Embedded Systems Final Project

Smart Swimming Pool Monitoring System

Supervisor: Dr. Belal H. Sababha

Name, Student ID, E-mail, Major:

Mohamad Halaweh, 20190059, moh20190059@std.psut.edu.jo, NIS Engineering
Mohammad Salhab, 20190663, moh20190663@std.psut.edu.jo, NIS Engineering
Arteen Topolian, 20190288, art20190288@std.psut.edu.jo, NIS Engineering

Contents

Abstract:

A collection of several sensors and actuators make up the Smart Swimming Pool Monitoring System, which collects data and measurements on a contained body of water and displays them on an LCD screen along with other information like water temperature and water depth. Water is dispensed to the water body by the system, which adjusts the water depth as a result.

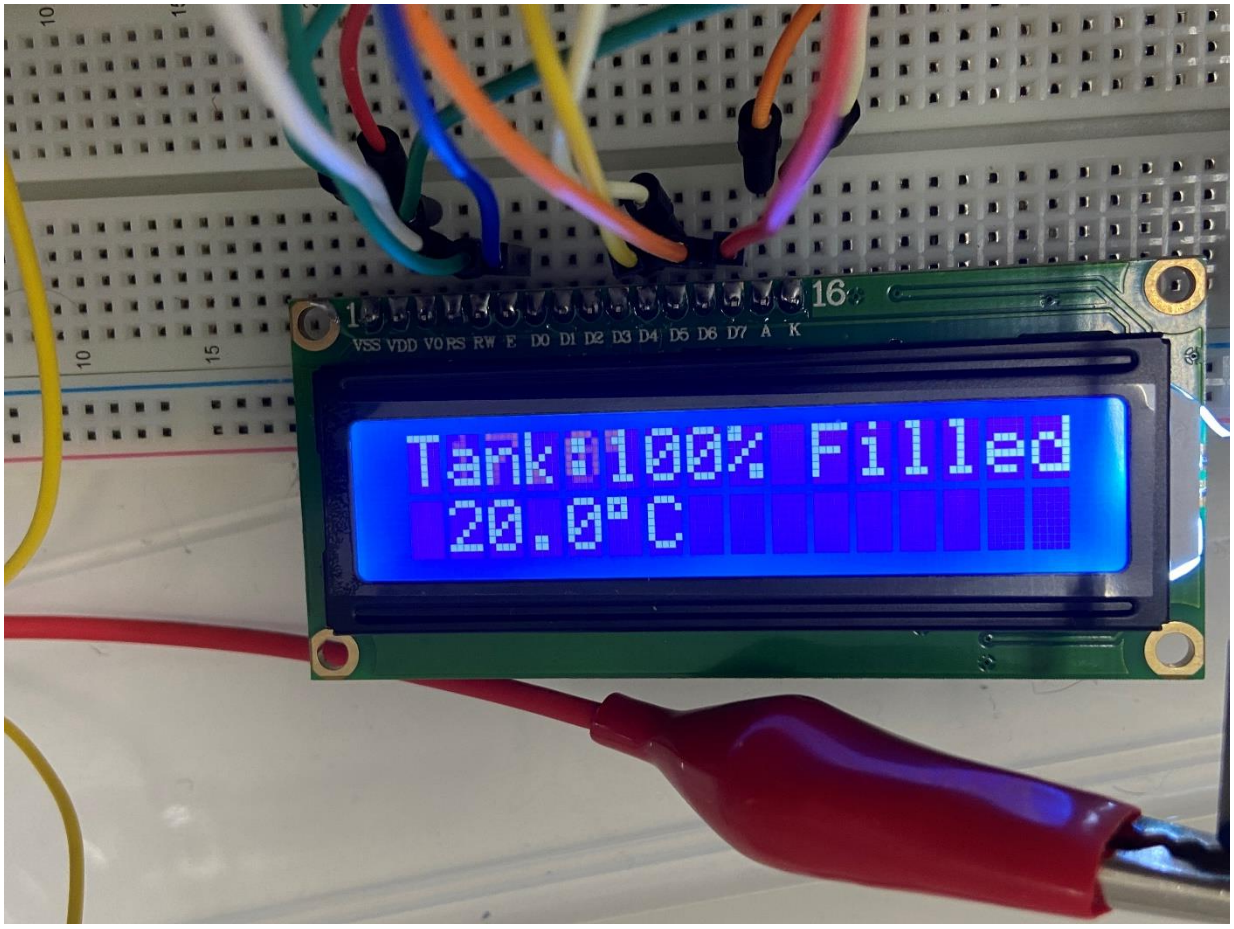
Introduction and Background:

The main idea for this project is to create a device that can measure more than one parameter for a liquid medium simultaneously, providing results to aid the user in determining the physical properties of the liquid. This device shows the distance and temperature of the liquid under test and displays it on a 16*2 LCD which helps the user to validate whether the properties of the liquid are within the wanted range.

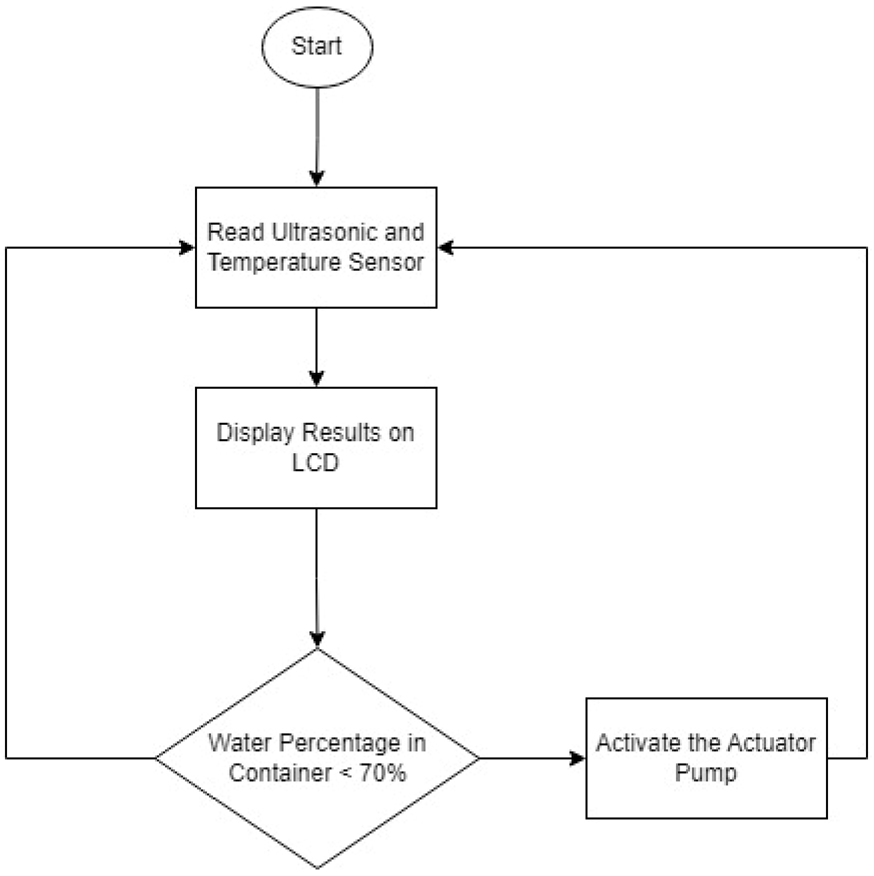
Components Used:

- PIC 16F877A
- 16*2 LCD display
- Waterproof 10KΩ NTC Thermistor Line Cable Temperature Sensor
- Ultrasonic sensor
- Bread board
- Potentiometer
- 8 MHz oscillator
- Wires
- Resistors
- Water Containers
- 220 µ Capacitors
- 470 µ Capacitors
- 10 p Capacitors

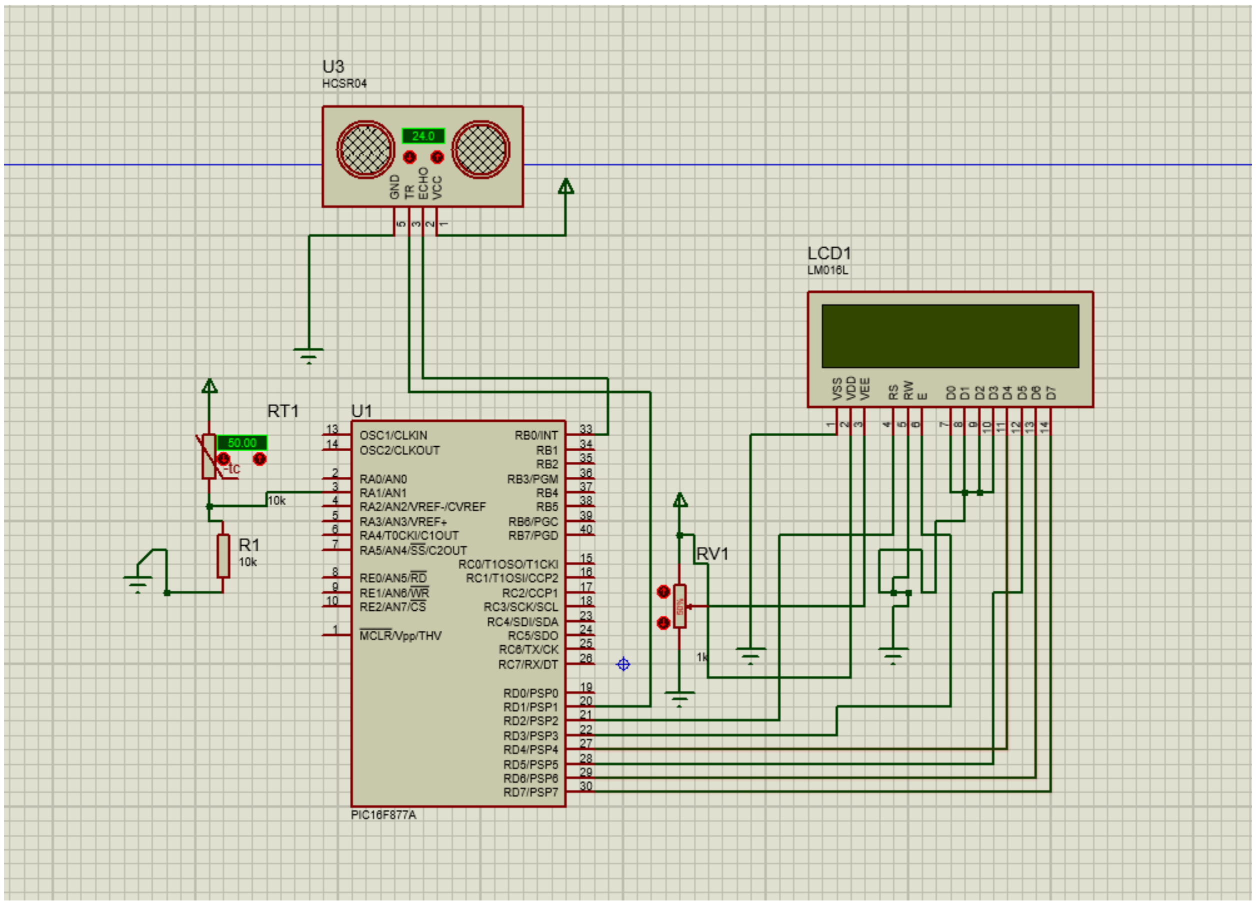
Output results on LCD:



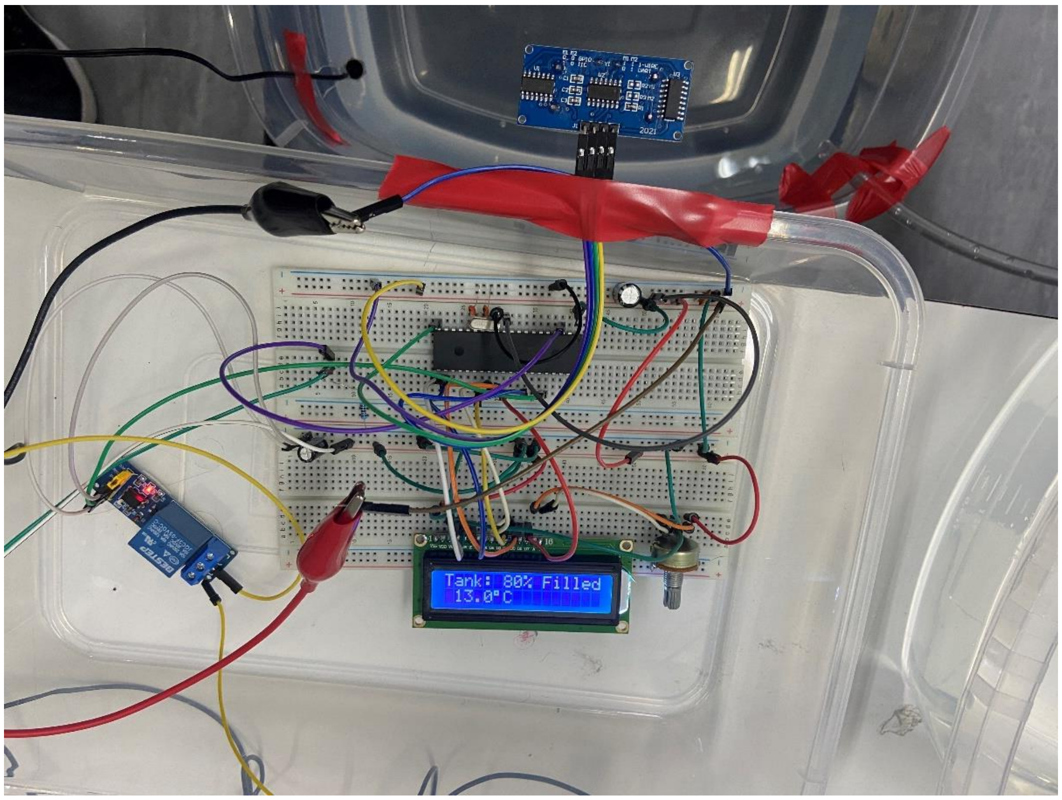
The Software Design:

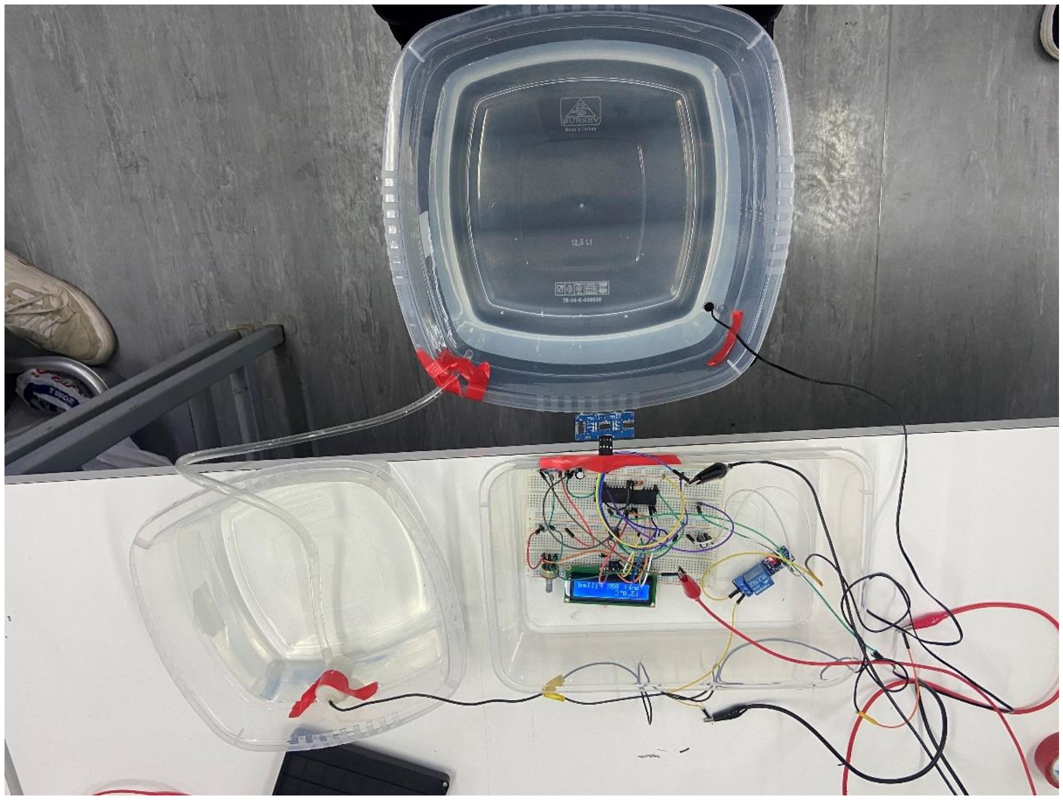

Start
Read Ultrasonic and Temperature Sensor
Display Results on LCD
Water Percentage in Container < 70%
Activate the Actuator Pump

Electrical Design:



Mechanical Design:





Problems Faced:

- During this course, we did not deal with libraries for any hardware. Therefore, we had trouble dealing with the LCD functions and displaying numbers as characters per the ASCII requirements for the LCD.

- We first bought a waterproof temperature sensor that was recommended by our parts supplier. However, after inquiring about this sensor, we figured out that it functions on the one wire implementation which requires using libraries that are not permitted to be used. Thus, we had to replace the temperature sensor with a waterproof NTC Thermistor Line Cable Temperature Sensor which allows us to practically apply knowledge that we acquired in the course such as ATD conversion.

- While calculating the temperature conversion from voltage to Celsius, the datasheet of the NTC Thermistor Line Cable Temperature Sensor stated that we implement a logarithmic function. However, after building the program with MikroC, we came across the error “Demo Limit” which we later discovered it is due to the logarithmic function we used in the equation. We worked our way around this issue by using an equation that estimated the temperature without any logarithmic function.

Recommendations & Future Plans:

In our opinion, we strongly advise students to look into how the actually works from a practical standpoint, and to look into the tools and expertise they will need to bring the project to life before submitting the proposal. In addition, we would recommend devising a Gantt chart to map out a realistic timeline for the project.

For future plans and modifications, we would add a pH sensor to measure the pH scale of the body of water, as well as adding two actuators that can modify the parameters read from the sensors related to the temperature and the pH scale.

Conclusion:

The Smart Swimming Pool Monitoring System is a device that aims to provide an easy and convenient way for users to monitor various parameters of a contained body of water, such as water temperature and depth. The system uses a combination of sensors and actuators to collect and display data on an LCD screen, and also includes a water dispensing function that allows users to adjust the water depth as needed. One of the key insights gained from this project is the importance of having accurate and reliable data when it comes to monitoring a liquid medium. By providing real-time measurements of the water temperature and depth, the system can help users ensure that the properties of the liquid are within the desired range, which is crucial for maintaining the overall health and safety of the swimming pool.

Source Code:

```
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;

void ATD_init(void);
unsigned int ATD_read(void);


 float temperature1;
  int temperature2;
     int v;

char Temperature[] = " 00.0 C";
char ds[] = " 000 cm";
unsigned int temp;
void usDelay(unsigned int);
void msDelay(unsigned int);

void main() {

  TRISB = 0xC3;
  TRISD = 0x00;
  T1CON.T1CKPS1 = 0;
 T1CON.T1CKPS0 = 0;
 T1CON.TMR1CS = 0;     //using internal clock fosc/4 & setting T1CKPS0
and T1CKPS1 to 0
  Lcd_Init();
 Lcd_Cmd(_LCD_CLEAR);
 Lcd_Cmd(_LCD_CURSOR_OFF);
 PORTB =0x08;
 msDelay(2000);
```

```
ATD_init();

  while(1){
int distance =0;
int T1=0;
TMR1L = 0;
TMR1H  = 0;
PORTD=0x02;
usDelay(10);
PORTD=0x00;

while(!PORTB.F0){}
T1CON.TMR1ON = 1;
while(PORTB.F0){}
T1CON.TMR1ON = 0;
T1 =  (TMR1H<<8) | (TMR1L) ;
distance = T1/117.64;       // calculate distance from sound speed and
time calculated using timer1


if(distance>17){            // Change Water Level Thresh
   PORTB=0x00;
}

else {
   PORTB =0x08;
}


if(distance>=30){
       lcd_Out(1, 1, "Tank:  0% Filled");
}

if(distance>=28 && distance<30){
       lcd_Out(1, 1, "Tank: 10% Filled");
}


if(distance>=26 && distance<28){
       lcd_Out(1, 1, "Tank: 20% Filled");
}
```

```
if(distance>=24 && distance<26){
       lcd_Out(1, 1, "Tank: 30% Filled");
}


if(distance>=22 && distance<24){
       lcd_Out(1, 1, "Tank: 40% Filled");
}



if(distance>=20 && distance<22){
       lcd_Out(1, 1, "Tank: 50% Filled");
}



if(distance>=18 && distance<20){
       lcd_Out(1, 1, "Tank: 60% Filled");
}



if(distance>=16 && distance<18){
       lcd_Out(1, 1, "Tank: 70% Filled");
}


if(distance>=14 && distance<16){
       lcd_Out(1, 1, "Tank: 80% Filled");
}



if(distance>=12 && distance<14){
       lcd_Out(1, 1, "Tank: 90% Filled");
}



if(distance<=10){
       lcd_Out(1, 1, "Tank:100% Filled ");
}
```

```
   msDelay(500);



   v = ATD_read();
    temp = v* 0.489;
    temp = temp / 5;
    temp = temp - 27;


    if (temp > 99)
      Temperature[0]  = 1 + 48;

    else
    Temperature[0]  = ' ';
    Temperature[1]  = (temp/ 10) % 10  + 48;
    Temperature[2]  =  temp % 10  + 48;
    Temperature[5] = 223;
    lcd_Out(2, 1, Temperature);
     msDelay(500);



  }

}

void ATD_init(void){
 ADCON0 = 0x49;  // ATD ON, Don't GO, CHannel 1, Fosc/16
 ADCON1 = 0xC0;  // All channels Analog, 500 KHz, right justified
 TRISA = 0xFF;

}
unsigned int ATD_read(void){
  ADCON0 = ADCON0 | 0x04;// GO
  while(ADCON0 & 0x04);
  return((ADRESH<<8) | ADRESL);
}

void usDelay(unsigned int usCnt){
unsigned int us=0;

for(us=0;us<usCnt;us++){
```

```
asm NOP;
asm NOP;
}
}

void msDelay(unsigned int msCnt){
unsigned int ms=0;
unsigned int cc=0;

for(ms=0;ms<msCnt;ms++){
for(cc=0;cc<155;cc++);
}
}
```