INTRODUCTION TO MICROPROCESSORS | EMBEDDED SYSTEMS DEVELOPMENT

CNG 336

MODULE 4 REPORT

Fatma Erem Aksoy – 2315075

Ece Erseven – 2385383

4.3 DESIGN AND REPORTING

4.3.1 Preliminary Work

a) Question: *The following figure is a block diagram of a successive approximation A/D converter. Explain the purpose of each part of the block diagram shown below. Do you think this A/D converter has parallel or serial digital interface? Explain.*

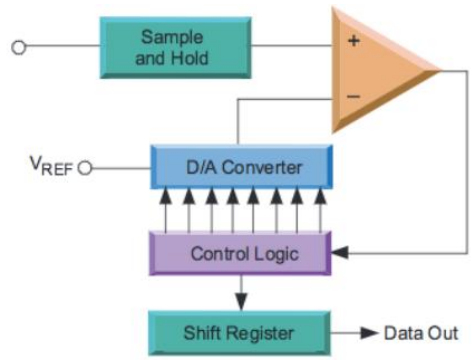


Figure 4.2. Successive Approximation ADC

Answer: This A/D converter has a serial digital interface as it uses a shift register that will out the data one bit at a time, so it's parallel-in-serial-out shift register used in a serial ADC.

Sample and Hold: It basically holds the analog signal that is changing over required amount of time and holds it so that the system (the following circuit) would have the necessary time to proceed further.

D/A Converter: It is used to convert the analog signal that is read and processed in the microcontroller, the controller unit, to digital form.

Control Logic: It takes the value generated and generate another signal based on this value. If this value (the input signal) is 0, then the new generated value will be a low signal otherwise this unit will generate a high signal and then this new generated signal will be added to the previous one.

Shift Register: It is used to send the data, which comes from the Control Logic, out one bit at a time as the data comes in bits format from the control unit.

b) Question: *If ATmega128 operates with an MCU clock frequency of 16-MHz, estimate the minimum possible single-ended A/D conversion time, showing corresponding MCU configuration requirements, and calculation. You may ignore the time it takes to initialize the analog circuitry, and may assume freerunning mode.*

Answer: MCU clock freq is given as 16-MHz, and to get the max amount of resolution, the ADC clock freq should be between 50 kHz and 200 kHz. As the ADC clock is supposed to be between those values for 10-bit accuracy, for the prescaler value, we need to use 128 prescaler and so we can get an ADC clock of 125 kHz. For the calculation part, we can show it as below:

16 MHz/128 = 125 kHz

ADCSRA configuration:

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

And the prescaler selection table to refer from the table given in lecture notes:

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

c) Question: *The following table shows the pre-amplified single-ended voltage range corresponding to the output of each sensor at the remote node, and the corresponding digital values they should be converted to. Since the system uses only 5 bits (based on the protocol described in Lab Module 2) to represent each of the sensed parameters, calculate and fill in the table with the effective resolution of the system in terms of voltage. Also, describe how you may use the existing 10-bit A/D in ATmega128 in obtaining the provided 5-bit adjusted values.*

Answer: To obtain the provided 5-bit adjusted values, we first set the ADLAR to read the data from ADCH and then shift to right three times to adjust 5-bit data. So the output will be left adjusted if the ADLAR is set, and those two registers will be holding the output of the ADC conversion.

Parameter	Min. (V)	Max. (V)	Min. (digital)	Max. (digital)	Eff. Resolution (mV)
T	2.0	4.0	0x03	0x1B	62.5 mV
M	1.8	4.2	0x02	0x1E	75 mV
W	2.0	2.8	0x04	0x1A	25 mV
B	3.0	5.0	0x01	0x01F	62.5 mV

For T:

Analog range: 2V-4V
Digital range: 0x03-0x1B
Span: 4V-2V = 2V
Resolution: 2V/2^5 = 62.5 mV

For M:

Analog range: 1.8V-4.2V
Digital range: 0x02-0x1E
Span: 4.2V-1.8V = 2.4V
Resolution: 2.4V/2^5 = 75 mV

For W:

Analog range: 2V-2.8V
Digital range: 0x04-0x1A
Span: 2.8V-2V = 0.8V
Resolution: 0.8V/2^5 = 25 mV

For B:

Analog range: 3V-5V
Digital range: 0x01-0x01F
Span: 5V-3V = 2V
Resolution: 2V/2^5 = 62.5 mV

d) Question: *Given the rotational speed of the waterpump (motor) will vary between 20% to 80%, depending on the moisture (M) level at the remote node, calculate and indicate relevant PWM generation settings you plan to program in the motor control section of your remote node solution. What will be the default motor speed before any data has been received from moisture sensor? Why?*

Answer: We need to use PWM, phase correct with non-inverting mode and the prescaler is 64 after making the necessary calculations. From the tables given in lecture notes, the TCCR0 value is 01100100 in binary and so 0x64 in hex. And as the lowest limit for the motor speed is given as 20%, we'll be using it as the lower limit, the default value, giving OCR0 value 0x33 in hex and 51 in decimal.

The calculations are as shown below:

For OCR0 (20% duty cycle):

Duty cycle: ((0.2)/255)*100 = 51 cycles

For OCR0 (80% duty cycle):

Duty cycle: ((0.8)/255)*100 = 204 cycles

For non-inverting mode (duty cycles):

MCU clock freq=8 MHz
245 Hz = 8 MHz/(510*Prescaler)
⇨ Prescaler=64

By referring to the table given in part c, the min and max moisture values are 30 and 2 in decimal. By using these values and the values we got from this section, we can find the OCR0 values as follow:

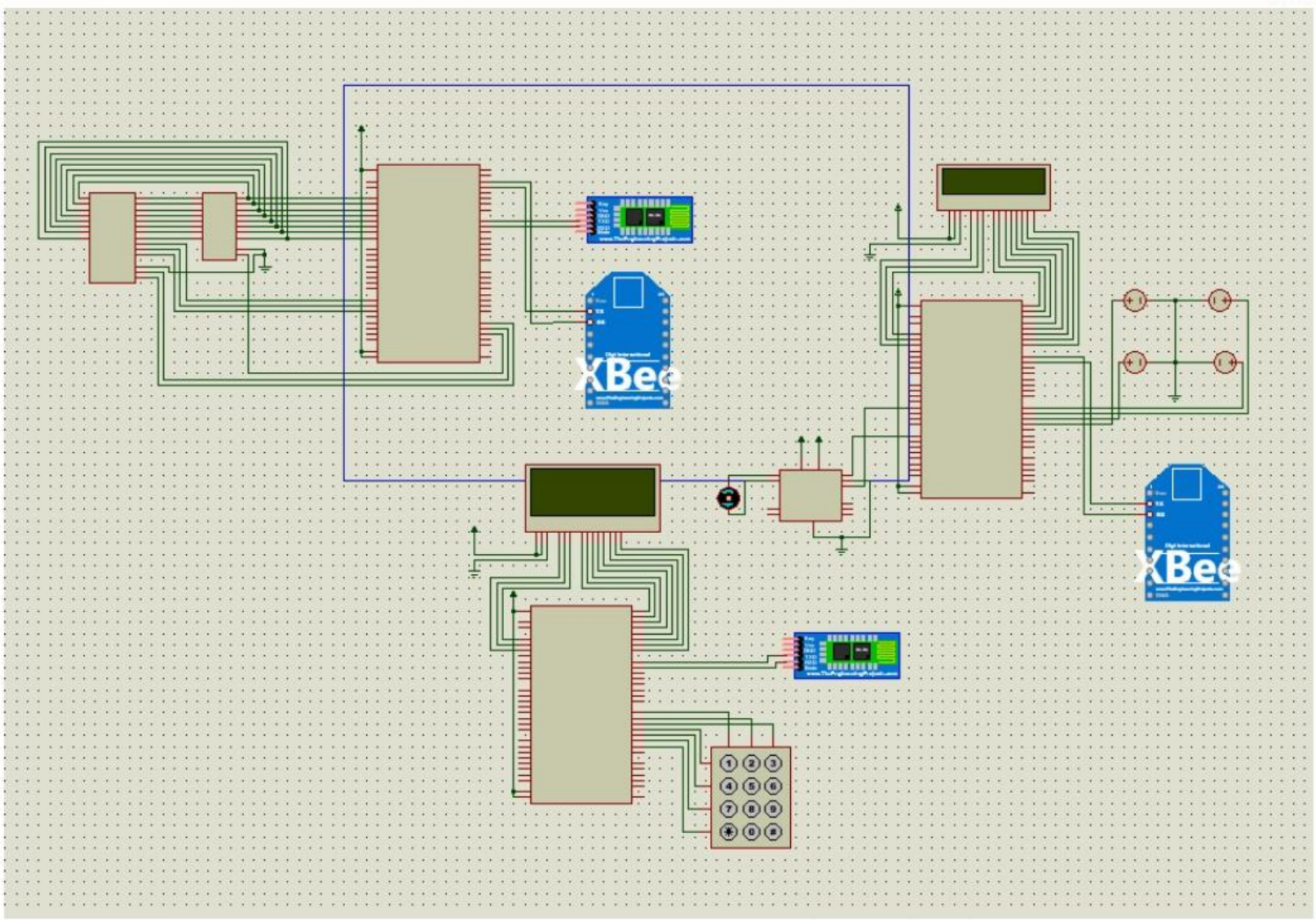
204 = 30*m + c
51 = 2*m + c
⇨ Using these two equations, we find m=4.14 and c=40.07

e) Question: *Outline the main differences between 16x2 LCD discussed in lectures, and 20x4 LCD to be utilized at the user node.*

Answer: As can be understand from their name, the size of these two LEDs are different so the character slots that they have different, more on 20x4 meaning that it has more commands, different ones than the ones in 16x2. Both have 14 pins on them and they operate on the same commands (except the extra ones in 20x4 LCD). As 20x4 is larger than 16x2, it is more efficient and useful to use it when there is loads of information to display, and so in our case, it makes more sense to use 20x4 in the user node as we'll have so much info to display.

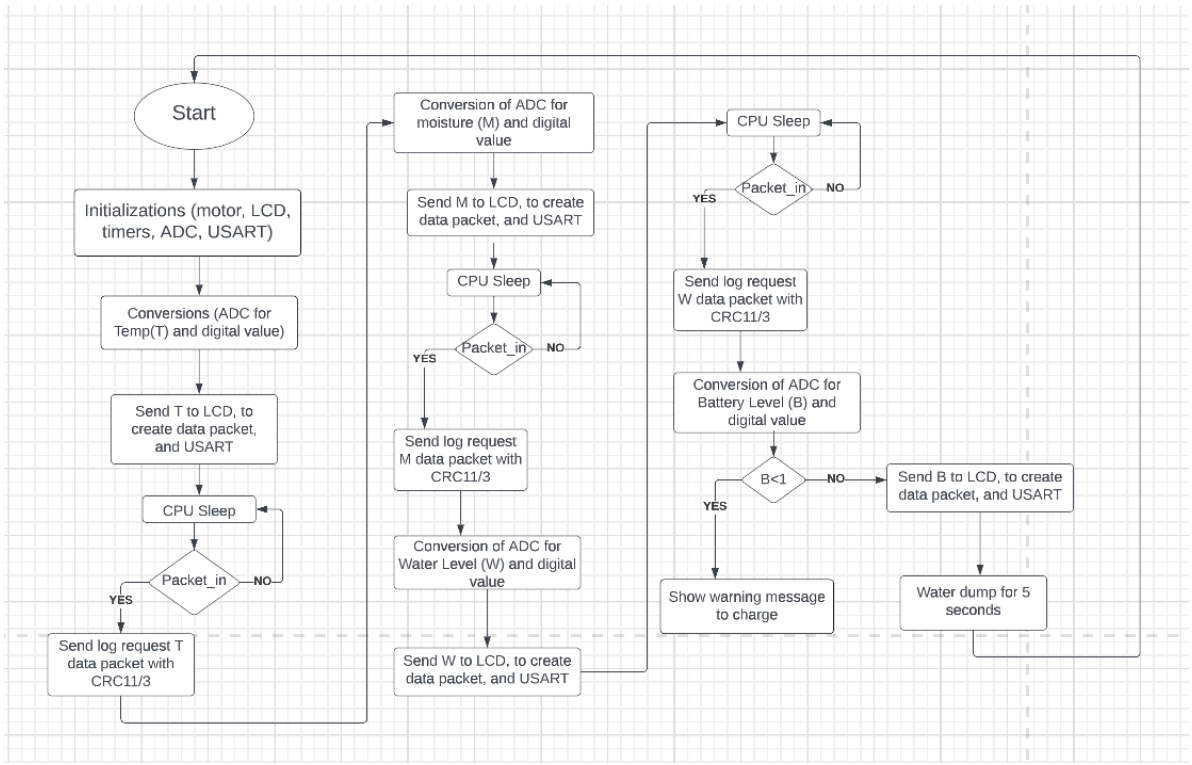
f) Question: *Sketch a system schematic diagram that has the full smart farming system, including 3 ATmega128 MCUs and their connectivity to the peripheral components. Your sketch should be organized and readable, preferably using a drawing application such as Visio. Pin level connectivity should be clear for each pin of each component.*

Answer:



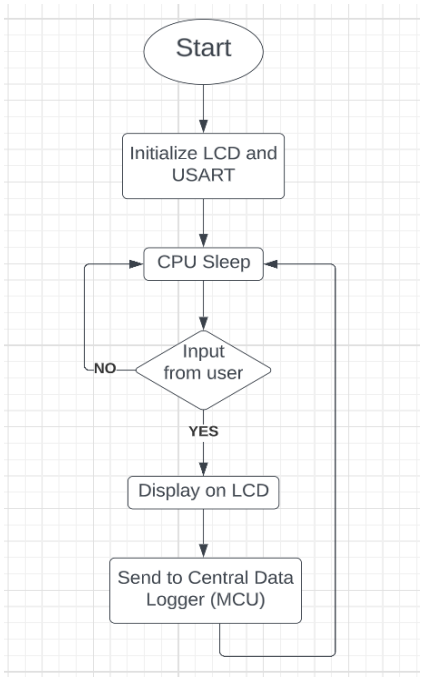
g) Question: *Sketch an algorithmic flowchart to accurately show the program executed in the Remote Sensor Node MCU.*

Answer:



h) Question: *Sketch an algorithmic flowchart to accurately show the program executed in the User Node MCU.*

Answer:



i) Question: *Considering your answers to (f-h), and sytem farming system representation in Figure 4.1, use component datasheets to investigate estimated minimum (IDLE) and maximum (ACTIVE) power dissipation for components in your system, including times when both wireless transmission interfaces are active into your worst-case power scenario. Complete the blanks in Table 4.1.*

Answer:

Component Power (mW)	Approx. best-Case (IDLE)	Approx. worst-Case (ACTIVE)
MCU (Central)	300	2000
MCU (User-node)	200	1600
MCU (Remote-node)	200	1600
Bluetooth Interface	40	100
Xbee Interface	6.3	132
16x2 LCD display	1	100
20x4 LCD display	2	100
Motor driver	0	200
Waterpump	0	1000

4.3.2 Design - Code:

For Central Data Logger:

main.c

main.c D:\Seventh Semester\CNG 336\Labs\Lab4\Module4\centralDataLogger\centralDataLogger\main.c Go

```c
/*
 * Module4.c
 *
 * Created: 1/2/2023 7:56:17 PM
 * Author : Erem
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define USER_TR_BUFFER_SIZE 128
#define SENSOR_TR_BUFFER_SIZE 5
#define Reset 0b00000000
#define LogRequest 0b00100000
#define Acknowledge 0b01000000
#define ErrorRepat 0b01100000
unsigned char user_tr_buffer[USER_TR_BUFFER_SIZE] = "";
unsigned char user_tr_index = 0;
unsigned char myList[]= "0123456789ABCDEF";
unsigned char Stack[20];
unsigned char ToS=-1;
unsigned char user_rv_buffer[2];
unsigned char user_rv_index=-1;
unsigned char sensor_rv_buffer[2];
unsigned char sensor_rv_index=-1;
unsigned char new_user_read_char;
unsigned char new_sensor_read_char;
unsigned char sensor_tr_buffer[SENSOR_TR_BUFFER_SIZE] = " ";
unsigned char sensor_tr_index = 0;
unsigned char new_sensor_read_char;
unsigned char cbit;
unsigned char *memadd=0x0500;
unsigned char *memPTR=0x0500;
unsigned char f_temp;
unsigned char s_temp;
unsigned char t_temp;
```

```c
unsigned char fourth_temp;
unsigned char c_ounter;
unsigned char flag;
unsigned char g_poly= 0b00110101;
unsigned char g_poly2= 0b11010100;
unsigned char try;


void usart_init(void)
{
    UCSR0B = (1<<RXEN0)|(1<<RXCIE0) | (1<<TXEN0) | (1<<TXCIE0);
    UCSR0C = (1<<UCSZ01) | (1<<UCSZ00); // 8 bit data 1 stop bit
    UBRR0L = 0x33; // XTAL = 8 MHz
    UBRR0H=0x0;
    UCSR1B = ((1<<RXEN1)|(1<<RXCIE1) | (1<<TXEN1) | (1<<TXCIE1));
    UCSR1C = (1<<UCSZ01) | (1<<UCSZ00) ;
    UBRR1L = 0x33;
    UBRR1H=0x0;
}

unsigned char crc3(unsigned char temp){
    g_poly= 0b00110101;
    f_temp=0;
    if (temp & (1<<6))
    {
        g_poly= g_poly<< 1;
        f_temp= temp;
        f_temp= f_temp^ g_poly;
        if (f_temp & (1<<5)){
            g_poly= g_poly>> 1;
            f_temp= f_temp^ g_poly;
            temp = temp + f_temp;
            return temp;
        }
        else{
            temp = temp + f_temp;
            return temp;
        }
    }

    else{
        f_temp= temp;
        if (f_temp & (1<<5))
        {
            f_temp= f_temp^ g_poly;
            temp = temp + f_temp;
            return temp;
        }
        else{
            temp = temp + f_temp;
            return temp;
        }
    }

    return temp;
}

unsigned char crc3_check (unsigned char temp){
    f_temp= temp;
    f_temp= crc3(f_temp);
    s_temp=temp;
    s_temp= s_temp& 0b00011111;
    f_temp= f_temp& 0b00011111;
    if(f_temp==s_temp){
        return 1;
    }
    else{
        return 0;
    }
}

unsigned char crc11_check(unsigned char temp,unsigned char f_temp){
    g_poly2= 0b11010100;
    t_temp= f_temp;
    fourth_temp= temp;
```

```
    fourth_temp= fourth_temp& 0b11100000;
    int c_ounter;
    for(c_ounter=11;c_ounter>0;c_ounter--)
    {
        if (t_temp & (1<<7))
        {
            t_temp= t_temp^ g_poly2;
        }
        cbit= fourth_temp& (1<<7);
        if (cbit==0b10000000)
        {
            cbit=1;
        }
        else{
            cbit=0;
        }
        fourth_temp= fourth_temp<< 1; //
        t_temp= t_temp<<1;
        t_temp= t_temp+ cbit;
        cbit=0;
    }
    t_temp= t_temp& 0b11111000;
    cbit=0;
    t_temp= t_temp>> 3;
    flag = 0;
    s_temp= temp;
    s_temp= s_temp& 0b00011111;
    if(s_temp== t_temp){
        return 1;
    }
    else{
        return 0;
    }
    return t_temp;
}
```

```
//wait until came interrupt
void Sleep_Wait (){
    sleep_enable();
    sei();
    sleep_cpu();
    sleep_disable();
}

void user_buf_tr_init (){
    unsigned char i;
    for(i=0;i<128;i++){
        user_tr_buffer[i]='\0';
    }
    user_tr_index =0;
}

void sen_buf_tr_init (){
    unsigned char i;
    for(i=0;i<5;i++){
        sensor_tr_buffer[i]='\0';
    }
    sensor_tr_index =0;
}

void user_buf_out (unsigned char data){
    while(!(UCSR0A & (1<<UDRE0))); //;
    UDR0=data; //send data
}

void sensor_buf_out (unsigned char data){
    while(!(UCSR1A & (1<<UDRE1)));
    UDR1=data;//send data
}

void init_sensor_rem(void) {
    push(crc3(Reset));
    Sen_Message (Stack[ToS]);
    strcpy(user_tr_buffer, "Sensor initialize\r");
```

```
    us_message();
}

void push(unsigned char x){
    ToS++;
    Stack[ToS] = x;
}

unsigned char pop(){
    unsigned char x;
    x=Stack[ToS];
    Stack[ToS]='/0';
    ToS--;
    return x;
}

void Sen_Message (){
    unsigned char i = strlen(sensor_tr_buffer);
    unsigned char j=0;
    while (j<i)
    {
        sensor_buf_out (sensor_tr_buffer[j]);
        j++;
    }
    sen_buf_tr_init ();
}

void us_message(){
    unsigned char i = strlen(user_tr_buffer);
    unsigned char j=0;
    while (j<i)
    {
        user_buf_out (user_tr_buffer[j]);
        j++;
    }
    user_buf_tr_init ();
}
void Req_to_Repeat (void){
    sen_buf_tr_init ();
    ascii_sensor (crc3(ErrorRepat));
    Sen_Message ();
    user_buf_tr_init ();
    strcpy(user_tr_buffer, "Error Repeat go to sensor\r");
    us_message();
}

void DataType(void){
    if(ToS>=0){ //Stack empty
        pop();
    }
    push(new_sensor_read_char);
    user_buf_tr_init ();
    strcpy(user_tr_buffer, "Data packet goes to stack\r");
    us_message();
}

void CommandType(void){
    unsigned char checkFlag =0;
    unsigned char checkIf=0;
    if ((Stack[ToS] & (1<<7)))
    {
        checkFlag= crc11_check(new_sensor_read_char,Stack[ToS]);
        try= new_sensor_read_char;
        if (checkFlag==0)
        {
            pop();
            Req_to_Repeat ();
        }
        else{ // if crc11 pass
            checkIf = new_sensor_read_char & 0b11100000;
            if(checkIf=0b00100000){
                if (memadd==0x18FF)
                {
                    memadd =0x0500;
                }
```

```
                if (memadd==0x10EB)
                {
                    memadd=0x1100;
                }
                memadd++;
                *memadd = Stack[ToS];
                push( crc3(Acknowledge));
                sen_buf_tr_init ();
                ascii_sensor (Stack[ToS]);
                Sen_Message ();
                //User information
                user_buf_tr_init ();
                strcpy(user_tr_buffer, "Stack content has gonesensor(Acknowledge)\r");
                us_message();
            }
        }
    }
    else{
        checkFlag = crc3_check(new_sensor_read_char);
        if (checkFlag==1)
        {
            checkIf = new_sensor_read_char & 0b11100000;
            if (checkIf==0b01000000)
            {
                user_buf_tr_init ();
                strcpy(user_tr_buffer, "Acknowledge is packetin\r");
                us_message();
                if (ToS>=0)
                {
                    pop();
                }
            }
            else{
                if (checkIf=0b01100000)
                {
                    user_buf_tr_init ();
                    strcpy(user_tr_buffer, "Repeat/Error is packet in\r");
                    us_message();
                    if (ToS>=0)
                    {
                        sen_buf_tr_init ();
                        ascii_sensor (Stack[ToS]);
                        Sen_Message ();
                        //User informations
                        user_buf_tr_init ();
                        strcpy(user_tr_buffer, "Stack content has gone sensor\r");
                        us_message();
                    }
                }
            }
        }
        else{
            Req_to_Repeat ();
        }
    }
}

void MemoryDump(){
    unsigned char x;
    unsigned char temp5;
    memPTR=0x500;
    while(memPTR!=0x10E0){
        user_buf_tr_init ();
        x= *memPTR;
        UsermakeASCII(x);
        us_message();
        memPTR++;
    }
    memPTR=0x1100;
    while(memPTR!=0x18FF){
        user_buf_tr_init ();
        x= *memPTR;
        UsermakeASCII(x);
        us_message();
        memPTR++;
    }
}
```

```c
void UsermakeASCII(unsigned char x){
    unsigned char temp5;
    temp5 = x >> 4;
    temp5 = temp5 & 0b00001111;
    user_tr_buffer[user_tr_index]=myList[temp5];
    user_tr_index++;
    x = x & 0b00001111;
    user_tr_buffer[user_tr_index]=myList[x];
    user_tr_index++;
    user_tr_buffer[user_tr_index]='\r'; //new line
}

void ascii_sensor (unsigned char x){
    unsigned char temp5;
    temp5 = x >> 4;
    temp5 = temp5 & 0b00001111;
    sensor_tr_buffer[sensor_tr_index]=myList[temp5];
    sensor_tr_index++;
    x = x & 0b00001111; //lest significant byte come
    sensor_tr_buffer[sensor_tr_index]=myList[x];
    sensor_tr_index++;
    sensor_tr_buffer[sensor_tr_index]='\r';
}

void lastEntry(){
    user_buf_tr_init ();
    strcpy(user_tr_buffer,"\r");
    us_message();
    unsigned char x;
    unsigned char t,v;
    x= *memadd;
    user_buf_tr_init ();
    UsermakeASCII(x);
    us_message();
}

//It finds the ASCII value of the data
unsigned char findValue(unsigned char asciiValue){
    unsigned char i=0;
    while (i<strlen(myList))
    {
        if (myList[i]==asciiValue)
        return i;
        i++;
    }
}

ISR(USART0_TX_vect){
    if (user_tr_index == 0){
        user_buf_tr_init ();
        UCSR0B &= ~((1 << TXEN0) | (1 << TXCIE0));
    }
}

ISR(USART0_RX_vect) {
    while (!(UCSR0A & (1<<RXC0) )){};
    new_user_read_char = UDR0;
    user_rv_index++;
    user_rv_buffer[user_rv_index]= new_user_read_char;
}

ISR(USART1_TX_vect){
    if (sensor_tr_index == 0){
        sen_buf_tr_init ();
        UCSR1B &= ~((1 << TXEN1) | (1 << TXCIE1));
    }
}

ISR(USART1_RX_vect) {
    while (!(UCSR1A & (1<<RXC1) )){};
    new_sensor_read_char = UDR1;
    sensor_rv_index ++;
    sensor_rv_buffer[sensor_rv_index]=new_sensor_read_char;
```

```
    if (sensor_rv_index==1)
    {
        sen_buf_tr_init ();
        strcpy(sensor_tr_buffer,"\r");
        Sen_Message ();
        sen_buf_tr_init ();
        f_temp=findValue(sensor_rv_buffer[0]);
        s_temp= findValue(sensor_rv_buffer[1]);
        f_temp= f_temp<< 4;
        new_sensor_read_char = f_temp+ s_temp;
        f_temp=0;
        s_temp=0;
        f_temp= new_sensor_read_char;
        if (new_sensor_read_char & (1<<7))
        {
            DataType();
        }
        else {
            CommandType();
        }
        sensor_rv_index=-1;
        sen_buf_tr_init ();
        strcpy(sensor_tr_buffer,"\r");
        us_message();
    }
}

void ConfigSlaveWD(){
    if (user_rv_buffer[1]=='A')
    {
        WDTCR= (1<<WDE) | (1<<WDP0); //26.Ms
        user_buf_tr_init ();
        strcpy(user_tr_buffer, "\rWatchdog -30ms\0");
        us_message();
    }
    else if (user_rv_buffer[0]=='B')
    {
        WDTCR= (1<<WDE) | (1<<WDP2);
        user_buf_tr_init ();
        strcpy(user_tr_buffer, "\rWatchdog -30ms\0");
        us_message();
    }
    else if (user_rv_buffer[0]=='B')
    {
        WDTCR= (1<<WDE) | (1<<WDP2);
        user_buf_tr_init ();
        strcpy(user_tr_buffer, "\rWatchdog -250ms\0");
        us_message();
    }
    else if (user_rv_buffer[0]=='C')
    {
        WDTCR= (1<<WDE) | (1<<WDP2) | (1<<WDP0);
        user_buf_tr_init ();
        strcpy(user_tr_buffer, "\rWatchdog -500ms\0");
        us_message();
    }
}

int main(void)
{
    unsigned char flag=0;
    MCUCR = 0x80;
    XMCRB =(1<<XMM1)|(1<<XMM0);
    ToS=-1; //Stack initialization
    usart_init();
    init_sensor_rem();
    sei();
    while(1){
        usart_init();
        sei();
        Sleep_Wait ();
        if (user_rv_index==1)
        {
            if (user_rv_buffer[1]=='.')
            {
                if (user_rv_buffer[0]=='1')
                {
                    MemoryDump();
```

```
                    user_buf_tr_init ();
                    strcpy(user_tr_buffer,"\r");
                    us_message();
                }
                else if (user_rv_buffer[0]=='2')
                {
                    lastEntry();
                    user_buf_tr_init ();
                    strcpy(user_tr_buffer,"\r");
                    us_message();
                }
                else{
                    user_buf_tr_init ();
                    sen_buf_tr_init ();
                    init_sensor_rem();
                }
                user_rv_index=-1;
            }
            else{
                user_rv_index=-1;
            }
        }
        else{
            if (user_rv_buffer[0]=='A' | user_rv_buffer[0]=='B' |
            user_rv_buffer[0]=='C')
            {
                ConfigSlaveWD();
                user_rv_index=-1;
            }
        }
    }
}
```

For the Remote Sensor Mode:

main.c

D:\Seventh Semester\CNG 336\Labs\Lab4\Module4\remoteSensorNode\remoteSensorNode\main.c

```c
/*
 * remoteSensorNode.c
 *
 * Created: 1/2/2023 7:58:41 PM
 *  Author: Erem
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/sleep.h>
#define F_CPU 8000000UL
#define BAUD_RATE 9600
#define Reset 0b00000000
#define LogRequest 0b00100000
#define Acknowledge 0b01000000
#define Error 0b01100000
#define LCD_DPRT PORTD //LCD DATA
#define LCD_DDDR DDRD
#define LCD_DPIN PIND
#define LCD_CPRT PORTA //LCD COMMAND
#define LCD_CDDR DDRA
#define LCD_CPIN PINA
#define LCD_EN 0 //LCD EN
#define LCD_RW 1 //LCD RW
#define LCD_RS 2 //LCD RS
unsigned char T;//temperature
unsigned char M;//moisture
unsigned char WL;// water level
unsigned char BL;
unsigned char sn=0;
unsigned char usartrecieve=0x00;
int motor_open_close= 0;
unsigned char f_crctemp;
unsigned char s_crctemp;
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'remoteSensorNode' (1 projec

remoteSensorNode

Dependencies

Output Files

Libraries

main.c

VA View VA Outline Solution Explorer

Properties

```c
void usart_init()
{
    UCSR0B = (1<<RXEN0)|(1<<RXCIE0) | (1<<TXEN0) | (1<<TXCIE0);
    UCSR0C = (1<<UCSZ01) | (1<<UCSZ00); // 8 bit data 1 stop bit
    UBRR0H = 0x33;
    UBRR0L = 0x0;
}

void sleeptime(){
    sleep_enable();
    sei(); // enable interrupts
    sleep_cpu();
    sleep_disable();
}

void IO_Configuration(){
    DDRB = 0xFF;
    DDRF = 0x00;
    DDRC = 0xFF;
}

unsigned char CRC3(unsigned char f_crctemp){
    unsigned char key2 = 0b00110101;
    s_crctemp=0;
    if (f_crctemp & (1<<6))
    {
        key2 = key2 << 1;
        s_crctemp = f_crctemp;
        s_crctemp = s_crctemp ^ key2;
        if (s_crctemp & (1<<5))
        {
            key2 = key2 >> 1;
            s_crctemp = s_crctemp ^ key2;
            f_crctemp = f_crctemp + s_crctemp;
            return f_crctemp;
        }
```

```
        else{
            f_crctemp = f_crctemp + s_crctemp;
            return f_crctemp;
        }
    }
    else{
        s_crctemp = f_crctemp;
        if (s_crctemp & (1<<5))
        {
            s_crctemp = s_crctemp ^ key2;
            f_crctemp = f_crctemp + s_crctemp;
            return f_crctemp;
        }
        else{
            f_crctemp = f_crctemp + s_crctemp;
            return f_crctemp;
        }
    }
}

unsigned char PWNlevel(unsigned char temp){
    unsigned char firstemp;
    unsigned char secondtemp;
    float x;
    switch(temp)
    {
        case 0x02:
        x=20;
        firstemp=255-((255*x)/100);
        return firstemp;
        break;
        case 0x1e:
        x=80;
        firstemp=255-((255*x)/100);
        return firstemp;
        break;
        default:
        temp=firstemp;
        secondtemp=firstemp-1;
        x=secondtemp*2.145;
        firstemp=255-((255*x)/100);
        return firstemp;
    }
}

void startmotor(){
    unsigned char moisturelevel = PWNlevel(M);
    DDRB |= (1<<3);
    DDRB |= (1<<4);
    DDRC =0xFF;
    PORTC=0;
    OCR0 = moisturelevel;
    PORTC=0xFF;
    TCCR0 = 0x64;
}

void init_motor(){
    DDRB |= (1<<4);
    OCR0 = 0;
    TCCR0 = 0x64;
}

void motor_timer(){
    DDRB |= 0x0;
    TCCR1A = 0x00; //00
    TCCR1B = 0x0D; //1D
    OCR1AH = 0x27; //98
    OCR1AL = 0x00; //96
    TIMSK = (1<<OCIE1A);
    sei ();
}

void lcdData( unsigned char data )
{
    LCD_DPRT = data;
    LCD_CPRT |= (1<<LCD_RS);
    LCD_CPRT &= ~ (1<<LCD_RW);
    LCD_CPRT |= (1<<LCD_EN);
```

```c
    _delay_ms(1);
    LCD_CPRT &= ~ (1<<LCD_EN);
    _delay_ms(1);
}

void lcdCommand( unsigned char command )
{
    LCD_DPRT = command;
    LCD_CPRT &= ~ (1<<LCD_RS)||~ (1<<LCD_RW);
    LCD_CPRT |= (1<<LCD_EN);
    _delay_ms(1);
    LCD_CPRT &= ~ (1<<LCD_EN);
    _delay_ms(1);
}

void SetCursor(unsigned char x, unsigned char y){
    unsigned char f_temp;
    switch(y)
    {
        case 1:
        f_temp=0x80+x-1;
        lcdCommand(f_temp);
        _delay_ms(30);
        break;
        default:
        f_temp=0xC0+x-1;
        lcdCommand(f_temp);
        _delay_ms(30);
    }
}

void lcd_init()
{
    LCD_DDDR = 0xFF;
    LCD_CDDR = 0xFF;
    LCD_CPRT &=~(1<<LCD_EN); //LCD_EN = 0
    _delay_ms(30);
    lcdCommand(0x38);
    lcdCommand(0x0E);
    lcdCommand(0x01);
    _delay_ms(30);
    lcdCommand(0x06);
}

void lcd_print( char * str )
{
    for(unsigned char i = 0; str[i]!='\0';i++)
    {
        _delay_ms(100);
        lcdData(str[i]);
    }
}

void Create_Packetin(unsigned char T,unsigned char M, unsigned char waterlvl,unsigned char batterylvl )
{
    unsigned char t_packet;
    unsigned char m_packet;
    unsigned char w_packet;
    unsigned char b_packet;
    t_packet = T|(1<<7);
    TranCDL(t_packet);
    TranCDL(CRC3(LogRequest));
    m_packet = M|(1<<7)|(1<<5);
    TranCDL(m_packet);
    TranCDL(CRC3(LogRequest));
    w_packet = waterlvl|(1<<7)|(1<<6);
    TranCDL(w_packet);
    TranCDL(CRC3(LogRequest));
    b_packet = batterylvl|(1<<7)|(1<<6)|(1<<5);
    TranCDL(b_packet);
    TranCDL(CRC3(LogRequest));
    sn=1;
}

void ADCconversion (){
    ADCSRA = 0x87;
    ADMUX = 0x20;
    ADCSRA |= (1<<ADSC);
```

```c
    while ((ADCSRA&(1<<ADIF))==0);
    BL = ADCH;
    BL = BL>>3;
    BL = BL & 0x1F;
    ADMUX++;
    ADCSRA |= (1<<ADSC);
    while ((ADCSRA&(1<<ADIF))==0);
    WL = ADCH;
    WL = WL>>3;
    WL = WL & 0x1F;
    ADMUX++;
    ADCSRA |= (1<<ADSC);
    while ((ADCSRA&(1<<ADIF))==0);
    M = ADCH;
    M = M>>3;
    M = M & 0x1F;
    ADMUX++;
    ADCSRA |= (1<<ADSC);
    while ((ADCSRA&(1<<ADIF))==0);
    T = ADCH;
    T = T>>3;
    T = T & 0x1F;
    Create_Packetin(T,M,WL,BL);
}

unsigned char hexToChar(unsigned char value)
{
    if (value> 0x09)
    value+=0x37;
    else
    value|=0x30;
    return value;
}

void print_data(){
    unsigned char str[8] = " =0x \0";
    _delay_ms(300);
    str[1] = 'T';
    str[5] = hexToChar(T>>4);
    str[6] = hexToChar(T&0x0F);
    lcd_print(str);
    _delay_ms(300);
    SetCursor(1,2);
    str[1] = 'M';
    str[5] = hexToChar(M>>4);
    str[6] = hexToChar(M&0x0F);
    lcd_print(str);
    SetCursor(8,1);
    _delay_ms(300);
    str[1] = 'W';
    str[5] = hexToChar(WL>>4);
    str[6] = hexToChar(WL&0x0F);
    lcd_print(str);
    SetCursor(8,2);
    _delay_ms(300);
    str[1] = 'B';
    str[5] = hexToChar(BL>>4);
    str[6] = hexToChar(BL&0x0F);
    lcd_print(str);
    _delay_ms(1000);
    if(BL<0x14){
        lcd_init();
        lcd_print("change battery");
        SetCursor(1,2);
        lcd_print("immediately!");
    }
}

void TimerADC(){
    TCNT3 = 65534;
    TCCR3A = 0x00;
    TCCR3B = (1<<CS30) | (1<<CS32);;
    TIMSK = (1 << TOIE3) ;
    sei();
}
```

```c
void TranCDL(unsigned char ch){
    UCSR0B |= (1 << TXEN0) | (1 << TXCIE0);
    while(!(UCSR0A & (1<<UDRE)));
    UDR0 = ch;
}

int main(void)
{
    TranCDL(CRC3(Acknowledge));
    usart_init();
    sei();
    IO_Configuration;
    lcd_init();
    ADCconversion();
    print_data();
    TimerADC();
    init_motor();
    motor_timer();
    while (1)
    {
        if (motor_open_close==0)
        init_motor();
        else
        startmotor();
    }
    return 0;
}

ISR(TIMER1_COMPA_vect) {
    motor_open_close = !motor_open_close;
}

ISR(USART0_TX_vect){
    if (sn== 0){
        UCSR0B &= ~((1 << TXEN0) | (1 << TXCIE0));
    }
}

ISR(USART0_RX_vect) {
    while (!(UCSR0A & (1<<RXC0) )){}; // Double checking flag
    usartrecieve = UDR0;
}

ISR(TIMER3_OVF_vect) {
    lcd_init();
    ADCconversion();
    print_data();
    TCNT3 = 65534;
}
```

For the User Side Node:

main.c

D:\Seventh Semester\CNG 336\Labs\Lab4\Module4\userNode\userNode\main.c

```c
/*
 * userNode.c
 *
 * Created: 1/2/2023 7:57:47 PM
 *  Author: Erem
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <util/delay.h>
#define DELAY_BACKWARD_COMPATIBLE
#define F_CPU 8000000UL
//#include <util/delay.h>
#define KEY_DDR DDRC //keyboard DDR
#define LCD_CPIN PINB //LCD COMMANDS PIN
#define LCD_CDDR DDRB //LCD COMMANDS DDR
#define KEY_PRT PORTC //keyboard PORT
#define KEY_PIN PINC //keyboard PIN
#define LCD_DPRT PORTA //LCD DATA PORT
#define LCD_DPIN PINA //LCD DATA PIN
#define LCD_CPRT PORTB //LCD COMMANDS PORT
#define LCD_DDDR DDRA //LCD DATA DDR
#define LCD_RS 0 //LCD RS
#define LCD_RW 1 //LCD RW
#define LCD_EN 2 //LCD EN
unsigned char user_nw_ch_read;
unsigned int us1=0;
unsigned char x=1;
unsigned char y=1;
unsigned char keypad[4][3] = { '1','2','3','4','5','6','7','8','9','.','0','#'};
```

```c
void usart_init(void)
{
    UCSR0B = (1<<RXEN0)|(1<<RXCIE0) | (1<<TXEN0) | (1<<TXCIE0);
    UCSR0C = (1<<UCSZ01) | (1<<UCSZ00);
    UBRR0L = 0x33;
    UBRR0H=0x0;
    UCSR1B = ((1<<RXEN1)|(1<<RXCIE1) | (1<<TXEN1) | (1<<TXCIE1));
    UCSR1C = (1<<UCSZ01) | (1<<UCSZ00) ;
    UBRR1L = 0x33;
    UBRR1H=0x0;
}


void lcd_gotoxy(unsigned char x, unsigned char y)
{
    char f_ch_add[]={0x80,0xC0,0x94,0xD4};
    lcdCommand(f_ch_add[y-1] + x - 1);
    _delay_us(100);
}


ISR(USART0_TX_vect){
    UCSR0B &= ~((1 << TXEN0) | (1 << TXCIE0));
}


ISR(USART0_RX_vect) {
    while (!(UCSR0A & (1<<RXC0) )){};
    user_nw_ch_read= UDR0;
    unsigned int user=0;
    if (user_nw_ch_read=='\r')
    {
        y++;
        if (y==5)
        {
            y=1;
            lcdCommand(1);
        }
        lcd_gotoxy(x,y);
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'userNode' (1 project)
- userNode
  - Dependencies
  - Output Files
  - Libraries
  - main.c

VA View VA Outline Solution Explorer

Properties

Search Solution Explorer (Ctrl+;)

Solution 'userNode' (1 project)
- userNode
  - Dependencies
  - Output Files
  - Libraries
  - main.c

VA View VA Outline Solution Explorer

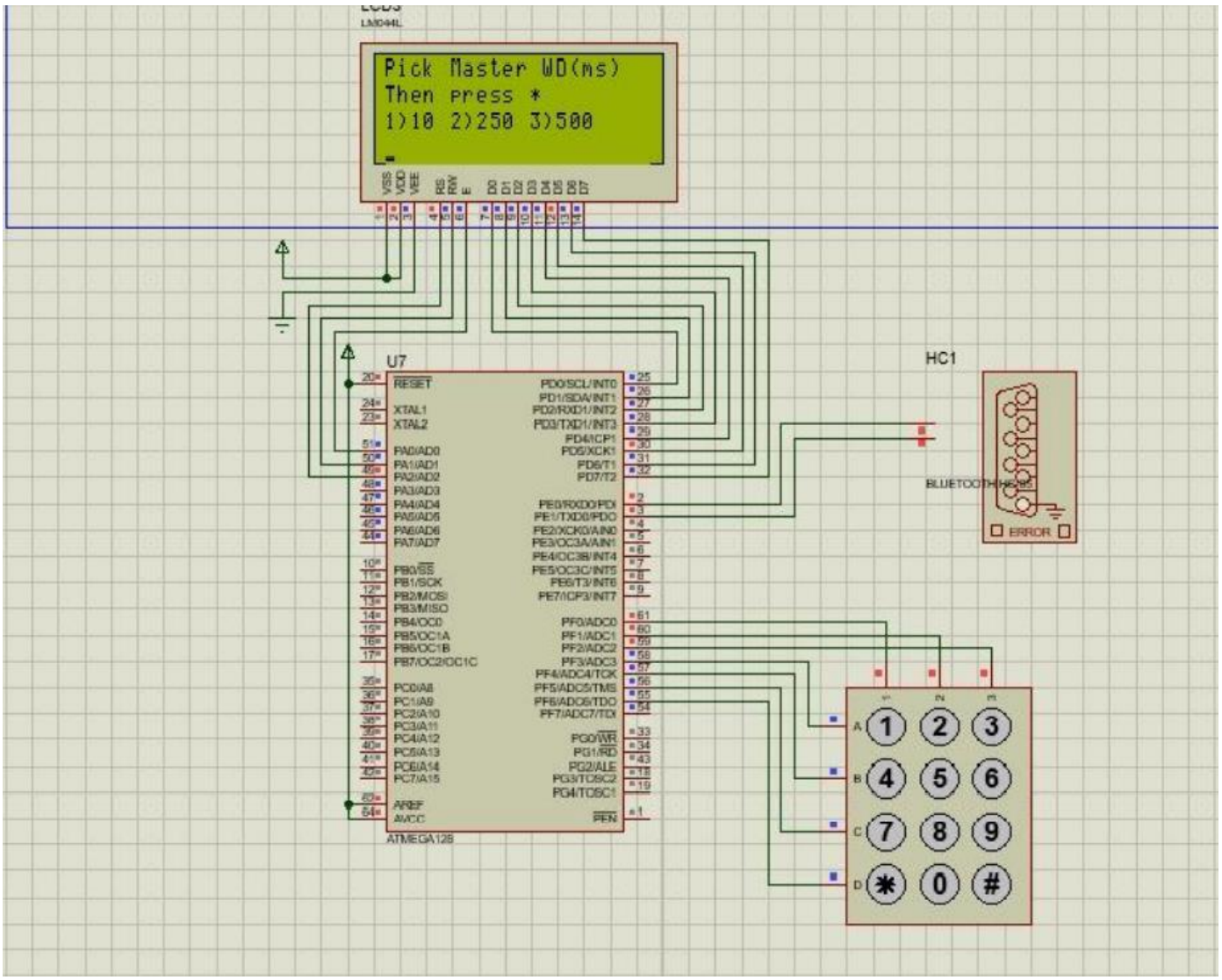
Properties

```
    }
    if (user_nw_ch_read=="\r" )
    {
        user++;
    }
    if(user==0 && us1==1 ){
        lcdCommand(1);
        us1=0;
    }
    lcdData(user_nw_ch_read);
}


void lcdCommand( unsigned char cmnd )
{
    LCD_DPRT = cmnd;
    LCD_CPRT &= ~ (1<<LCD_RS);
    LCD_CPRT &= ~ (1<<LCD_RW);
    LCD_CPRT |= (1<<LCD_EN);
    _delay_us(1);
    LCD_CPRT &= ~ (1<<LCD_EN);
    _delay_us(100);
}


void lcd_init()
{
    LCD_DDDR = 0xFF;
    LCD_CDDR = 0xFF;
    LCD_CPRT &=~(1<<LCD_EN);
    _delay_us(2000);
    lcdCommand(0x38);
    lcdCommand(0x0E);
    lcdCommand(0x01);
    _delay_us(2000);
    lcdCommand(0x06);
}
void lcdData( unsigned char data )
{
    LCD_DPRT = data;
    LCD_CPRT |= (1<<LCD_RS);
    LCD_CPRT &= ~ (1<<LCD_RW);
    LCD_CPRT |= (1<<LCD_EN);
    _delay_us(1);
    LCD_CPRT &= ~ (1<<LCD_EN);
    _delay_us(100);
}


int main(void)
{
    usart_init();
    lcd_init();
    lcd_gotoxy(x,y);
    unsigned char column, row;
    DDRD = 0xFF;
    KEY_DDR = 0xF0;
    KEY_PRT = 0xFF;
    while(1)
    {
        usart_init();
        sei();
        do
        {
            KEY_PRT &= 0x0F;
            column= (KEY_PIN & 0b00000111);
        } while(column != 0b00000111);
        do
        {
            _delay_ms(20);
            column=(KEY_PIN&0b00000111);
            while(column == 0b00000111){
                _delay_ms(20);
                column=(KEY_PIN&0b00000111);
            }
            _delay_ms(20);
```

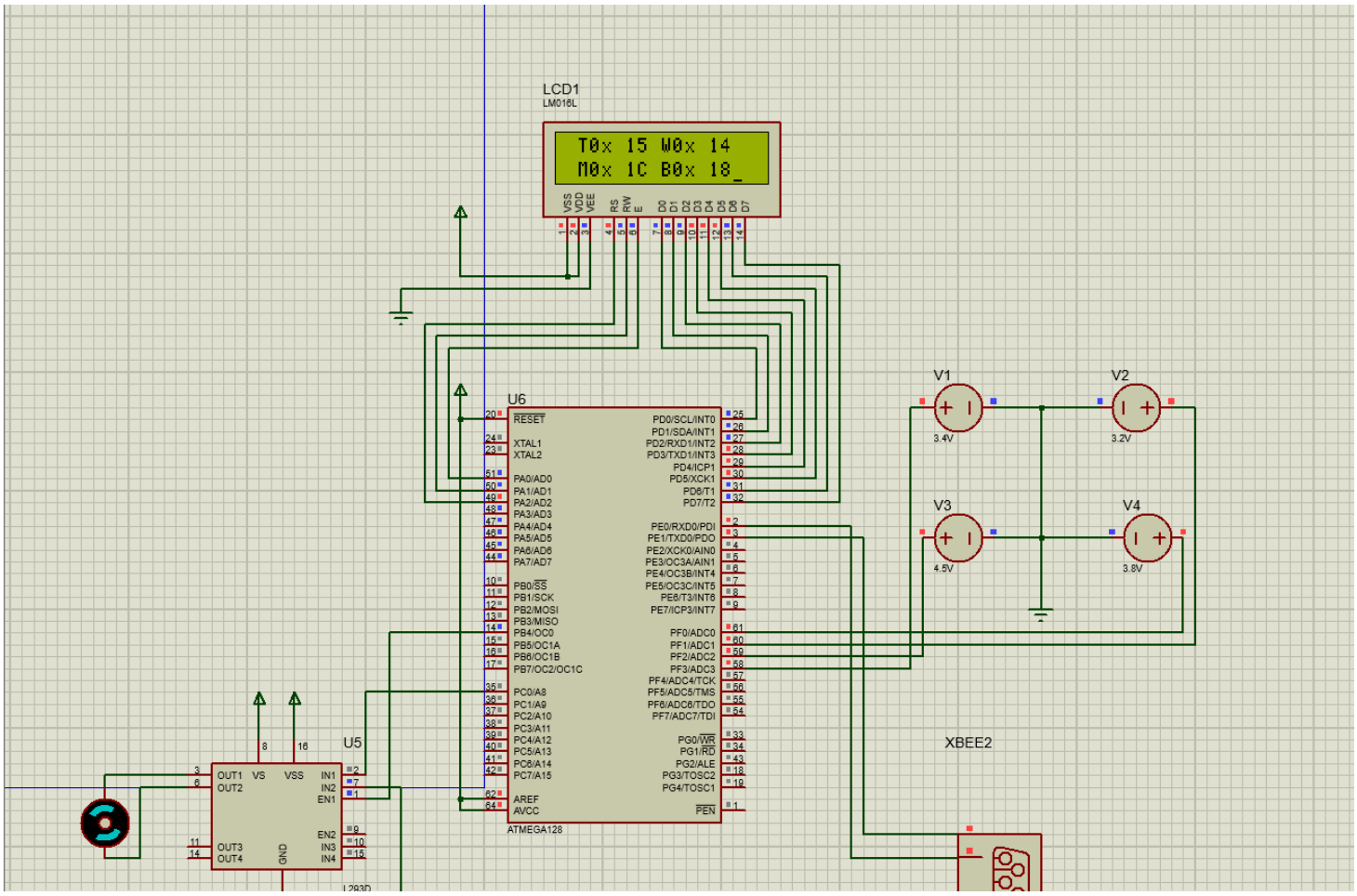
```
            column= (KEY_PIN & 0b00000111);
        }while(column == 0b00000111);
        while(1)
        {
            KEY_PRT = 0xEF; //ground row 0
            column= (KEY_PIN & 0b00000111);
            if(column != 0b00000111)
            {
                row= 0;
                break;
            }
            KEY_PRT = 0xDF;
            column= (KEY_PIN & 0b00000111);
            if(column != 0b00000111)
            {
                row= 1;
                break;
            }
            KEY_PRT = 0xBF;
            column= (KEY_PIN & 0b00000111);
            if(column != 0b00000111)
            {
                row= 2;
                break;
            }
            KEY_PRT = 0x7F;
            column= (KEY_PIN & 0b00000111);
            row= 3;
            break;
        }
        if(column == 0b00000110){
            while(!(UCSR0A & (1<<UDRE0)));
            UDR0=(keypad[row][0]);
        }
        else if(column == 0b00000101){
            while(!(UCSR0A & (1<<UDRE0)));
            UDR0=(keypad[row][1]); //send data
        }
         else if(column == 0b00000011){
             while(!(UCSR0A & (1<<UDRE0)));
             UDR0=(keypad[row][2]);
         }
    }
    return 0 ;
}
```

4.3.3 Design - Proteus

In this figure below, the User Node is shown and it is asking for the WD timer value to be shown on 20x4 LCD and for the user to respond (to select an option), there is a keypad available at the bottom part.



In this figure below, the voltage values (which represent Temperature, Moisture, Water level and battery level) that are not lower than 3.2V are set and now ADC converts these analog values to digital values and displays them in hex.



If the voltage values set are lower than 3.2 Volt, 20x4 LCD screen displays a message saying "change battery immediately!", and this case can be shown in the figure below.