EITF 040

Digital and Analogue Project Report

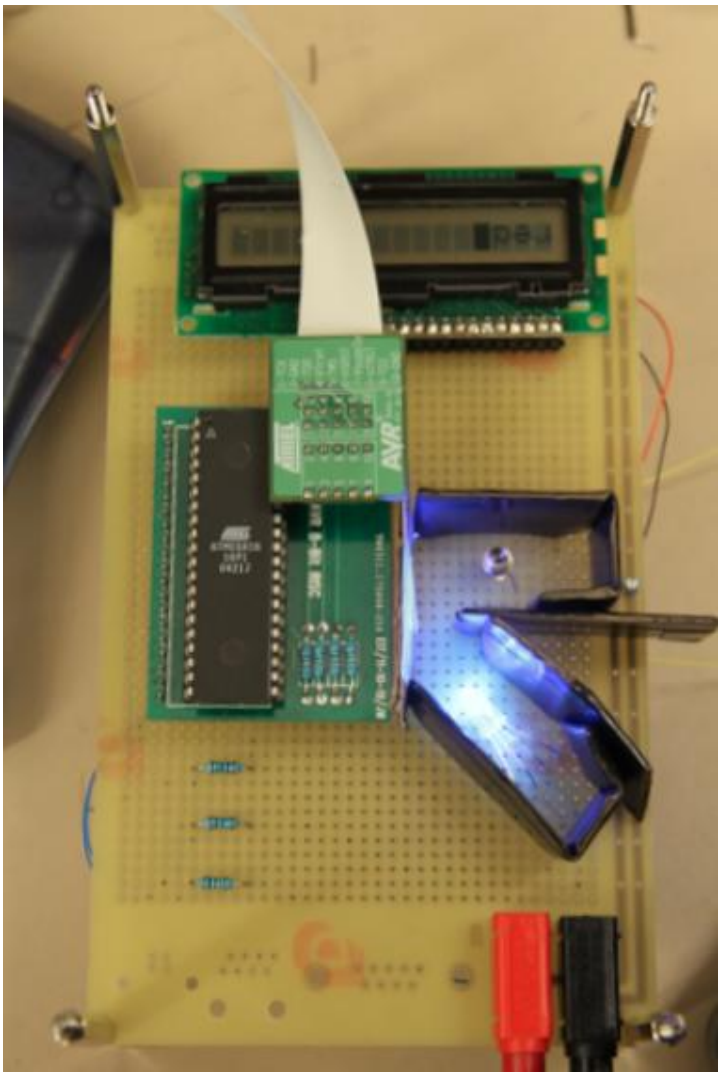
Group 6

Fida Saidani

Qinghua Liu

March, 2013

Abstract



The aim of this project is to build an electronic device that makes use of the law of light reflection, to detect the simple colours of an object. A common anode LED produces light of a certain colour in the direction of an object. The object receives light and reflects it towards a phototransistor. The colour of the object can be recognized using the voltage difference this light produced between the collector and the emitter of the phototransistor detected, and the result will be displayed on the LCD display.

Table of contents

Part I: Introduction

1.1 Aim:

The primary goal of this project is to make use of an AT Mega16 microprocessor and C programming language, to build an electronic device that is able to detect the basic colours of an object, in our case, blue and read, and the result will be displayed on a LCD display.

1.2 Electronic components:

The electronic components involved in this device are:

- AT mega 16 microprocessor
- Sharp dot matrix LCD display
- Common anode LED light
- BYP62 bi – polar junction transistor
- Resistors

1.3 Background theories:

The primary idea of the device is to make use of reflection of light. When a colour light is projected on the object, if the object is of the same colour as the projection light, the reflection will be very strong; on the other hand, if the colour of the object is different from the projection light, there will be very weak reflection. In this device, a common anode LED, which produces red, blue and green(not used in this project) colour is used to emit the light to be object, the reflection light is caught by the base of the photo transistor. Black colour cardboards are placed around the LED and the phototransistor to prevent the light from emitting directly to the transistor, and also to reduce the disturb from the natural light. The reflection light creates a potential difference between the collector and the emitter, which is read in by the microprocessor, and the final result is displayed on the LCD screen.

Part II Hardware and schematic

2.1 ATMega16 microprocessor

The AT mega 16 microprocessor is a 40 pins dual - in – line microprocessor. We make use of the 8 pins of portB(PB) and 3 pins of portD(PD) for the LCD; 3 other pins in portD are connected to the LED light. 2 pins in PortA(PA) are reserved for the photo transistor, and there is no connection for PortC, however, 4 pins in PortC are reserved for the JTag.

2.2 Sharp dot matrix LCD display

The Sharp dot matrix LCD display is a 14 – pin LCD display, with 8 – bit data bus and 3 control signals (Register set instruction/data, Read/Write, Enable). The 8-bit data busses (DB0 – DB7) are connected to PB0 – PB7. The three control signals are connected to PD0 – PD2: Register select is connected to PD0, Read/write is connected to PD1 and enable pin is connected to PD2.

2.3 LED light

The LED has 4 ends: the blue end, green end, red end and the longest pin which is to be connected to ground. The LED light that produces the three colours is connected to PD4 – PD6 of the microprocessor. The blue end is connected to PD4, green end to PD5, red end to PD6, and the other pin, is connected to ground (pin number 11 on the microprocessor). Note: this LED light is a common anode LED, not a common cathode LED as shown in the device label.

2.4 BPY 62-2 Phototransistor

The BPY 62-2 phototransistor is a high linearity bi-polar junction transistor. The collector of the phototransistor is connected to PA0 and the emitter is connected to PA1, since there is a built – in analogue to digital converter in portA. The base is not connected as it is used to receive the reflection light.

2.5 Various resistors

One 100 Ω resistor is connected between each end (except the end connected to the ground) of the LED and the microprocessor to prevent the strong current from destroying the LED.

Part III programming method

3.1. I/O method

According to our device, PortB and PortD are used for output while portA is used for input. The C command

DDRX = 0xff;

is used to set PortX as output.

3.2 methods for LED to light up and toggle

To light up / turn off the LED, the main theory is to set the corresponded pin to high / low. A delay method is used to set the frequency for the LED to change colour. To turn on the LED, the OR logic and the shift method is used. To turn off the LED, the AND logic is used. Below are the common C codes for turn on and turn off the LED.

To turn on pin number x of PORTD: PORTD|= (1<<PDx);

To turn off pin number x of PORTD: PORTD&= (0<<PDx);

3.3 Functions for the LCD display

i) LCD Initialization function

The LCD initialization function int_LCD(void) method is defined to initialize the LCD display. The function consists of four parts: function sets, display on/off control, entry mode set and clear display.

ii). LCD write function

We have defined the function LCD_write (unsigned char data) to display a single character on the display screen.

3.4 The Analogue to Digital method

The power supply voltage we are using delivers 5V. If we connect AVCC to the same supply voltage, the analogue input is capable of delivering digital values corresponding to analog inputs up to 5V. The analogue to digital converter is a 10 bit converter. That means it has 2^10=1024 digital values to convert to 0 to 5V range, 0 corresponding to 0V, 1024 to 5V.

3.5 The main function

The overall logic is our main function. First, the LCD screen is initialized and the analogue-to-digital converter (ADC) is also initialized. After that, the red light is turned on and the delay time is set, for the ADC to capture the voltage values from the phototransistors. The for-loop is used to calculate the average of 50 different values to receive a more trustworthy result. Secondly, red light is turned off and blue light is turned

on. The for-loop was gone through again to calculate the value. In the end, the two results are compared against a certain voltage threshold, if both of the results are below the threshold, the screen will display “not recognized”. If either or both of the value are above the threshold, the screen will display the name of the colour which is corresponded to the higher value, e.g. red.

Part IV Result and discussion

4.1 Testing result

In our testing process, we connected the multi-meter to the collector and the emitter of the phototransistor and collected the following result.

Colour of light &object	No light	Red light Red object	Blue light Red object	Red light Blue object	Blue light Blue object
Potential difference(mV)	0 -3	7 -10	2-3	2-4	7-11

From the table, we can clearly tell that when the colour of the object is the same as the colour of the projected light, there is a significant increase in the potential difference of the emitter and the collector, and the potential differences of both cases (red on red, blue on blue) are above 5mV. On the other hand, when the colours of the object and the projected light are different, the increase in potential difference is not significant. In our algorithm, the two results are first compared with 5 mV, if both of them are below 5mV, we classify this category as "not recognized", except this case, we choose the higher value to decide the colour. According to the table above, this algorithm should be able to decide whether the object is blue or red. However, when the object is not red or blue, the algorithm is not accurate enough.

4.2 Discussion and future development

This device is able to detect the simple colour of the object; however, there are a lot of limitations:

i). As mentioned in the result part, the device can accurately detect the colour of the object only if the colour of the object is blue or red. When the colour of the object is some colour that is close to blue or red, e.g. green or orange, the device might decide that this object is blue and red. To fix this problem, the LED lights that produce other colours can be used. What's more, an amplifier should be used to amplify the potential difference across the phototransistor so that the increase in potential difference can be more significant.

ii). The intensity of the light produced by the LED is very low, especially compared to nature light. In our project, we have built "walls" to block the natural light, also to prevent the LED from lighting directly to the phototransistor. However, it is very difficult to make sure that the walls can stand on the device stably. A more proper light - blocking device might be developed in the future.

iii). Since the light blocking arrangement cannot guarantee an entire isolation from the natural light, the accuracy of the device might be easily influenced by the brightness of the room. If possible, this device can be developed into a device that does not require light-blocking arrangement by using multiple LEDs or stronger power lights to produce stronger lights, whose intensity is comparable to the natural light. Also, the sensitivity of the microprocessor might be improved, or an amplifier can be used.

Appendix

A.1. C -programme code

```
/*
Analogue and digital projects

Fida Saidani

Qinghua Liu

ATmega168

color detector

*/


#include <stdint.h>
#include <stdlib.h>
#include <avr/interrupt.h>
#include <avr/io.h>
#include <util/delay.h>
#include "lcd.h"
#ifndef F_CPU
#define F_CPU 3686400
#endif


uint16_t duty_cycle;

/* IRQ routine at 500hz */

ISR(TIMER1_OVF_vect) {

   /* if the duty is at 95% the restart from 5% */
   if (duty_cycle > 950)
      duty_cycle = 50;

   /*
                increment the duty cycle by 1 step at the time
                1 step is 0.1% of TOP value ICR1 = 1000 steps
                in this way 100% is completed in 2 seconds
                */

                duty_cycle++;

   /* set the level where the output pin OC1A will toggle */
```

```
    OCR1A = duty_cycle;

    /* set the level where the output pin OC1B will toggle
      this will be the inverse of OC1A level */
    //OCR1B = 1000 - duty_cycle;
}

void counter_setup(void)
{
    /* Pin OC1A & B set to output */
    DDRD |= _BV(PD4) | _BV(PD5);

    /*
                Clear OC1A & B on compare match, set it at BOTTOM.
                Waveform generation mode on 14

                In non-inverting Compare Output mode, the Output Compare (OC1x) is
                cleared on the compare match between TCNT1 and OCR1x, and set at
BOTTOM.

                In inverting Compare Output mode output is set on compare match and
                cleared at BOTTOM

                */

    TCCR1A = _BV(COM1A1) | _BV(WGM11) ; //| _BV(COM1B1)
    TCCR1B = _BV(WGM13) | _BV(WGM12);

    /* TOP value*/
    ICR1 = 1000;

    /* start with a duty of 50% */
    OCR1A = 500;
    //OCR1B = 500;

    /* enable interrupt on timer overflow */
    TIMSK = _BV(TOIE1);
}

void init_LCD(void);
void LCD_write(unsigned char data);
void adc_init();
int adc_read(uint8_t ch);
void  int_to_char(uint16_t data);


int main()
{
    DDRB=0xff;                          // making LCD_DATA port as output port
    DDRD=0xff;                          // making signal as out put
```

```
 init_LCD();                              // initialization of LCD, call function
            adc_init();                   //initialize analog to digital converter, call function
 _delay_ms(50);          // delay of 50 milli seconds


int time= 5000;

            //DDRD=0xff;          // set PortD for output
            //DDRB=0xff;                                          // set PortB for output


unsigned char i;
int ctr,sum1=0, sum2=0;
uint16_t adc_result0,adc_result1,adc_result[10];



            PORTD |= (1<<PD6);                  //red light on

            _delay_ms(time);                    //wait delay

            //get 15 results
            for (ctr=1; ctr<=50; ctr++)
{
            _delay_ms(50);
            adc_result0 = adc_read(0);      // read adc value at PA0
            adc_result1 = adc_read(1);      // read adc value at PA1
            sum1= sum1+ abs(adc_result0-adc_result1);


}
            sum1=sum1/ctr;

            //int_to_char(adc_result);

            PORTD &= ~(1<<PD6);                 //red light off
            PORTD |= (1<<PD4);                  //blue light on
            _delay_ms(time);                    //wait delay

            //get 15 results
            for (ctr=1; ctr<=50; ctr++)
{
            _delay_ms(50);
            adc_result0 = adc_read(0);      // read adc value at PA0
            adc_result1 = adc_read(1);      // read adc value at PA1
            sum2= sum2+ abs(adc_result0-adc_result1);


}
            sum2=sum2/ctr;
```

```
                if (sum1<1 && sum2<1)
                {
                LCD_write('n');
                LCD_write('o');
                LCD_write('t');
                LCD_write(' ');
                LCD_write('r');
                LCD_write('e');
                LCD_write('c');
                LCD_write('o');
                LCD_write('g');
                LCD_write('n');
                LCD_write('i');
                LCD_write('s');
                LCD_write('e');
                LCD_write('d');

                }
                else
                {
                if (sum1<sum2)
                {
                LCD_write('b');
                LCD_write('l');
                LCD_write('u');
                LCD_write('e');
                }
                else
                {
                LCD_write('r');
                LCD_write('e');
                LCD_write('d');
                }
}
                return 1;

}

//function to initialize lcd

void init_LCD(void)

{
DDRB=0xff;                  //set port B as output, data register
DDRD= DDRD| 0b00000111;             //set pins 0,1 and 2 from port D to output

//first step: function set

PORTD= PORTD & 0b11111100;          //set register select (data/instruction)
```

```
                // and read/write to 0

PORTD= PORTD | 0b00000100;          //set enable to 1

PORTB= 0b00110000;                              //send function set to port B
_delay_ms(50);
PORTD= PORTD & 0b11111011;          //set enable as 0
_delay_ms(50);                                                  //wait
PORTD= PORTD | 0b00000100;          //set enable as 1
_delay_ms(50);

//Display On/Off Control:
PORTB= 0b00000011;
_delay_ms(50);
PORTD= PORTD & 0b11111011;          //set enable as 0
_delay_ms(50);                                                  //wait
PORTD= PORTD | 0b00000100;          //set enable as 1
_delay_ms(50);


//Entry Mode Set:
PORTB= 0b00001111;
_delay_ms(50);
PORTD= PORTD & 0b11111011;          //set enable as 0
_delay_ms(50);                                                  //wait
PORTD= PORTD | 0b00000100;          //set enable as 1
_delay_ms(50);


//clear display
PORTB= 0b00000001;
_delay_ms(50);
PORTD= PORTD & 0b11111011;          //set enable as 0
_delay_ms(50);                                                  //wait
PORTD= PORTD | 0b00000100;          //set enable as 1
_delay_ms(50);


   return 1;


}


//function to write on LCD

void LCD_write(unsigned char data)
{
   PORTD= PORTD |0b00000101;                    //set enable to 1 and I/D to data
```

```
                _delay_ms(50);
                PORTB=data;
                                //set the data port to actual data value
                _delay_ms(50);
                //wait
                PORTD= PORTD & 0b11111011;        //set enable as 0
                _delay_ms(50);                                                  //wait
                PORTD= PORTD | 0b00000100;        //set enable as 1
                _delay_ms(50);

                return 1;
                }

//function to initialize ADC

void adc_init()
{
   // AREF = AVcc

   ADMUX = (1<<REFS0);


   // ADC Enable and prescaler of 128

   // 16000000/128 = 125000

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

}

//function to read value from ADC
int adc_read(uint8_t ch)
{
   // select the corresponding channel 0~7
   // ANDing with '7' will always keep the value
   // of 'ch' between 0 and 7
   ch &= 0b00000111;  // AND operation with 7
   ADMUX = (ADMUX & 0xF8)|ch;     // clears the bottom 3 bits before ORing

   // start single conversion
   // write '1' to ADSC
   ADCSRA |= (1<<ADSC);

   // wait for conversion to complete
   // ADSC becomes '0' again
   // till then, run loop continuously
   while(ADCSRA & (1<<ADSC));
    return
}
```

A.2. Schematic