A-system-consisted-of-LM35sensor-DCmotor-7segment

Final Project of Microprocessor and Computer Structure Lab

Manoosh Samiei

In this project, a system consisted of AT mega 32, LM35 sensor, 7 segments, and DC motor is constructed. The 3 seven segments show the temperature of the environment, measured by LM35 sensor, up to 1 decimal point. The rotational speed of DC motor varies according to the temperature of the environment. Its maximum speed is in 90 centigrade and minimum speed is in 0 centigrade. The computer shows the temperature, time, and date, received from the microcontroller using USART communication. It is also possible to set the speed of the DC motor manually, and show a desired temperature manually on the segments, using the computer's serial communication.

This system has separated settings for time, date, motor speed, and segments. The user can change all these settings on the ATmega32 microcontroller, using USART communication.

To display the temperature from the lm35 sensor, we use analog-to-digital conversion of micro-controller, which can equate the analog voltage of a pin with digital binary number. The ADC feature of atmega32 is available on port A. Therefore, the temperature sensor is connected to the pin A0. The following code is used to display sensor temperature on segments:

```
 n=(float)read_adc(0)/4;   //n is a float variable

c=read_adc(0)/4;   //c is a character variable

if(l==0){

T=(n*10);}

temp[0]=T%10;

temp[1]=(T/10)%10;

temp[2]=(T/100)%10;

pd=0x01;

for(i=0;i<3;i++){

PORTB=~pd;

PORTC=display[temp[i]];

if(i==1){PORTC+=0x80;}

pd=pd<<1;

delay_ms(5); }
```

In the first line of the above code, read_adc function is called, which puts the temperature of lm35 sensor in variable "n". The output voltage of the sensor is 4 times the temperature; therefore, by dividing the voltage by 4, the temperature is achieved. The value of the float type variable "n" is assigned to a character type variable named "c" to be used in the nest sections of the code. But to show temperature on the segments up to 1 decimal point, the float variable "n" is used. To show this float number on the segments, at first, I multiplied it by 10 and then show each of those digits separately on each of the segments. To have a float display, the dp point of the segment should also be turned on. There is also a conditional expression before multiplying n by 10. This condition checks whether the variable l is equal to 0 or not. This condition is used when we want to show our desired number on the segments regardless of the environment temperature. At the beginning of the program l=0. When we enter the segment's settings in USART serial monitor, this value is incremented by 1 unit, and therefore it will no longer be 0 to enter this part of the program, and only shows our entered number in the USART serial monitor.

```
if(l==0){

T=(n*10);}

temp[0]=T%10;

temp[1]=(T/10)%10;

temp[2]=(T/100)%10;
```

"temp" is a character type array with three indexes. IT is used to divide the numbers based on their place-value. Then for displaying the number on segments I exploited the Refreshing method. I used one multiplexed seven segment, containing three segments. Each segment should be turned on with their appropriate digit with 5 milliseconds delay from the previous one; this delay is not recognizable for human eyes, and therefore it seems like all segments are always on. For a common cathode segment, grounding each of the 3 connectors of segments turns on the respective segment. The data is given to the pins a, b, c, d, e, f, and g. The following code clarifies the refreshing method.

```
pd=0x01;

for(i=0;i<3;i++){

PORTB=~pd;

PORTC=display[temp[i]];

if(i==1){PORTC+=0x80;}

pd=pd<<1;

delay_ms(5); }
```

Instead of shifting 0 for the common cathode, I shifted 1 and then inverted the result. The display array is defined globally at the beginning of the program, and is used to convert the data to an appropriate form for displaying on the segments.

The code :   if(i==1){PORTC+=0x80;} is used to turn on the dp point on the second segment from the left. i=1 is the index of the array of the second segment's data.

Adjustment rotational speed of the DC motor is done by PWM pulse. The duty cycle of a pulse is the fraction of one period in which a signal is active (here equal to 1). The more the duty cycle of a pulse, the more the speed of the motor, for applying voltage for a longer time. To generate this PWM pulse, I used Timer 1 (working with 16 bits) since it has the capability of defining the maximum value for it, by giving the top value to the ICR1 register manually. I gave the number 90 (the temperature in which the motor speed is the maximum) to the lower 8 bits of register ICR1L=0x5A. Now it is possible to adjust the duty cycle of the pulse by using OCR1 register. Thus, I stored the temperature of the sensor in a character variable named "c" and assigned value of c to the bits if lower value in OCR1 register: if(k==0){ OCR1AL=c ; } the if condition is used to make it possible for the user to give the motor a desired rotational speed. In the USART communication the user can set the speed value manually, this way value of k is incremented by 1 unit, preventing it to enter this part of the program. It is also possible for the user to choose again to control the motor again by the temperature.

In the USART section of the program, the user can choose to display a desired 3-digit number on the segments, rotate the motor with a desired speed, and also give the initial value of time and date. For displaying USART communication, I used the virtual machine option in Proteus software. At the beginning of the program, the user is asked to press key “v” to enter motor setting, press key “t” to enter time and date setting, and press key “g” to enter 7 segment setting. The code of this part is as follows:

```
if(p==0){

    putsf("to enter motors' settings press v");

    putchar('\n\r');

    putsf("to enter time settings press t");

    putchar('\n\r');

    putsf("to enter 7segments's settings press g");

    putchar('\n\r');

    putchar('\n\r');

    p++;}
```

The conditional expression p==0, is used to prevent the repetition of the USART massage on virtual machine. If the user press key “v” she will enter the motor setting section, and will be asked if she wants to enter an arbitrary speed for the motor’s rotation, or wants it to be controlled by temperature. For the former, the key “a”, and for the latter the key “b” must be pressed. IF the user chooses to enter the speed manually, a 2-digit number is received from her and is directly placed in ocr1 register. If the user chooses to control the speed by temperature, as previously mentioned, the variable c which contains the temperature is placed in register ocr1. The following code clarifies this section of program:

```
if(key=='v'){

    key=getchar();

    putsf("motor's settings:");

    putchar('\n\r');

    putsf("set motors velocity or control it by temprature? press a or b");

    putchar('\n\r');

    key=getchar();

    if(key=='a'){

    putsf("velocity:");

    A=getchar();

    B=getchar();

    A=A-48;

    B=B-48;

    v=(A*10)+B;

    printf("%d",v);
    putchar('\n\r');

    OCR1AH=0x00;

    OCR1AL=v ;}

    if(key=='b'){

    OCR1AH=0x00;

    OCR1AL=c ;

    putsf("temperature control ok");

    putchar('\n\r');}

    k++;

    }
```

To receive the number from USART, a value of 48 should be subtracted from the received number, to equalize the character type to an integer type. Then, the digits are multiplied by their place-value.

If user presses “t” she will enter the section of initial settings for time and date. At first, she is asked to enter the initial value for year, month, day, hour, minute and second. (if she enters a meaningless data such as the number 13 for moths, she will be asked to enter the appropriate number again.) Then, she is asked if she likes to display the time and date on the screen. (This part of the code is not put here because of being too lengthy.)

Calculating time and date should be done repeatedly. For this purpose, the variable “second” is incremented by one after each delay of 1 second. However, since the refreshing part of the code is placed in the “while(1)” section and thus is sensitive to delay, instead of a 1-second delay, I used Timer 0. The frequency of timer is set on 0.997 khz; thus, after 1000 milliseconds delay, it should have counted 977 times (977 clock pulses). The calculation is as follows: 1000/(1/0.977)=1000*0.977=977

Timer 0 works with 8-bit data, and in each overflow it counts 256 times. Therefore, after 977 times of counting (3x256=768), 3 overflows are occurred, and (977-768=209) 209 clocks of the 4th run is also counted. Therefore, if the timer meets the value of 209 for 4 times, it means that it has counted for 977 times, and 1 second is passed. To start again from zero in the next runs, the current value of the timer in TCNT0 register is set to zero, in order to not include the rest of counting from 209 to 256.

For a better function of timer, the compare match interrupt is used, and the value of ocr0 is set to 209 or its hexadecimal equivalence: D1. Then in the interrupt function pf timer, a variable named “comparematch” is incremented by 1 unit. It means that each time the interrupt occurs; one unit is added to the variable. Whenever “comparematch” is equal to 4, 4 interrupts have occurred, and 1 second is passed. Thus, the variable “second” is incremented by one, and whenever it equals 60, the minute is incremented by 1. The same process is used for hour, month, day, and year. However, since displaying date and time spends a lot of time in the interrupt function, the program lags behind and does not execute the “while(1)” part of the code. Thus the segments blink. I wrote the program in a way that whenever the user wants to see the date and time, she should press the “s” button. The current date and time calculated from the initial values entered by user is displayed on the Virtual machine. It is also possible to enter another initial value for time and date by pressing “t” button. This part of the code is placed in the timer 0 interrupt function as follows:

```
interrupt [TIM0_COMP] void timer0_comp_isr(void) {

     comparematch++;

     if(comparematch==4){
```

```
TCNT0=0X00;
 second++;
 comparematch=0;
 if(key=='s'){
 key=getchar();
 if(second==60){second=0; minute++;}
 if(minute==60){minute=0; hour++;}
 if(hour==24){hour=0;day++;}
 if(day==31){day=0; mounth++;}
 if(mounth==12){mounth=0; year++;}
 //show temp
 printf("temprature is: %d" , c);
 putchar('\n\r');


 //show Date
  putchar('\n\r');
  putchar('\n\r');
 putsf("Date:");
 printf("year=%d",year);
 putchar('/');
 printf("month=%d",mounth);
 putchar('/');
 printf("day=%d",day);
 putchar('\n\r');
 //show Time
 putsf("Time:");
 printf("hour=%d",hour);
 putchar(':');
 printf("minute=%d",minute);
 putchar(':');
 printf("second=%d",second);
 putchar('\n\r');
 putchar('\n\r');
```

```
    key='h';   }}}
```

When user presses the “g” button, she will enter the segment setting section, and is asked to if she likes to display an arbitrary number on the segments or if she wants to display the temperature. For the former she should press the “a” button, and for the latter she should press the “b” button. If she presses the “a” key, she will be asked to enter the number she likes to display. Then after subtracting a value of 48 from the number and multiplying by its place value, the result will be stored in variable T. This variable is multiplied by 10, to be adjusted to the float format of displaying in the program. This variable in the beginning of “while(1)” loop, is placed in the “temp” array and enters the refreshing loop of the segment. If user presses “b” button, the temperature “n” is received from the sensor, multiplied by 10, and placed in variable T. In the end, the variable l, is incremented by 1 to prevent entering the first section of the while, which determined the segment’s data. This part of the code is as follows:

```
if(key=='g'){

key=getchar();

putsf("7segment's settings:");

putchar('\n\r');

putsf("Would you like to show an arbitrary number or the sensors temperature on 7 segment ? press a or b");

putchar('\n\r');

key=getchar();

if(key=='a'){

putsf("enter the number");

A=getchar();

B=getchar();

A=A-48;

B=B-48;

T=(A*10)+B;

printf("%d",T);

putchar('\n\r');

T=T*10;        }

if(key=='b'){

T=(n*10);

putsf("sensor temperature ok");

putchar('\n\r');}

l++;}
```