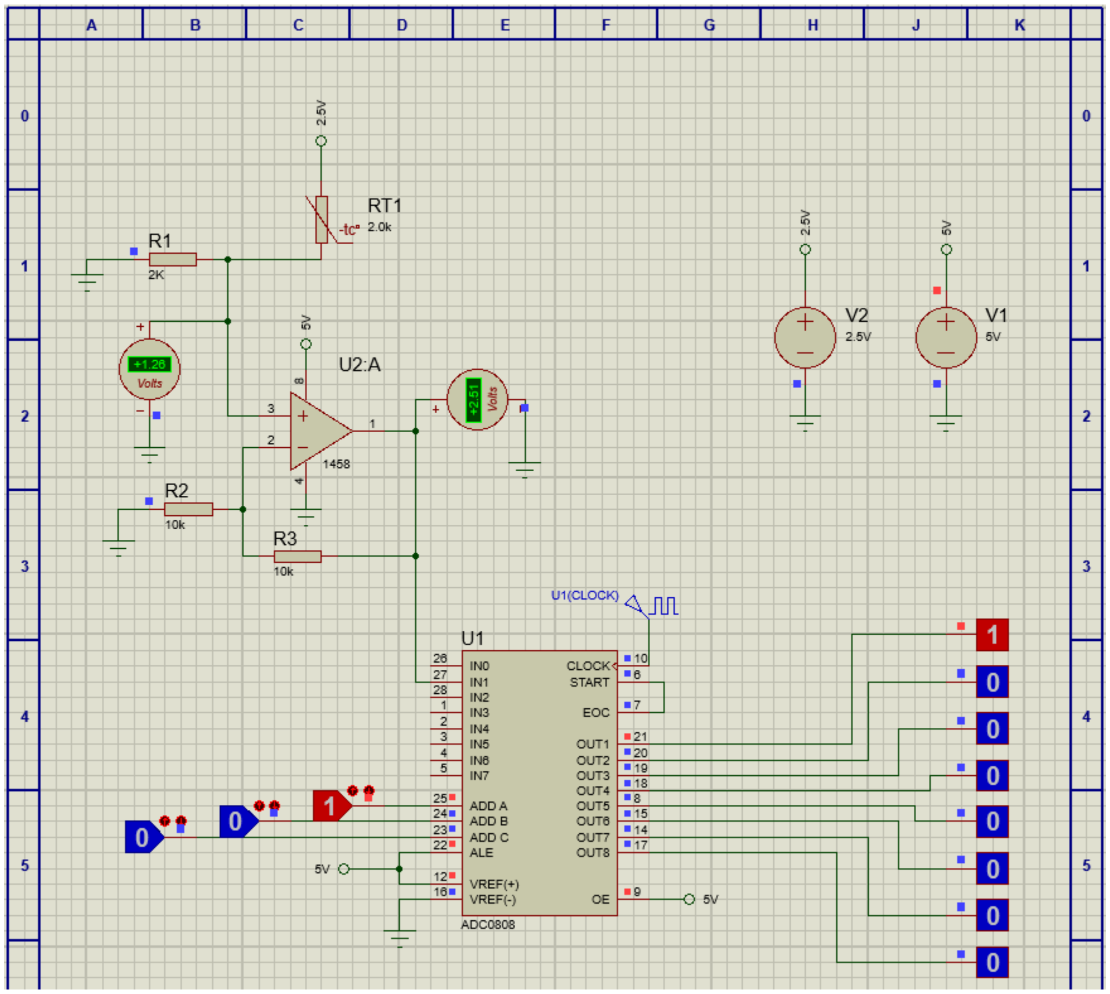


*Figure 2 Simulation at 25 (T0)*

To test the reading and the output of the ADC we compute with :

$$RT = \left(\frac{2^N}{ADC} - 1\right) * R1$$

RT(25) computed with the formula above gives : 2000 Ohm which is correct because the resistance at 25 degrees from the datasheet is exactly 2000 Ohm.

RT(35) computed with the formula above gives : 1368.42 Ohm.

RT(0) computed with the formula above gives : 5757.57 Ohm.

Which concludes the fact that our sensor works as a NTC and it varies with the temperature. Now above the fact that we computed with one formula , we will compare the results with the computations made in the first part of choosing this sensor.

$$T = \frac{B}{\ln\left(\frac{R}{R0}\right) + \ln e^{\frac{B}{T0}}}$$

Where we have the rated resistance R0(2000) at T0(25 degrees) and beta 3560 from the datasheet.

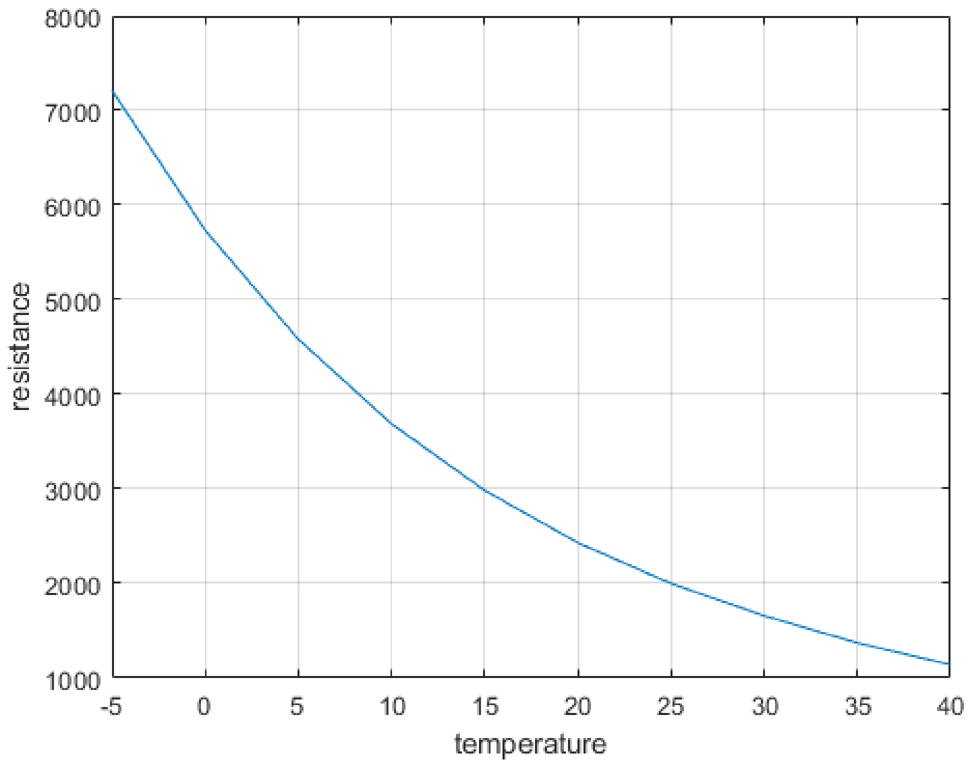


*Figure 3 Temperature versus Resistance computed in Matlab with the Steinhart - Hard equation*

```
T = 3560 ./ (log(R./2000) + log(exp(3560/298.15)));

T_C = T - 272.15;
plot(R,T_C),grid,xlabel("resistance"),ylabel("temperature")
```

**The Steinhart-Hard equation implemented in Matlab and used to generate the graph.**

Pop Razvan Valentin

# *Communication protocols and interfacing with microcontroller*

## 1.The communication protocol

A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods. Protocols may be implemented by hardware, software, or a combination of both.

## 2.The I2C Protocol

### The physical I2C bus

This is just two wires, called SCL and SDA. SCL is the clock line. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. The SCL & SDA lines are connected to all devices on the I2C bus. There needs to be a third wire which is just the ground or 0 volts. There may also be a 5volt wire is power is being distributed to the devices. Both SCL and SDA lines are "open drain" drivers. What this means is that the chip can drive its output low, but it cannot drive it high. For the line to be able to go high you must provide pull-up resistors to the 5v supply. There should be a resistor from the SCL line to the 5v line and another from the SDA line to the 5v line. You only need one set of pull-up resistors for the whole I2C bus, not for each device.

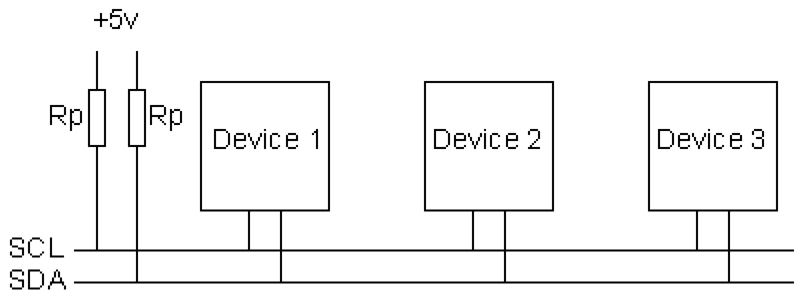


*Figure 1 pull up resistors for i2c communication*

### Masters and Slaves

The devices on the I2C bus are either masters or slaves. The master is always the device that drives the SCL clock line. The slaves are the devices that respond to the master. A slave cannot initiate a transfer over the I2C bus, only a master can do that. There can be, and usually are, multiple slaves on the I2C bus, however there is normally only one master. It is possible to have multiple masters, but it is unusual and not covered here. On your robot, the master will be your controller and the slaves will be our modules such as the SRF08 or CMPS03. Slaves will never initiate a transfer. Both master and slave can transfer data over the I2C bus, but that transfer is always controlled by the master.

### I2C Device Addressing

All I2C addresses are either 7 bits or 10 bits. This means that you can have up to 128 devices on the I2C bus, since a 7bit number can be from 0 to 127. When sending out the 7 bit address, we still always send 8 bits. The extra bit is used to inform the slave if the master is writing to it or reading from it. If the bit is zero the master is writing to the slave. If the bit is 1 the master is reading from the slave. The 7 bit address is placed in the upper 7 bits of the byte and the Read/Write (R/W) bit is in the LSB (Least Significant Bit).

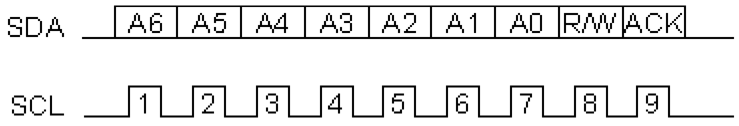
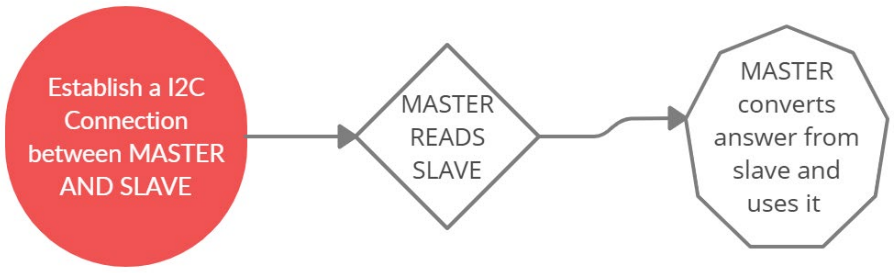


*Figure 2 bit configuration for i2c communication*

3.Organigram


Establish a I2C Connection between MASTER AND SLAVE
MASTER READS SLAVE
MASTER converts answer from slave and uses it
*Figure 3 program organigram*

4.Circuit for microcontroller

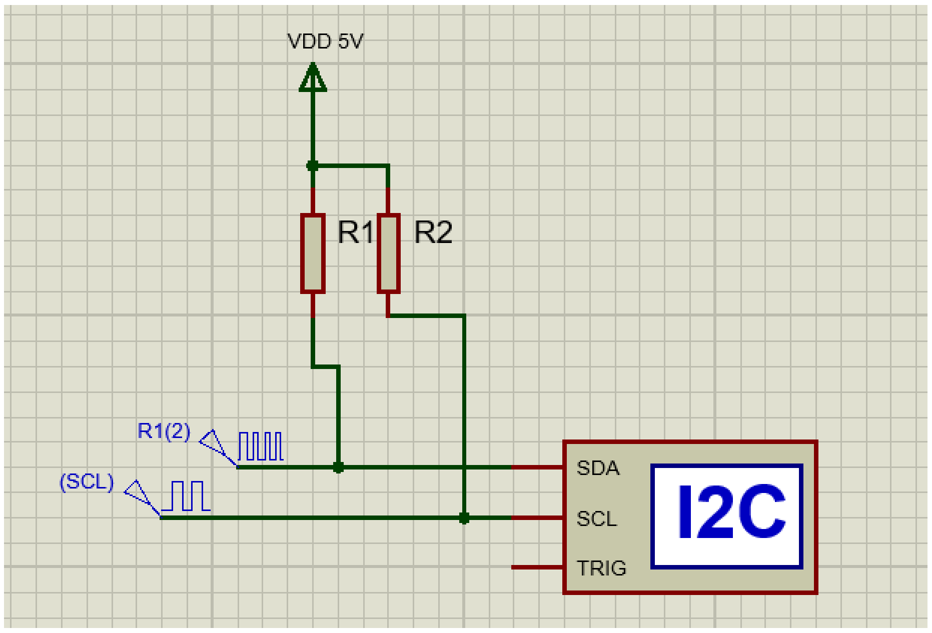


*Figure 4 proteus configuration for i2c communication*

# Microcontroller

## Description

The 8051 microcontroller is designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins DIP (dual inline package), 4kb of ROM storage and 128 bytes of RAM storage, 2 16-bit timers. It consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement. An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.

## Models

| NAME | ADE 5169 | ADE 5566 | AT89C51 |
|---|---|---|---|
| TIMERS | 3 TIMERS | 3 TIMERS | 3 TIMERS |
| INTERRUPTS | 12 Interrupt sources | 12 Interrupt sources | 12 Interrupt sources |
| COMMUNICATION | SPI/I2C | SPI/I2C | SPI/I2C |
| ROM MEMORY | 62 KBytes ROM | 62 KBytes ROM | 64 KBytes ROM |
| RAM MEMORY | 2KB RAM | 2KB RAM | 2KB RAM |
| MAXIMUM CLOCK FREQUENCY | 4MHZ | 1MHZ | 60MHZ |
| PRICE | 16RON | 23RON | 18RON |

*prices from https://www.findchips.com/

## Chosen model

AT 89C52 because has a high clock frequency , and a relative lower price in comparison with the others.

Another significant factor of the choosing process was the availability of the digital support when prototyping the whole circuit,ADE 5566/5169 are not available in Proteus.

# LCD

## Description

The desired LCD must be able to properly show the data that is required for a good system operation, and user interaction .

## Models

| Name | MC21605C6W-SPR-V2 | MC42005A12W-VNMLW | PC1602LRS-FWA-B-Q |
|---|---|---|---|
| Voltage | 5V | 5V | 5V |
| Display | STN & Reflective | VATN & Transmissive | STN & Transreflective |
| Format | 16x2 | 20x4 | 16x2 |
| Price | 34RON | 109RON | 84RON |

*all are alphanumeric

*prices from www.ro.farnell.com

## Chosen model

Due to a smaller request on the user interface the chosen model is MCR21605C6W-SPR-V2 , it is low cost and its smaller form in comparison to the 20x4 LCD is  a great advantage when encapsulation is needed.

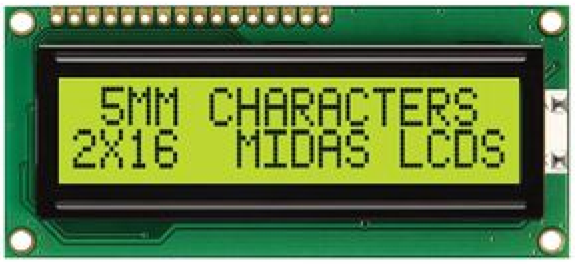


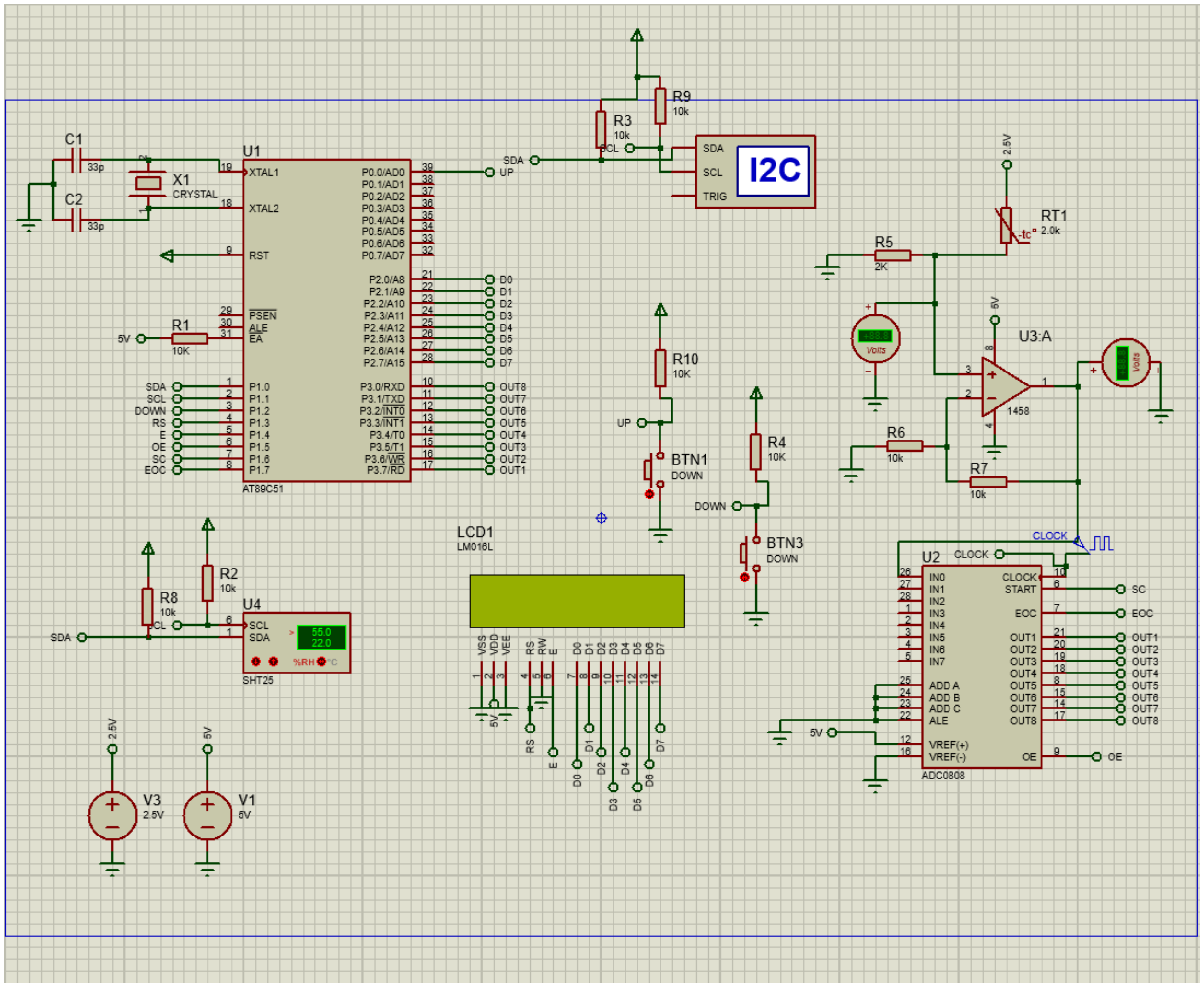
*Figure 1 MCR LCD00*

Another factor that is affecting this model choosing is the availability of it in Proteus, maybe not exactly the same model but similar models of LCD’s are available.

Pop Razvan Valentin

## Table of Contents

## Schematic



Code

## C.

```
#include<reg51.h>

#include<stdio.h>

#include<math.h>
```

```
#define lcd P2

#define inputadc P3



//globals

sbit pushdown=P1^2;

sbit pushup=P0^0;

sbit rs=P1^3;

sbit e=P1^4;

sbit oe=P1^5;

sbit sc=P1^6;

sbit eoc=P1^7;

sbit SDA=P1^0;

sbit SCL=P1^1;

float number=0;

char reading[1];//temp

char Show[1] ; //hum

float NOMINAL_RESISTANCE = 2000;

float BCOEFICIENT = 3560 ;

float humidity;

int p = 0 ;


//declarations

void delay (int);

void cmd (char);

void display (char);

void string (char *);

void init (void);
void read(void);

void convert(void);

void I2Cinit(void);



void I2CInit()

{

        SDA = 1;

        SCL = 1;

}


void I2CStart()

{

        SDA = 0;

        SCL = 0;

}


void I2CRestart()

{

        SDA = 1;

        SCL = 1;

        SDA = 0;

        SCL = 0;

}


void I2CStop()

{

        SCL = 0;

        SDA = 0;
        SCL = 1;
        SDA = 1;
}

void I2CAck()
{
        SDA = 0;
        SCL = 1;
        SCL = 0;
        SDA = 1;
}

void I2CNak()
{
        SDA = 1;
        SCL = 1;
        SCL = 0;
        SDA = 1;
}

unsigned char I2CSend(unsigned char Data)
{
         unsigned char i, ack_bit;
         for (i = 0; i < 8; i++) {
                if ((Data & 0x80) == 0)
                        SDA = 0;
                else
                        SDA = 1;
                SCL = 1;
```

```
                        SCL = 0;
                        Data<<=1;
            }
            SDA = 1;
            SCL = 1;
            ack_bit = SDA;
            SCL = 0;
            return ack_bit;
}

unsigned char I2CRead()
{
            unsigned char i, Data=0;
            for (i = 0; i < 8; i++) {
                        SCL = 1;
                        if(SDA)
                                    Data |=1;
                        if(i<7)
                                    Data<<=1;
                        SCL = 0;
            }
            return Data;
}

void delay(int n)
{
   int i=n,j;
   while(i-->0){
                                    j=100;
```

```
				while(j-->0){
					//do nothing;
				}
			}
}
void cmd (char c)
{
		lcd=c;
		rs=0;
		e=1;
		delay(5);
		e=0;
}
void display (char c)
{
		lcd=c;
		rs=1;
		e=1;
		delay(5);
		e=0;
}
void string (char *p)//pointer to the start of the string
{
		int i = 0;
		while(p[i]!=0){
			display(p[i]);
			i++;
		}
}
```

```
void init (void)

{

            cmd(0x38);

            cmd(0x0c);

            cmd(0x01);

            cmd(0x80);//position cursor on first line

}

void set_c_secondline(void){

            cmd(0x0C1);

}

void set_c_firstline(void){

            cmd(0x80);//position cursor on first line

}

void read(void)

{

  sc=1;

  delay(1);

  sc=0;

  while(eoc==1);

  while(eoc==0);

  oe=1;

  number=inputadc;

  delay(1);

  oe=0;

}

void convert(void){


            float Resistance = (256/number-1)*2000;
```

```
	float steinhart = 0 ;

	steinhart = Resistance / 2000.0;

	steinhart = log(steinhart);

	steinhart = (1/BCOEFICIENT) * steinhart ;

	steinhart = 1.0 / (25.0 + 273.15) + steinhart;

	steinhart = 1.0 / steinhart;

	steinhart = steinhart - 273.15 ;


	sprintf(reading,"%f",steinhart);

	//string(reading);

}


void read_hum(void){

	unsigned char Data[2];


	I2CStart();


	I2CSend(0x80);


	I2CSend(0xE5);


	I2CStop();


	I2CStart();


	I2CSend(0x81);


	Data[0] = I2CRead();
	I2CAck();


	Data[1] = I2CRead();


	I2CNak();


	I2CStop();


	humidity = (((Data[0]*256.0+Data[1])*125.0)/65536.0)-6;


	sprintf(Show,"%f",humidity);


}
void main()
{
	unsigned char Data[2];
	init();


	pushdown = 1;
	pushup = 1;
	read();
	convert();
	I2CInit();
	//string(Show);


	string("SYSTEM ON");


while(1){
		if(pushdown == 0){

			read();

			convert();

			read_hum();

			init();

			set_c_firstline();

			string("TEMPERATURE : ");

			set_c_secondline();

			string(reading);

		}else if(pushup==0){

			init();

			set_c_firstline();

			string("HUMIDITY : ");

			set_c_secondline();

			read_hum();

			delay(500);

			read_hum();

			string(Show);

		}


		delay(1000);


};


}
```

## ASM

```
org 0000h


tobewritten:
```

```
db 'Message:',0;message

UP equ P0.1
DOWN equ P1.2
RS equ P1.3
E equ P1.14
OE equ P0.1
START equ P0.2
SDA equ P1.0
SCL equ P1.1
EOC equ P0.0

;LCD_init()
mov a,#38h ;LCD-8bit mode
acall LCD_cmd
acall delay
mov a,#0ch ;display on, cursor off
acall LCD_cmd
acall delay
mov a,#01h ;clear
acall LCD_cmd
acall delay
mov a,#80h ;cursor at line 1
acall LCD_cmd
acall delay

mov P3,#00h
mov P2,#0FFh
```

```
setb UP

setb DOWN


setb EOC

setb OE

setb START


mov r0,#6;counter for dptr

mov dptr,#tobewritten

here1:

mov a,#00h

movc a,@a+dptr

mov p2,a

acall LCD_disp

acall delay

inc dptr

djnz r0,here1


;LCD_cmd()

LCD_cmd:

mov P2,a

setb RS ;RS=1

setb E; E=1

acall delay

clr e ;E=0

ret


;LCD_disp()

LCD_disp:
```

```
mov P2,a

setb RS ;RS=1

setb E ;E=1

acall delay

clr E

ret


;delay()

delay:

clr tr0

orl TMOD,#01

mov TH0, #0eeh

mov TL0, #00h

setb TR0

again:

jnb TF0, again

clr TF0

clr TR0

ret


end
```