8η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"

Εφαρμογή Internet of Things

Δρυς-Πεντζάκης Οδυσσεύς el19192
Μαρκαντωνάτος Γεράσιμος el19149

Χρησιμοποιήσαμε το PORT EXPANDER για να μπορέσουμε να χρησιμοποιήσουμε ταυτόχρονα το keypad και την οθόνη, και χρησιμοποιήσαμε το θερμόμετρο μέσω του PORTD.
Για να προσομοιώσουμε σωστά την θερμοκρασία ανθρώπου προσθέσαμε στην τιμή που παίρναμε από τον αισθητήρα, και για να παίρνουμε τιμές πίεσης 0-20 από τον ADC τετραπλασιάσαμε την τιμή του ADC έτσι ώστε από εκεί που οι τιμές του ήταν 0-5 τώρα είναι 0-20.
Πρώτα ελέγχουμε αν η τιμή της θερμοκρασίας είναι εντός ορίων και αν δεν είναι στέλνουμε στο status του payload και εμφανίζουμε στην οθόνη το μήνυμα CHECK TEMP, αλλιώς ελέγχουμε και την τιμή της πίεσης και αν είναι εκτός ορίων εμφανίζουμε CHECK PRESSURE.

Κώδικας:

```
#include <stdlib.h>
#include <stdio.h>
#include <avr/io.h>
#define          F_CPU   16000000UL
#include <util/delay.h>
#include<avr/interrupt.h>
#define cbi(reg,bit) (reg &= ~(1 << bit))
#define sbi(reg,bit) (reg |= (1 << bit))
#define PCA9555_0_ADDRESS 0x40 //A0=A1=A2=0 by hardware
#define TWI_READ 1 // reading from twi device
#define TWI_WRITE 0 // writing to twi device
#define SCL_CLOCK 100000L // twi clock in Hz
//Fscl=Fcpu/(16+2*TWBR0_VALUE*PRESCALER_VALUE)
#define TWBR0_VALUE ((F_CPU/SCL_CLOCK)-16)/2
// PCA9555 REGISTERS
typedef enum {
REG_INPUT_0 = 0,
REG_INPUT_1 = 1,
REG_OUTPUT_0 = 2,
REG_OUTPUT_1 = 3,
REG_POLARITY_INV_0 = 4,
```

```
REG_POLARITY_INV_1 = 5,
REG_CONFIGURATION_0 = 6,
REG_CONFIGURATION_1 = 7
} PCA9555_REGISTERS;
//---------- Master Transmitter/Receiver -------------------
#define TW_START 0x08
#define TW_REP_START 0x10
//--------------- Master Transmitter ----------------------
#define TW_MT_SLA_ACK 0x18
#define TW_MT_SLA_NACK 0x20
#define TW_MT_DATA_ACK 0x28
//--------------- Master Receiver ----------------
#define TW_MR_SLA_ACK 0x40
#define TW_MR_SLA_NACK 0x48
#define TW_MR_DATA_NACK 0x58
#define TW_STATUS_MASK 0b11111000
#define TW_STATUS (TWSR0 & TW_STATUS_MASK)
//initialize TWI clock
void twi_init(void)
{
TWSR0 = 0; // PRESCALER_VALUE=1
TWBR0 = TWBR0_VALUE; // SCL_CLOCK 100KHz
}
// Read one byte from the twi device (request more data from device)
unsigned char twi_readAck(void)
{
TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
while(!(TWCR0 & (1<<TWINT)));
return TWDR0;
}
//Read one byte from the twi device, read is followed by a stop condition
unsigned char twi_readNak(void)
{
TWCR0 = (1<<TWINT) | (1<<TWEN);
while(!(TWCR0 & (1<<TWINT)));
return TWDR0;
}
// Issues a start condition and sends address and transfer direction.
// return 0 = device accessible, 1= failed to access device
unsigned char twi_start(unsigned char address)
{
uint8_t twi_status;
// send START condition
TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
// wait until transmission completed
while(!(TWCR0 & (1<<TWINT)));
// check value of TWI Status Register.
```

```
twi_status = TW_STATUS & 0xF8;
if ( (twi_status != TW_START) && (twi_status != TW_REP_START)) return 1;
// send device address
TWDR0 = address;
TWCR0 = (1<<TWINT) | (1<<TWEN);
// wail until transmission completed and ACK/NACK has been received
while(!(TWCR0 & (1<<TWINT)));
// check value of TWI Status Register.
twi_status = TW_STATUS & 0xF8;
if ( (twi_status != TW_MT_SLA_ACK) && (twi_status != TW_MR_SLA_ACK) )
{
return 1;
}
return 0;
}
// Send start condition, address, transfer direction.
// Use ack polling to wait until device is ready
void twi_start_wait(unsigned char address)
{
uint8_t twi_status;
while ( 1 )
{
// send START condition
TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
// wait until transmission completed
while(!(TWCR0 & (1<<TWINT)));
// check value of TWI Status Register.
twi_status = TW_STATUS & 0xF8;
if ( (twi_status != TW_START) && (twi_status != TW_REP_START)) continue;
// send device address
TWDR0 = address;
TWCR0 = (1<<TWINT) | (1<<TWEN);
// wail until transmission completed
while(!(TWCR0 & (1<<TWINT)));
// check value of TWI Status Register.
twi_status = TW_STATUS & 0xF8;
if ( (twi_status == TW_MT_SLA_NACK )||(twi_status ==TW_MR_DATA_NACK) )
{
/* device busy, send stop condition to terminate write operation */
TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
// wait until stop condition is executed and bus released
while(TWCR0 & (1<<TWSTO));
continue;
}
break;
}
}
```

```
// Send one byte to twi device, Return 0 if write successful or 1 if write failed
unsigned char twi_write( unsigned char data )
{
// send data to the previously addressed device
TWDR0 = data;
TWCR0 = (1<<TWINT) | (1<<TWEN);
// wait until transmission completed
while(!(TWCR0 & (1<<TWINT)));
if( (TW_STATUS & 0xF8) != TW_MT_DATA_ACK) return 1;
return 0;
}
// Send repeated start condition, address, transfer direction
//Return: 0 device accessible
// 1 failed to access device
unsigned char twi_rep_start(unsigned char address)
{
return twi_start( address );
}
// Terminates the data transfer and releases the twi bus
void twi_stop(void)
{
// send stop condition
TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
// wait until stop condition is executed and bus released
while(TWCR0 & (1<<TWSTO));
}
void PCA9555_0_write(PCA9555_REGISTERS reg, uint8_t value)
{
twi_start_wait(PCA9555_0_ADDRESS + TWI_WRITE);
twi_write(reg);
twi_write(value);
twi_stop();
}
uint8_t PCA9555_0_read(PCA9555_REGISTERS reg)
{
uint8_t ret_val;
twi_start_wait(PCA9555_0_ADDRESS + TWI_WRITE);
twi_write(reg);
twi_rep_start(PCA9555_0_ADDRESS + TWI_READ);
ret_val = twi_readNak();
twi_stop();
return ret_val;
}



uint8_t one_wire_receive_bit(){
```

```
	uint8_t bit,temp;
	sbi(DDRD,PD4);
	cbi(PORTD,PD4);
	_delay_us(2);
	cbi(DDRD,PD4);
	cbi(PORTD,PD4);
	_delay_us(10);
	temp = (PIND & 0x10);
	bit = 0x00;
	if (temp == 0x10) bit = 0x01;
	_delay_us(49);
	return bit;
}

uint8_t one_wire_receive_byte(){
	uint8_t bit;
	uint8_t byte = 0x00;
	uint8_t i = 0x08;
	while(i != 0){
		bit = one_wire_receive_bit();
		byte = (byte >> 1);
		if (bit == 0x01) bit = 0x80;
		byte = (byte | bit);
		i--;
	}
	return byte;
}

void one_wire_transmit_bit(uint8_t bit){
	sbi(DDRD,PD4);
	cbi(PORTD,PD4);
	_delay_us(2);
	if (bit == 0x01) sbi(PORTD,PD4);
	if (bit == 0x00) cbi(PORTD,PD4);
	_delay_us(58);
	cbi(DDRD,PD4);
	cbi(PORTD,PD4);
	_delay_us(1);
	return;
}

void one_wire_transmit_byte(uint8_t byte){
	uint8_t bit;
	uint8_t i = 0x08;
	while(i != 0){
		bit = (byte & 0x01);
		one_wire_transmit_bit(bit);
```

```
                byte = (byte >> 1);
                i--;
        }
        return;
}

uint8_t one_wire_reset(){
        sbi(DDRD,PD4);
        cbi(PORTD,PD4);
        _delay_us(480);
        cbi(DDRD,PD4);
        cbi(PORTD,PD4);
        _delay_us(100);
        uint8_t temp = PIND;
        _delay_us(380);
        temp = (temp & 0x10);
        uint8_t res = 0x00;
        if (temp == 0x00)
      res = 0x01;
        return res;
}

int f=0;
uint8_t key1,key2;
uint8_t temp_lo, temp_hi, temp_sign, temp_dec,one,two,three,four;
uint16_t temp_final, temp_hi_16, temp_final_o;
uint8_t dec_1 = 0;
      uint8_t dec_2 = 0;
      uint8_t dec_3 = 0;
      uint8_t dec_4 = 0;
      int sum=0;
void usart_init(unsigned int ubrr){
UCSR0A=0;
UCSR0B=(1<<RXEN0)|(1<<TXEN0);
UBRR0H=(unsigned char)(ubrr>>8);
UBRR0L=(unsigned char)ubrr;
UCSR0C=(3 << UCSZ00);
return;
}
/* Routine: usart_transmit
Description:
This routine sends a byte of data
using usart.
parameters:
data: the byte to be transmitted
return value: None. */
void usart_transmit(uint8_t data){
```

```
while(!(UCSR0A&(1<<UDRE0)));
UDR0=data;
}

/* Routine: usart_receive
Description:
This routine receives a byte of data
from usart.
parameters: None.
return value: the received byte */
uint8_t usart_receive(){
while(!(UCSR0A&(1<<RXC0)));
return UDR0;
}
void usart_connect() {
    usart_transmit('E');
    usart_transmit('S');
    usart_transmit('P');
    usart_transmit(':');
    usart_transmit('c');
    usart_transmit('o');
    usart_transmit('n');
    usart_transmit('n');
    usart_transmit('e');
    usart_transmit('c');
    usart_transmit('t');
    usart_transmit('\n');

    return;
}



void usart_url()  {
    usart_transmit('E');
    usart_transmit('S');
    usart_transmit('P');
    usart_transmit(':');
    usart_transmit('u');
    usart_transmit('r');
    usart_transmit('l');
    usart_transmit(':');
    usart_transmit('"');
    usart_transmit('h');
    usart_transmit('t');
    usart_transmit('t');
    usart_transmit('p');
```

```
    usart_transmit(':');
    usart_transmit('/');
    usart_transmit('/');
    usart_transmit('1');
    usart_transmit('9');
    usart_transmit('2');
    usart_transmit('.');
    usart_transmit('1');
    usart_transmit('6');
    usart_transmit('8');
    usart_transmit('.');
    usart_transmit('1');
    usart_transmit('.');
    usart_transmit('2');
    usart_transmit('5');
    usart_transmit('0');
    usart_transmit(':');
    usart_transmit('5');
    usart_transmit('0');
    usart_transmit('0');
    usart_transmit('0');
    usart_transmit('/');
    usart_transmit('d');
    usart_transmit('a');
    usart_transmit('t');
    usart_transmit('a');
    usart_transmit("");
    usart_transmit('\n');
    return;
}


void write_2_nibbles(uint8_t c) {
    uint8_t temp = c;
    uint8_t prev = PCA9555_0_read(REG_INPUT_0);
    prev &= 0x0F;
    c &= 0xF0;
    c |= prev;
    PCA9555_0_write(REG_OUTPUT_0, c);
    c |= 0x08;
    PCA9555_0_write(REG_OUTPUT_0, c);
    c &= 0xF7;
    PCA9555_0_write(REG_OUTPUT_0, c);

    c = temp;
    c &= 0x0F;
    c = c << 4;
```

```
    c |= prev;
    PCA9555_0_write(REG_OUTPUT_0, c);
    c |= 0x08;
    PCA9555_0_write(REG_OUTPUT_0, c);
    c &= 0xF7;
    PCA9555_0_write(REG_OUTPUT_0, c);

    return;
}

void LCD_data(uint8_t c) {
    uint8_t temp = PCA9555_0_read(REG_INPUT_0);
    temp |= 0x04;
    PCA9555_0_write(REG_OUTPUT_0, temp);
    write_2_nibbles(c);
    _delay_us(100);
    return;
}

void LCD_command(uint8_t c) {
    uint8_t temp = PCA9555_0_read(REG_INPUT_0);
    temp &= 0xFB;
    PCA9555_0_write(REG_OUTPUT_0, temp);
    write_2_nibbles(c);
    _delay_us(100);
    return;
}

void LCD_init(void) {
    _delay_ms(40);

    PCA9555_0_write(REG_OUTPUT_0, 0x30);
    PCA9555_0_write(REG_OUTPUT_0, 0x38);
    PCA9555_0_write(REG_OUTPUT_0, 0x30);

    _delay_us(100);

    PCA9555_0_write(REG_OUTPUT_0, 0x30);
    PCA9555_0_write(REG_OUTPUT_0, 0x38);
    PCA9555_0_write(REG_OUTPUT_0, 0x30);

    _delay_us(100);

    PCA9555_0_write(REG_OUTPUT_0, 0x20);
    PCA9555_0_write(REG_OUTPUT_0, 0x28);
    PCA9555_0_write(REG_OUTPUT_0, 0x20);
```

```
    _delay_us(100);

    LCD_command(0x28);
    LCD_command(0x0C);
    LCD_command(0x01);
    _delay_us(5000);

    LCD_command(0x06);
    return;
}



uint8_t scan_row (int row){
    if (row==1){
        PCA9555_0_write(REG_OUTPUT_1, 0xFE);

         uint8_t temp ;
        temp =PCA9555_0_read(REG_INPUT_1);
        return temp;


    }
    if (row==2){
        PCA9555_0_write(REG_OUTPUT_1, 0xFD);

         uint8_t temp ;
         temp =PCA9555_0_read(REG_INPUT_1);
         return temp;

    }
    if (row==3){
        PCA9555_0_write(REG_OUTPUT_1, 0xFB);

         uint8_t temp ;
        temp =PCA9555_0_read(REG_INPUT_1);
        return temp;

    }
    if (row==4){
        PCA9555_0_write(REG_OUTPUT_1, 0xF7);

         uint8_t temp ;
        temp =PCA9555_0_read(REG_INPUT_1);
        return temp;
```

```
    }
}


uint8_t scan_keypad (){
    uint8_t row1, row2, row3, row4;
    row1= scan_row (1);
    row2= scan_row (2);
    row3= scan_row (3);
    row4= scan_row (4);
    uint8_t temp1= row1&0b11110000;
    uint8_t temp2= row2&0b11110000;
    uint8_t temp3= row3&0b11110000;
    uint8_t temp4= row4&0b11110000;
    if (temp1!=0b11110000)
        return row1;
   else if (temp2!=0b11110000)
        return row2;
    else if (temp3!=0b11110000)
        return row3;
    else if (temp4!=0b11110000)
        return row4;
   else return 0;


}




uint8_t scan_keypad_rising_edge() {
    uint8_t start, finish;
   start = scan_keypad();
   _delay_ms(10);
   finish = scan_keypad();

   if ((start!=finish) && (finish==0) ) {
       return start;}
   return 0;
}

uint8_t keypad_to_ascii() {
    uint8_t chari;
    chari = scan_keypad_rising_edge(); //
    if (chari == 0b11101110) return 0b00101010; //*
    if (chari == 0b11011110) return 0b00110000; //0
    if (chari == 0b10111110) return 0b00100011;
```

```
    if (chari == 0b01111110) return 0b01000100;
    if (chari == 0b11101101) return 0b00110111;
    if (chari == 0b11011101) return 0b00111000;
    if (chari == 0b10111101) return 0b00111001;
    if (chari == 0b01111101) return 0b01000011;
    if (chari == 0b11101011) return 0b00110100;//B
    if (chari == 0b11011011) return 0b00110101;
    if (chari == 0b10111011) return 0b00110110;
    if (chari == 0b01111011) return 0b01000010;
    if (chari == 0b11100111) return 0b00110001;
    if (chari == 0b11010111) return 0b00110010;
    if (chari == 0b10110111) return 0b00110011;
    if (chari == 0b01110111) return 0b01000001;
    return 0;
}

void print_lcd(uint8_t num, int check){
    num |= 0x30;
    if (check == 1) {
        LCD_command(0x01);
            _delay_ms(2);
            LCD_data(num);
                                LCD_data('.');
            LCD_data('S');
            LCD_data('u');
            LCD_data('c');
            LCD_data('c');
            LCD_data('e');
            LCD_data('s');
            LCD_data('s');
    }
    else {
        LCD_command(0x01);
            _delay_ms(2);
            LCD_data(num);
                                LCD_data('.');
            LCD_data('F');
            LCD_data('a');
            LCD_data('i');
            LCD_data('l');

    }
    return;
}
ISR (TIMER1_OVF_vect){
    f = 1;
```

```
}
uint8_t a1,a2,a3,a4,a5,a6,a7,a8,a9;
uint8_t key,i,j,k;
static volatile float adc;
static volatile uint8_t adc1;
static volatile uint8_t adc2;
static volatile uint8_t adc3;
float decimal;



int main(void) {
    twi_init();

          DDRB = 0x3F;
          DDRD = 0xFF;
    DDRC = 0x00;
    ADMUX = (1 << REFS0); //| (1 << MUX1);
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
    ADCSRB = 0x00;
    DIDR0 = ~(1 << ADC2D);

    PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
    PCA9555_0_write(REG_CONFIGURATION_1, 0xF0);
    usart_init(103);
    TIMSK1=(1<<TOIE1);
    TCCR1B=(1<<CS12) | (0<<CS11) | (1<<CS10);
    sei();


          while(1) {
    LCD_init();
    _delay_ms(2);



    usart_connect();
    a1 = usart_receive();
    a2 = usart_receive();
    if (a2 == 'S') {
       a1 = usart_receive();
       a1 = usart_receive();
       a1 = usart_receive();
       a1 = usart_receive();
       a1 = usart_receive();
       a1 = usart_receive();
       a1 = usart_receive();
```

```
    a1 = usart_receive();
    print_lcd(1,1);
}
else if(a2 == 'F') {
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    print_lcd(1,2);
}
_delay_ms(5000);

usart_url();
a1 = usart_receive();
a2 = usart_receive();
if (a2 == 'S') {
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    print_lcd(2,1);
}
else if(a2 == 'F') {
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    a1 = usart_receive();
    print_lcd(2,2);
}

_delay_ms(5000);

//Temperature
if (!one_wire_reset()) {


                    continue;
                }
                one_wire_transmit_byte(0xCC);
                one_wire_transmit_byte(0x44);
                while(one_wire_receive_bit() != 0x01);
```

```
            if (!one_wire_reset()) {



                    continue;
            }

            one_wire_transmit_byte(0xCC);
            one_wire_transmit_byte(0xBE);
            temp_lo = one_wire_receive_byte();
            temp_hi = one_wire_receive_byte();
temp_dec = temp_lo & 0x0F;
            temp_sign = temp_hi & 0xF8;
temp_hi_16 = temp_hi << 8;
temp_final = (temp_hi_16 + temp_lo);

if (temp_sign == 0xF8) {
   temp_final_o = 65534-temp_final;
}

            else
   temp_final_o = temp_final;

temp_dec = temp_final_o&15;

 dec_1 = 0;
 dec_2 = 0;
 dec_3 = 0;
 dec_4 = 0;
 sum=0;
 one = temp_dec&0x08;
if (one == 8)sum = sum + 5000;
 one = temp_dec&0x04;
if ( one == 4) sum = sum + 2500;
one = temp_dec&0x02;
if (one == 2) sum = sum + 1250;
one = temp_dec&0x01;
if (one == 1) sum = sum + 625;

while (sum >= 1000) {
   dec_1++;
   sum = sum - 1000;
}

while (sum >= 100) {
   dec_2++;
   sum = sum - 100;
```

```
}

while (sum >= 10) {
   dec_3++;
   sum = sum - 10;
}

while (sum >= 1) {
   dec_4++;
   sum = sum - 1;
}

i = 0;
j = 0;
k = 0;

temp_final_o = temp_final_o>>4;

while (temp_final_o >= 100) {
   i++;
   temp_final_o = temp_final_o - 100;
}

while (temp_final_o >= 10) {
   j++;
   temp_final_o = temp_final_o - 10;
}

while (temp_final_o >= 1) {
   k++;
   temp_final_o = temp_final_o - 1;
}
j = 3;
//POT0
ADCSRA |= (1 << ADSC);
while ((ADCSRA & (1 << ADSC)) == (1 << ADSC));

adc = ADC;
adc = (adc * 20) / 1024;
if (adc>=10) adc = adc/10;
adc1 = (uint8_t)adc;
decimal = adc - adc1;
adc2 = (uint8_t)(decimal * 10);
adc3 = (uint8_t)(((decimal * 10) - adc2) * 10);


LCD_init();
```

```
_delay_ms(2);
uint8_t t1,t2,t3,t4,t5,t6,p1,p2;
f=0;
t1 = j;
t2 = k;
t3 = dec_1;
t4 = dec_2;
t5 = dec_3;
t6 = dec_4;

t1 |= 0x30;
t2 |= 0x30;

t3 |= 0x30;
t4 |= 0x30;
t5 |= 0x30;
t6 |= 0x30;
p1=adc1;
p2=adc2;
p1 |= 0x30;
p2 |= 0x30;

if(t1!=0x30)LCD_data(t1);
LCD_data(t2);
LCD_data('.');
LCD_data(t3);
LCD_data(t4);
LCD_data(t5);
LCD_data(t6);
LCD_data(223);
LCD_data('C');
LCD_data(' ');
LCD_data(p1);
if (adc1>1) LCD_data('.');
LCD_data(p2);
LCD_data(' ');
LCD_data('c');
LCD_data('m');

LCD_command(0b11000000);
_delay_ms(2);
int flag1 = 0;
if ((k < 4) | (k > 7)) {
   flag1 = 1;
   LCD_data('C');
   LCD_data('H');
   LCD_data('E');
```

```
    LCD_data('C');
    LCD_data('K');
    LCD_data(' ');
    LCD_data('T');
    LCD_data('E');
    LCD_data('M');
    LCD_data('P');

}
else {
    if (((adc2 > 2) && (adc1 ==1)) |((adc2 < 4) && (adc1 > 1))) {
        flag1 = 2;
        LCD_data('C');
        LCD_data('H');
        LCD_data('E');
        LCD_data('C');
        LCD_data('K');
        LCD_data(' ');
        LCD_data('P');
        LCD_data('R');
        LCD_data('E');
        LCD_data('S');
        LCD_data('S');
        LCD_data('U');
        LCD_data('R');
        LCD_data('E');
    }
    else {
        LCD_data('O');
        LCD_data('K');
    }
}
if ( flag1 == 0) {
    TCNT1=10847;
    while(1) {
    key1 = keypad_to_ascii();
    if((key1 !=0) | (f == 1)) break;
    }
    if ( key1 == 0x35) {
        flag1 = 3;
            LCD_init();
            _delay_ms(2);
            if(t1!=0x30)LCD_data(t1);
            LCD_data(t2);
            LCD_data('.');
            LCD_data(t3);
            LCD_data(t4);
```

```
        LCD_data(t5);
        LCD_data(t6);
        LCD_data(223);
        LCD_data('C');
        LCD_data(' ');
        LCD_data(p1);
        if (adc1>1) LCD_data('.');
        LCD_data(p2);
        LCD_data(' ');
        LCD_data('c');
        LCD_data('m');
        LCD_command(0b11000000);
        _delay_ms(2);
        LCD_data('N');
        LCD_data('U');
        LCD_data('R');
        LCD_data('S');
        LCD_data('E');
        LCD_data(' ');
        LCD_data('C');
        LCD_data('A');
        LCD_data('L');
        LCD_data('L');

}
TCNT1=10847;
while(1) {
key2 = keypad_to_ascii();
if((key2 !=0) | (f == 1)) break;
}
if ( key2 == '#') {
        LCD_init();
        _delay_ms(2);
        if(t1!=0x30)LCD_data(t1);
        LCD_data(t2);
        LCD_data('.');
        LCD_data(t3);
        LCD_data(t4);
        LCD_data(t5);
        LCD_data(t6);
        LCD_data(223);
        LCD_data('C');
        LCD_data(' ');
        LCD_data(p1);
        if (adc1>1) LCD_data('.');
        LCD_data(p2);
        LCD_data(' ');
```

```
            LCD_data('c');
            LCD_data('m');
            LCD_command(0b11000000);
            _delay_ms(2);
            LCD_data('O');
            LCD_data('K');
            flag1 = 0;
        }
    }
_delay_ms(5000);
j |= 0x30;
k |= 0x30;
dec_1 |= 0x30;
adc2 |= 0x30;
usart_transmit('E');
usart_transmit('S');
usart_transmit('P');
usart_transmit(':');
usart_transmit('p');
usart_transmit('a');
usart_transmit('y');
usart_transmit('l');
usart_transmit('o');
usart_transmit('a');
usart_transmit('d');
usart_transmit(':');
usart_transmit('[');
usart_transmit('{');
usart_transmit('"');
usart_transmit('n');
usart_transmit('a');
usart_transmit('m');
usart_transmit('e');
usart_transmit('"');
usart_transmit(':');
usart_transmit(' ');
usart_transmit('"');
usart_transmit('t');
usart_transmit('e');
usart_transmit('m');
usart_transmit('p');
usart_transmit('e');
usart_transmit('r');
usart_transmit('a');
usart_transmit('t');
usart_transmit('u');
usart_transmit('r');
```

```
usart_transmit('e');
usart_transmit("");
usart_transmit(',');
usart_transmit("");
usart_transmit('v');
usart_transmit('a');
usart_transmit('l');
usart_transmit('u');
usart_transmit('e');
usart_transmit("");
usart_transmit(':');
usart_transmit(' ');
usart_transmit("");
usart_transmit(j);
usart_transmit(k);
usart_transmit('.');
usart_transmit(dec_1);
usart_transmit("");
usart_transmit('}');
usart_transmit(',');
usart_transmit('{');
usart_transmit("");
usart_transmit('n');
usart_transmit('a');
usart_transmit('m');
usart_transmit('e');
usart_transmit("");
usart_transmit(':');
usart_transmit(' ');
usart_transmit("");
usart_transmit('p');
usart_transmit('r');
usart_transmit('e');
usart_transmit('s');
usart_transmit('s');
usart_transmit('u');
usart_transmit('r');
usart_transmit('e');
usart_transmit("");
usart_transmit(',');
usart_transmit("");
usart_transmit('v');
usart_transmit('a');
usart_transmit('l');
usart_transmit('u');
usart_transmit('e');
usart_transmit("");
```

```
usart_transmit(':');
usart_transmit(' ');
usart_transmit("");
if (adc1>1) {
   usart_transmit('0');
   usart_transmit(p1);
   usart_transmit('.');
   usart_transmit(adc2);
}
else {
usart_transmit(p1);
usart_transmit(adc2);
usart_transmit('.');
usart_transmit('0');
}
usart_transmit("");
usart_transmit('}');
usart_transmit(',');
usart_transmit('{');
usart_transmit("");
usart_transmit('n');
usart_transmit('a');
usart_transmit('m');
usart_transmit('e');
usart_transmit("");
usart_transmit(':');
usart_transmit(' ');
usart_transmit("");
usart_transmit('t');
usart_transmit('e');
usart_transmit('a');
usart_transmit('m');
usart_transmit("");
usart_transmit(',');
usart_transmit("");
usart_transmit('v');
usart_transmit('a');
usart_transmit('l');
usart_transmit('u');
usart_transmit('e');
usart_transmit("");
usart_transmit(':');
usart_transmit(' ');
usart_transmit("");
usart_transmit('5');
usart_transmit('5');
usart_transmit("");
```

```
usart_transmit('}');
usart_transmit(',');
usart_transmit('{');
usart_transmit('"');
usart_transmit('n');
usart_transmit('a');
usart_transmit('m');
usart_transmit('e');
usart_transmit('"');
usart_transmit(':');
usart_transmit(' ');
usart_transmit('"');
usart_transmit('s');
usart_transmit('t');
usart_transmit('a');
usart_transmit('t');
usart_transmit('u');
usart_transmit('s');
usart_transmit('"');
usart_transmit(',');
usart_transmit('"');
usart_transmit('v');
usart_transmit('a');
usart_transmit('l');
usart_transmit('u');
usart_transmit('e');
usart_transmit('"');
usart_transmit(':');
usart_transmit(' ');
usart_transmit('"');
if (flag1 == 0 ) {
   usart_transmit('O');
   usart_transmit('K');
}
else if ( flag1 == 1) {
   usart_transmit('C');
   usart_transmit('H');
   usart_transmit('E');
   usart_transmit('C');
   usart_transmit('K');
   //usart_transmit(' ');
   usart_transmit('T');
   usart_transmit('E');
   usart_transmit('M');
   usart_transmit('P');

}
```

```
else if ( flag1 == 2) {
   usart_transmit('C');
   usart_transmit('H');
   usart_transmit('E');
   usart_transmit('C');
   usart_transmit('K');
   //usart_transmit(' ');
   usart_transmit('P');
   usart_transmit('R');
   usart_transmit('E');
   usart_transmit('S');
   usart_transmit('S');
   usart_transmit('U');
   usart_transmit('R');
   usart_transmit('E');
}
else if ( flag1 == 3) {
   usart_transmit('N');
   usart_transmit('U');
   usart_transmit('R');
   usart_transmit('S');
   usart_transmit('E');
   //usart_transmit(' ');
   usart_transmit('C');
   usart_transmit('A');
   usart_transmit('L');
   usart_transmit('L');

}
usart_transmit("");
usart_transmit('}');
usart_transmit(']');
usart_transmit('\n');



a1 = usart_receive();
a2 = usart_receive();
if (a2 == 'S') {
   a3 = usart_receive();
   a1 = usart_receive();
   a1 = usart_receive();
   a1 = usart_receive();
   a1 = usart_receive();
   a1 = usart_receive();
   a1 = usart_receive();
   a1 = usart_receive();
```

```
        _delay_ms(2);
        print_lcd(3,1);
    }
    else if(a2 == 'F') {
        a1 = usart_receive();
        a1 = usart_receive();
        a1 = usart_receive();
        a1 = usart_receive();
        a1 = usart_receive();
        LCD_init();
        _delay_ms(2);
        print_lcd(3,2);
}


    _delay_ms(2000);

    usart_transmit('E');
    usart_transmit('S');
    usart_transmit('P');
    usart_transmit(':');
    usart_transmit('t');
    usart_transmit('r');
    usart_transmit('a');
    usart_transmit('n');
    usart_transmit('s');
    usart_transmit('m');
    usart_transmit('i');
    usart_transmit('t');
    usart_transmit('\n');

    a1 = usart_receive();
    a2 = usart_receive();
    a3 = usart_receive();
    a4 = usart_receive();
    a5 = usart_receive();
    a6 = usart_receive();
    a7 = usart_receive();

        LCD_init();
        _delay_ms(2);
        LCD_data('4');
        LCD_data('.');
        LCD_data(a1);
        LCD_data(a2);
        LCD_data(a3);
        LCD_data(a4);
```

```
        LCD_data(a5);
        LCD_data(a6);
    if (a1 == 'F') {
        LCD_init();
        _delay_ms(2);
        LCD_data('4');
        LCD_data('.');
        LCD_data('2');
        LCD_data('0');
        LCD_data('0');
        LCD_data(' ');
        LCD_data('O');
        LCD_data('K');

    }
   _delay_ms(2000);
}
}
```