DEPARTMENT OF PHYSICS, UNIVERSITY OF COLOMBO

PH3032-EMBEDDED SYSTEM LABORATORY

HOME AUTOMATION SYSTEM

Name : Imesh Anuththara

Index: S14345

Date: 03/06/2021

Abstract

Home Automation System is a System which is can be used to easily control the utilities and features of home. It makes life more convenient and secure. Most of modern houses consist of Home Automation System and Security System. Basically, this project only consists few features of whole Home Automation System. Atmega32 microcontroller is used for this project.

This project has 3 stages. In the first stage there is a password base door lock. If user enter the correct password using the keypad & the door will be opened. If password is wrong buzzer will be activated. In the second stage automatically room temperature control fan. If room temperature become higher than pre-defined value then fan will be switched on. In the third stage there is a remote-control light sensor bulb. In the night, bulb automatically switch on and it can be off using Remote.

Table of Contents

1. Introduction

Technology has been developed in different fields with the time. Most of devices that humans are using day to day life, are created using new technology. Microcontroller is the one of the great inventions. As a result of that high quality electronic devices are made.

1.1 Microcontroller

In this project Atmega32 microcontroller is used to create Home Automation System. Atmega32 has 40 pins and most of pins are specified to special function. It has 4 ports such as **PORTA, PORTB, PORTC** and **PORTD**. Each port has 8 pins. All ports are general purpose and most of ports have dual functions. There are 3 different registers (DDRx, PORTx, PINx) are related to the various port operations.

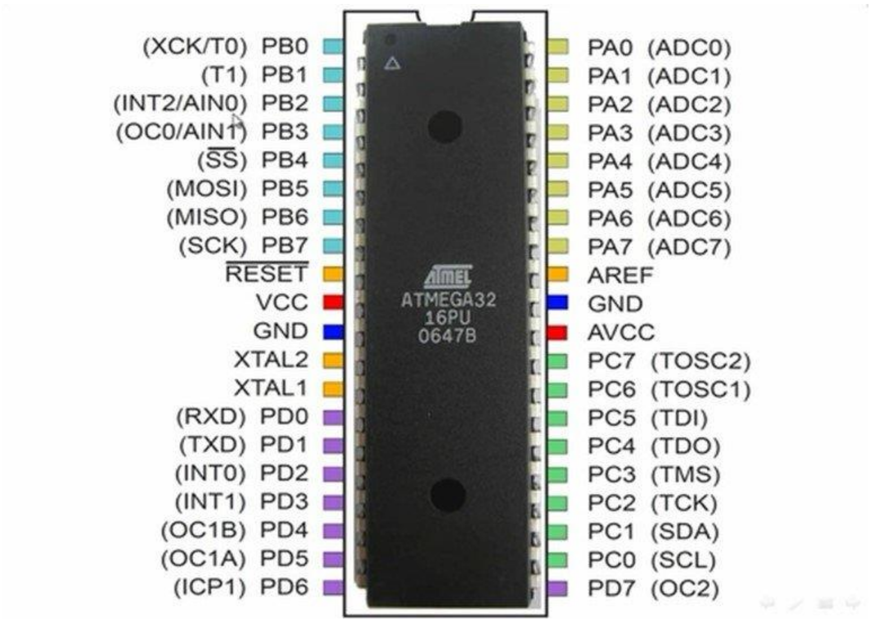


Figure 1.1 Atmega32 Microcontroller pin diagram

2. Theory

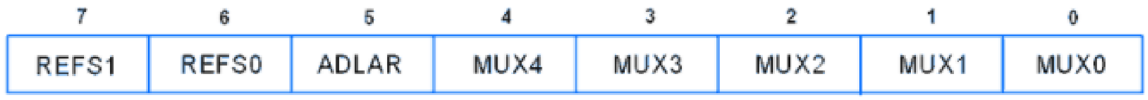
2.1 ADC in Atmega32

In the embedded systems ADC (Analog to Digital Converter) is the widely used. 10-bit ADC is inbuild in the Atmega32. That means digital output range is 0 to 1023. Atmega32 have 8 ADC pins. Pin no 33 to 40 are ADC pins.

ADC Registers

- **ADCH**: digital converted data holds in higher byte
- **ADCL**: digital converted data holds in lower byte
- **ADMUX**: ADC multiplexer selection register
- **ADCSRA**: ADC control and status register

ADMUX Register

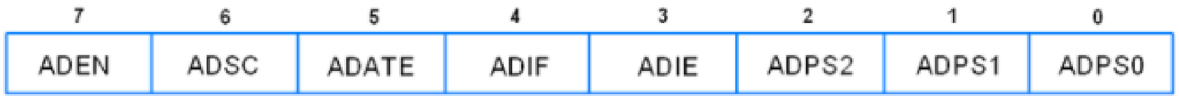


7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Figure 2.1 ADMUX Register Overview

ADMUX register is 8 bits register. Bit 7 & 6 used to select reference voltage to ADC. Bit 0 to 4 bits are Analog Channel and gain selection bits.

ADCSRA Register



7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Figure 2. 2 ADCSRA Register Overview

ADEN – used to enable or disable ADC

ADSC – used to start the conversion

ADATE – used to ADC auto trigger enable

ADIF – ADC Interrupt Flag

ADIE – used to ADC Interrupt enable

ADPS 0:2 – used to set pre-scaler

3. Methodology

3.1 Password Based Door Lock

- Using proteus software password-based door lock circuit was designed.
- Programmer's notepad software was used to write code and programme the microcontroller.

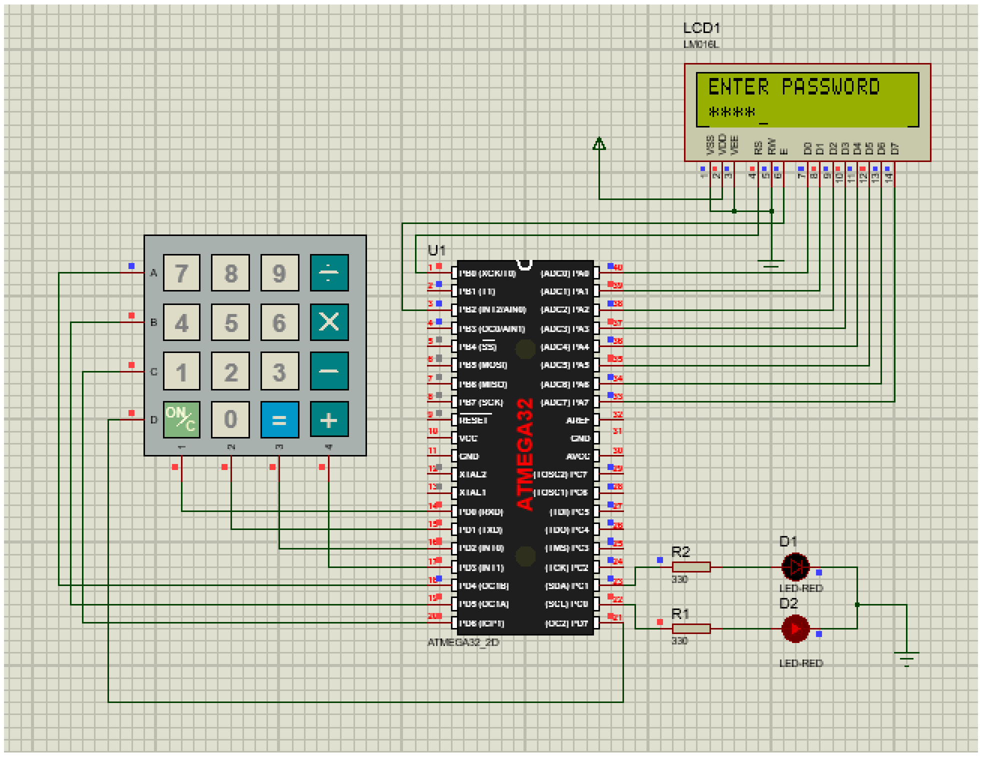


Figure 3.1. 1 Proteus Circuit Diagram of Password Based Door lock

- According to the circuit diagram LCD Display and Keypad are connected to the Atmega32 microcontroller.
- Mainly two LED bulbs are connected to the circuit. But in the circuit, those are represented as the Solenoid door lock and Buzzer.
- Solenoid door lock need 12V, therefore 5v Relay used to activate lock.

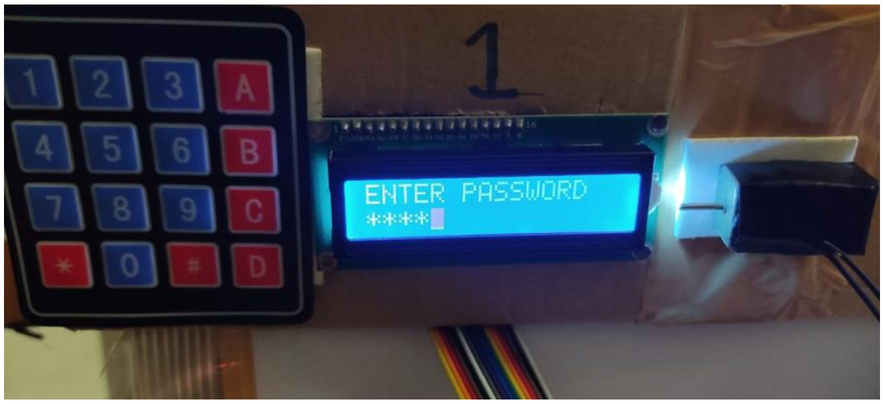


Figure 3.1. 2 View of the Password Based Door Lock

3.2 Automatic Room Temperature Control Fan

- In this section Automatic Room Temperature Control Fan is created.
- LM35 temperature sensor used to measure room temperature.
- LCD Display used to show room temperature and pre-defined temperature is **40 C^{0}**.

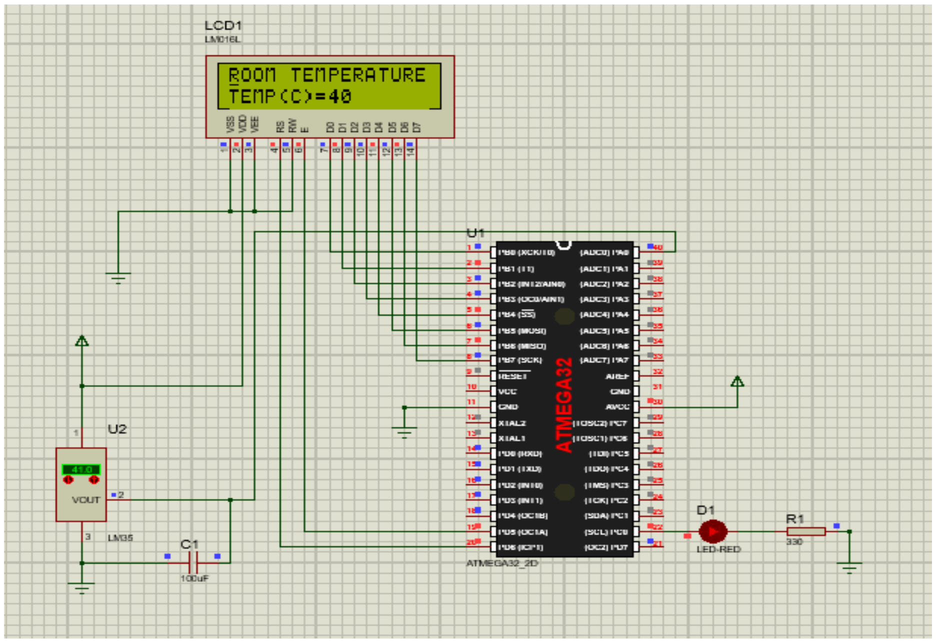


Figure 3.2. 3 Proteus Circuit Diagram of Room Temperature Control Fan

- Proteus circuit diagram is given above (***figure 3.2. 4***).

- The circuit was created according to the above diagram.
- If the room temperature become greater than **40 C^0** then the fan will be switched on and also room temperature become less than **40 C^0** then the fan will be switched off automatically.

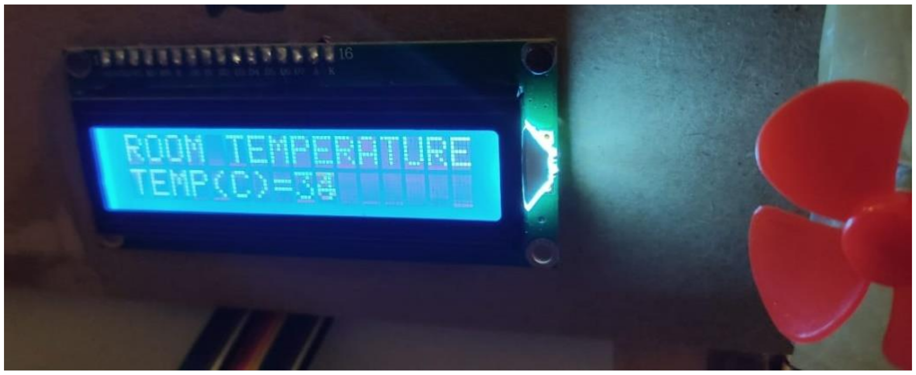


Figure 3.2. 5 View of the Room Temperature Control Fan

3.3 Remote Control & Light Sensor Bulb

- This is the final section of this Home Automation System project.
- TSOP 1738 IR sensor is used to detect IR received signal from Remote.

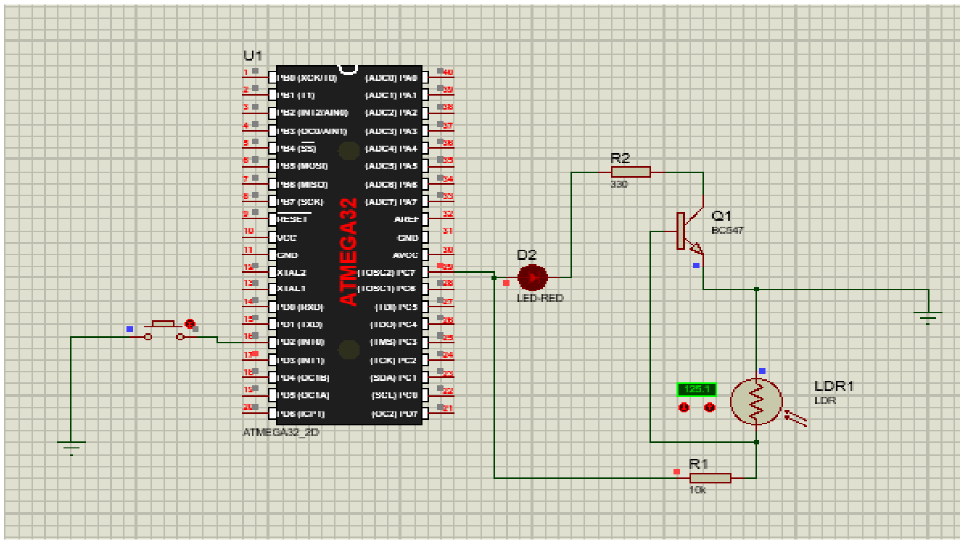


Figure 3.3. 6 Proteus Circuit Diagram of Remote Control & Light Sensor Bulb

- In this circuit LDR sensor is used to detect darkness.
- Proteus circuit is given above.
- To detect IR signal, another circuit was created.
- In this circuit output pin is connected to the Atmaga32 as an external Interrupt.
- In the proteus circuit it is shown as pull up switch.

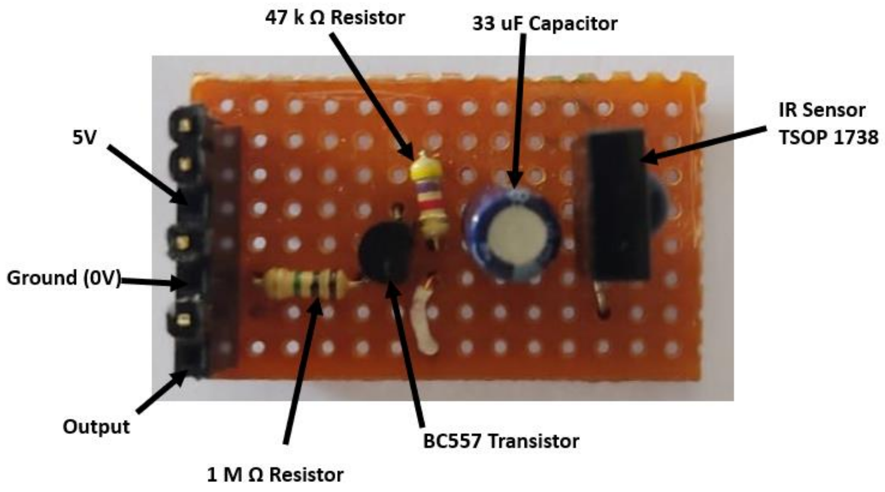


Figure 3.3. 7 Circuit of the IR Receiver

- After that LDR connected to transistor and another circuit was created. It is shown in the proteus circuit.

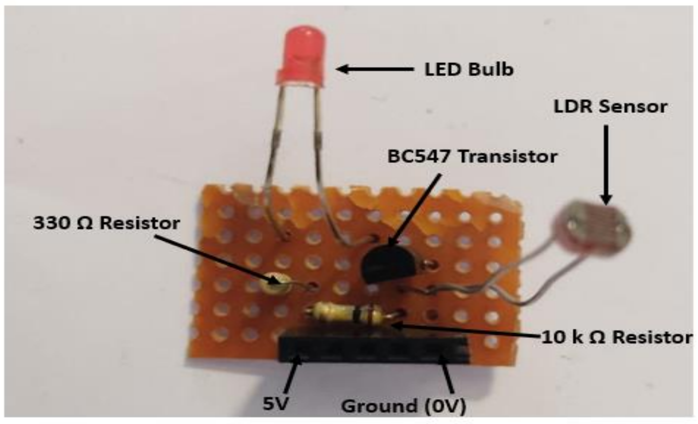


Figure 3.3. 3 Circuit of the Light Sensor

4. References

ENVIREMENTALB.COM. 2021. *REMOTE CONTROL SWITCH CIRCUIT USING 4017 - ENVIREMENTALB.COM*. [ONLINE] AVAILABLE AT: <HTTPS://ENVIREMENTALB.COM/REMOTE-CONTROL-SWITCH/> [ACCESSED 13 AUGUST 2021].

BUILDCIRCUIT.COM. 2021. *DARK/LIGHT SENSOR USING TRANSISTOR*. [ONLINE] AVAILABLE AT: <HTTPS://WWW.BUILDCIRCUIT.COM/DARKLIGHT-SENSOR-USING-TRANSISTOR/> [ACCESSED 13 AUGUST 2021].

2021. [ONLINE] AVAILABLE AT: <HTTPS://ATMEGA32-AVR.COM/PASSWORD-BASED-DOOR-LOCKING-SYSTEM/> [ACCESSED 13 AUGUST 2021].

5. Appendix

5.1 Password Based Door Lock – Code

```
#include<avr/io.h>

#define F_CPU 1000000

#include<util/delay.h>

#include <stdlib.h>


#define lcdport PORTA

#define signal PORTB

#define DDR_A DDRA

#define DDR_B DDRB

#define DDR_C DDRC

#define DDR_D DDRD

#define en 2

#define rw 1

#define rs 0


char password[1];

char lock[0];

char key;

char keycheck();

char scankey();


void lcdcmd(unsigned char cmd);


void lcdint();

unsigned char  lcd_read();
```

```
void lcddata(unsigned char data);

//void command(unsigned char cmd);

void charactersend(unsigned char character);

void String(char *characters);


int main(){

      DDR_A=0xff;

      DDR_B=0x0f;

      DDR_D=0xf0;


      DDR_C|=1<<0 && 1<<1;

      PORTC=0x01;

      _delay_ms(10);


      char key;

      lcdint();

      String ("ENTER PASSWORD");

      lcdcmd(0xC0);

  for(int i=0;i<5;i++){


            key=scankey();

            _delay_ms(1000);


            lcddata('*');

            password[i]=key;

      }

      lcdcmd(0x01);


      if((password[0]=='1')&&(password[1]=='2')&&(password[2]=='3')&&(passwor
d[3]=='4')&&(password[4]=='5')){
            String ("CORRECT PASSWORD");

            //DDRC=0xff;
            DDR_C|=1<<0 && 1<<1;
            PORTC=0x00;
            _delay_ms(5000);
            PORTC=0x00;
            lcdcmd(0x01);
            lcdcmd(0x80);

            String ("LOCK? PRESS 1");
            lcdcmd(0xC0);

            key=scankey();
            _delay_ms(1000);



            lcddata(key);
            lock[0]=key;

            if(lock[0]=='1'){
                  lcdcmd(0x01);

                  String ("LOCKED");
                  _delay_ms(1000);
                  return main();
              }
      }

      else{
```

```
            String ("WRONG PASSWORD");

            DDR_C=0xff;

            //DDRC|=1<<0 && 1<<1;

                  PORTC=0x03;

            //    PORTC=0x02;

                  _delay_ms(3000);

                  PORTC=0x00;

            return main();

      }

}

char scankey(){


      char key='a';

      while(key=='a'){

            key=keycheck();

      }

      return key;

}


char keycheck(){


      PORTD=0b11101111;

      _delay_ms(10);


      if((PIND&0b00000001)==0){

            _delay_ms(10);

            return '7';

      }


      if((PIND&0b00000010)==0){
            _delay_ms(10);
            return '8';
        }

        if((PIND&0b00000100)==0){
            _delay_ms(10);
            return '9';
        }
        if((PIND&0b00001000)==0){
                _delay_ms(10);
                return 'C';
        }

        PORTD=0b11011111;
        _delay_ms(10);
        if((PIND&0b00000001)==0){
            _delay_ms(10);
            return '4';
        }

        if((PIND&0b00000010)==0){
            _delay_ms(10);
            return '5';
        }

        if((PIND&0b00000100)==0){
            _delay_ms(10);
            return '6';

        }
```

```
if((PIND&0b00001000)==0){
      _delay_ms(10);
      return 'B';
}

PORTD=0b10111111;
_delay_ms(10);

if((PIND&0b00000001)==0){
      _delay_ms(10);
      return '1';
}

if((PIND&0b00000010)==0){
      _delay_ms(10);
      return '2';
}

if((PIND&0b00000100)==0){
      _delay_ms(10);
      return '3';
}
if((PIND&0b00001000)==0){
      _delay_ms(10);
      return'A';
}

PORTD=0b01111111;
_delay_ms(10);
```

```
        if((PIND&0b00000001)==0){
                _delay_ms(10);
                return '*';
        }
        if((PIND&0b00000010)==0){
                _delay_ms(10);
                return '0';
        }

        if((PIND&0b00000100)==0){
                _delay_ms(10);
                return '#';
        }
        if((PIND&0b00001000)==0){
                _delay_ms(10);
                return'D';
        }
        return 'a';
}

void lcdint(){
        lcdcmd(0x38);
        _delay_ms(1);

        lcdcmd(0x01);
        _delay_ms(1);

        lcdcmd(0x0F);
        _delay_ms(10);
```

```
   lcdcmd(0x82);
   _delay_ms(10);

   lcdcmd(0x89);
   _delay_ms(10);

   lcdcmd(0x01);
   _delay_ms(10);
}
void lcdcmd(unsigned char x){
      lcdport=x;
      signal=(0<<rs)|(0<<rw)|(1<<en);
      _delay_ms(1);
      signal=(0<<rs)|(0<<rw)|(0<<en);
      _delay_ms(50);

}
void lcddata(unsigned char data){

      lcdport= data;
      signal= (1<<rs)|(0<<rw)|(1<<en);
      _delay_ms(1);
      signal= (1<<rs)|(0<<rw)|(0<<en);
      _delay_ms(50);
}
unsigned char  lcd_read(){

      signal= (1<<rs)|(1<<rw)|(1<<en);
      _delay_ms(1);
      signal= (1<<rs)|(1<<rw)|(0<<en);
      _delay_ms(50);
      }
void charactersend(unsigned char data)
{

   lcdport= data;
      signal= (1<<rs)|(0<<rw)|(1<<en);
      _delay_ms(1);
      signal= (1<<rs)|(0<<rw)|(0<<en);
      _delay_ms(50);
}
void String(char *characters)
{
    while(*characters > 0)
    {
        charactersend(*characters++);
    }
}
```

5.2 Automatic Room Temperature Control Fan - Code

```
#include <avr/io.h>
#define F_CPU 1000000
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>

#define enablepoints   5
#define registerpoints 6

void Instruction(unsigned char Ins);
void characterSend(unsigned char character);
void String(char *str);
//void ADC_Read(char channel);
int main(void)
{
    DDRA = 0x00;
    DDRB = 0xFF;
    DDRC = 0x01;
    DDRD = 0xFF;
    //DDRD=0xFB;              /* PORTD as input */
    //PORTD=0b00000100;       /* Make pull up high */
    //PORTD=1<<3;


    _delay_ms(50);

    //ADMUX |=(1<<REFS0)|(1<<REFS1);
```

```
//ADCSRA |=(1<<ADEN)|(1<<ADATE)|(1<<ADPS0)|(1<<ADPS1)|(1<<ADPS2);

ADCSRA=0x87;           // Enable ADC, fr/128
ADMUX = 0x40;            // Vref: Avcc

float Temp;

char SHOWA [3];

Instruction(0x01); //Clear Screen 0x01 = 00000001
_delay_ms(50);
Instruction(0x38);
_delay_ms(50);
Instruction(0b00001111);
_delay_ms(50);

ADCSRA |=(1<<ADSC);
while(1)
{
      Temp = (ADC_Read(0)*4.88);
      Temp = (Temp/10);

    String ("ROOM TEMPERATURE");
    Instruction(0x80 + 0x40 + 0);
    String ("TEMP(C)=");
    Instruction(0x80 + 0x40 + 8);
    itoa(Temp,SHOWA,10);
    String(SHOWA);
    String ("      ");
```

```
        Instruction(0x80 + 0);

          if(Temp >= 40){
                PORTC = 0x01;
                //String ("FAN ON");
                //Instruction(0x80 + 0x40 + 0);
          }else{
                PORTC = 0x00;
    }
}

void ADC_Read(char channel)
{
    ADMUX = 0x40 | (channel & 0x07);   // set input channel to read
    ADCSRA |= (1<<ADSC);               //Start ADC
    while (!(ADCSRA & (1<<ADIF)));     //interrupt flag
    ADCSRA |= (1<<ADIF);               // Clear interrupt
    _delay_ms(1);                      // Wait a little bit
    return ADCW;                       // Return ADC word
}

void Instruction(unsigned char Ins)
{
    PORTB = Ins;
    PORTD &= ~ (1<<registerpoints);
    PORTD |= 1<<enablepoints;
    _delay_ms(20);
    PORTD &= ~1<<enablepoints;
    PORTB = 0;
```

```
}
void characterSend(unsigned char character)
{
    PORTB = character;
    PORTD |= 1<<registerpoints;
    PORTD |= 1<<enablepoints;
    _delay_ms(20);
    PORTD &= ~1<<enablepoints;
    PORTB = 0;
}
void String(char *str)
{
    while(*str > 0)
    {
        characterSend(*str++);
    }
}
```

5.3 Remote Control & Light Sensor Bulb - Code

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
char port=0;

ISR(INT0_vect)
{
//    PORTC=~PORTC;          /* Toggle PORTC */
//    _delay_ms(50);   /* Software debouncing control delay */
      port = ~port;
      if (port!=0){

      PORTC &= ~(1<<7);
      //_delay_ms(1000);
      }
      if (port==0){

      DDRC=1<<7;
      PORTC |= (1<<7);
      //_delay_ms(1000);
      }

}

int main(void)
```

```
{

      DDRD=0x00;             // PORTD input
      //PORTD=0b00000100;        // pull up high
      PORTD=1<<3;

      GICR = 1<<INT0 | 1<<INT1;        // Enable INT0
      MCUCR = 1<<ISC01 | 1<<ISC00;  // Trigger INT0 on rising edge

      sei();                 // Enable Interrupt

      while(1);
}
```