Embedded systems project

ITI summer training

Introduced by

Ali Essam Hassan Shams Eldin

Mohamed Wael Mohamed Elsayed

Ahmed Adel Mohamed Ahmed

Project overview

The project consists of 3 Micro controllers:

Hardware:

1- HMI ECU: responsible for the user interface, it's connected to keypad, LCD, change mode button (INT0) and Motor control button (Any DIO pin)

2-Motor control ECU: connected to a DC motor, fan and a temperature sensor (LM35)

3-Lighting ECU: connected to 4 leds which represent the car light system. The 4 leds represent car front left light, car front right light, car rear left light and car rear right light.

COMMUNICATION:

1- HMI ECU and Motor Control ECU are connected through UART.

2- HMI ECU and Lighting ECU are connected through I2C.

Required functionality:

1- Once the simulation starts the user should choose which ECU he/she wants to control. The LCD will show two options in two lines

1: Control the motor.

2: Control the lighting.

If mode 1 is selected, the user will be asked, through a message on LCD, to enter the motor speed he wants the motor to operate on (from 0 to 100).

After writing the input speed, the motor won't start until the motor control button is pressed. If the button is released the motor shall stop and if pressed again the motor shall run again with the last input speed.

The LCD should say the motor state (running, stopped)

Required functionality(CONTD)

If mode 2 is pressed, the user is asked to select which light he wants to turn on. A two line message should appear on the LCD to let the user choose the lighting option.

Line one of the message is(11: FR, 12:FL, 15:F)

Line two of the message is(13:RR, 14:RL, 16: R)

Required functionality(CONTD)

Motor cooling:

The motor temperature is measured using the LM35 sensor, if the temperature became > 70 do the following:

- Turn the fan on for 20 sec.

- If the temperature becomes< 70 within the 20 sec the fan should be turned off.

- If the temperature remained > 70 even after the 20 sec the following condition should happen:

Turn the motor off

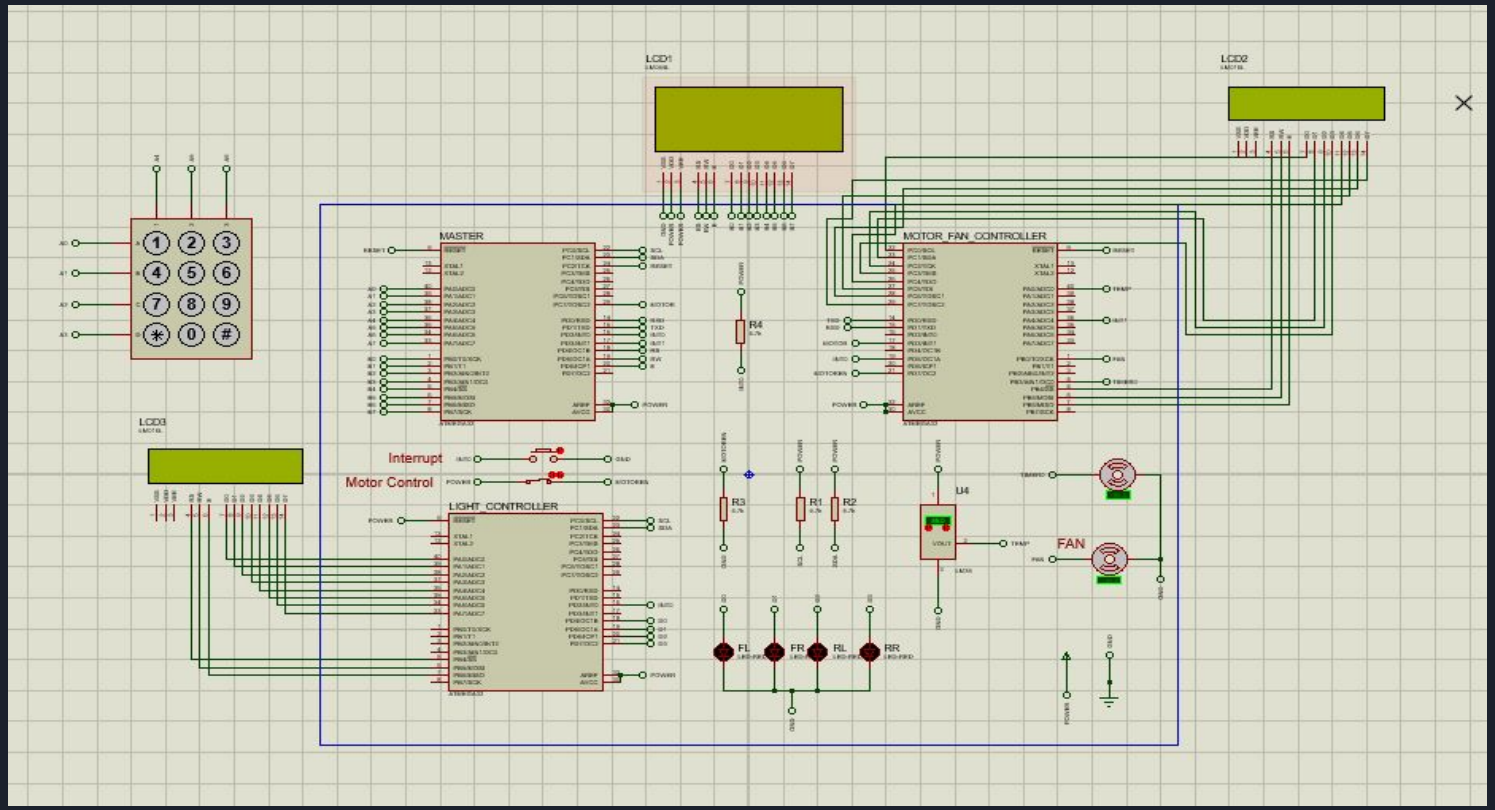
Turn all the lights off

Show "System failure" on LCD.

wait for 30 sec until you can turn the system on again

When system turns on again, repeat the first step of the project (let user choose the mood)

Proteus simulation



Code : HMI ECU

```c
#include "REG_DEF.h"
#include "BIT_MATH.h"
#include "STD_TYPES.h"
#include "lcd.h"
#include "keypad.h"
#include "uart.h"
#include "avr/interrupt.h"
#include "I2C_Master.h"

#define I_BIT       7
#define MCUCR_ISC01  1
#define MCUCR_ISC00  0
#define GICR_INT0  6

s32 LCD_ReadInput();

u8 current_screen = 1;
u8 interrupt = 0;
int main(void)
{
    /*The reset button using interrupt*/

    /*************************interrupts*************************/
    /*INT0 input  */
    GICR = 1<<INT0 | 1<<INT1;       /* Enable INT0 and INT1*/
    MCUCR = 1<<ISC01 | 1<<ISC00;  /* Trigger INT0 on rising edge */
    //SET_BIT(SREG,I_BIT);     // Doesn't work as one of the interrupts will work and not the two of them so we use sei
    sei();          /* Enable Global Interrupt ********From Google */

    CLR_BIT(DDRD,PD2);
    SET_BIT(PIND,PD2);

    CLR_BIT(DDRD,PD3);
    SET_BIT(PIND,PD3);
    /************************************************************/

```

Code : HMI ECU(CONTD)

```
/**/
LCD_init();
UART_init();
TWI_init_master();

SET_BIT(PORTC,PC2);
SET_BIT(DDRC,PC2);


SET_BIT(DDRC,PC7);
CLR_BIT(PORTC,PC7);


s8 key;
s16 speed = 0;
s16 led = 0;
_delay_ms(200); //if system failure happen more than one time to avoid displaying control lighting
if(interrupt == 0 && GET_BIT(PIND,PD2) == 1)
{
    LCD_goToRowColumn(0,0);
    LCD_displayString("1 Control motor");
    LCD_goToRowColumn(1,0);
    LCD_displayString("2 Control Lighting");
}
while(1)
{
    if (current_screen == 1 && interrupt == 1 && GET_BIT(PIND,PD2) == 1)
    {
        LCD_clearScreen();
        LCD_goToRowColumn(0,0);
        LCD_displayString("1 Control motor");
        LCD_goToRowColumn(1,0);
        LCD_displayString("2 Control Lighting");
        interrupt = 0;
    }
```

Code : HMI ECU(CONTD)

```
if(GET_BIT(PIND,PD2) == 0)
{
    LCD_goToRowColumn20(0,0);
    LCD_displayString("System Failure");
    UART_sendByte(0);
    SET_BIT(PORTC,PC7);
    _delay_ms(250);
    CLR_BIT(PORTC,PC7);
}
if(interrupt == 0)
{
    key = KeyPad_getPressedKey();
}
if(key == 1 || ((key == -1 && current_screen == 2) && GET_BIT(PIND,PD2) == 1))
{
    LCD_clearScreen();
    LCD_goToRowColumn(0,0);
    LCD_displayString("Speed :");
    current_screen = 2;
    interrupt = 0;
    speed = LCD_ReadInput();
    if(speed != -1)
    {
        LCD_displayString(" %");
        LCD_goToRowColumn20(1,0);
        LCD_displayString("to go back enter ");
        LCD_goToRowColumn20(2,0);
        LCD_displayString("1 to control motor");
        LCD_goToRowColumn20(3,0);
        LCD_displayString("2 to control light");
        /*Control the speed using UART*/
        UART_sendByte(speed);
        SET_BIT(PORTC,PC7);
        _delay_ms(250);
        CLR_BIT(PORTC,PC7);
    }

}
```

Code : HMI ECU(CONTD)

```
else if(key == 2 || ((interrupt == 1 && current_screen == 3) && GET_BIT(PIND,PD2) == 1))
{
    LCD_clearScreen();
    LCD_goToRowColumn(0,0);
    LCD_displayString("FL:11, FR:12, F:15");
    LCD_goToRowColumn(1,0);
    LCD_displayString("RL:13, RR:14, R:16");
    current_screen = 3;
    interrupt = 0;
    LCD_goToRowColumn20(2,0);
    led = LCD_ReadInput();
    if(led != -1)
    {
        if(led < 11 || led > 16)
        {
            LCD_clearScreen();
            LCD_displayString("Invalid option ");
            LCD_goToRowColumn20(1,0);
            LCD_displayString("to go back enter ");
            LCD_goToRowColumn20(2,0);
            LCD_displayString("1 to control motor");
            LCD_goToRowColumn20(3,0);
            LCD_displayString("2 to control light");
        }
        else
        {
            LCD_clearScreen();
            LCD_displayString("option ");
            LCD_intgerToString(led);
            LCD_goToRowColumn20(1,0);
            LCD_displayString("to go back enter ");
            LCD_goToRowColumn20(2,0);
            LCD_displayString("1 to control motor");
            LCD_goToRowColumn20(3,0);
            LCD_displayString("2 to control light");
```

Code : HMI ECU(CONTD)

```
            // Function to initialize TWI
            TWI_start(); // Function to send start condition
            TWI_write_address(address+write); // Function to write address and data direction bit(write) on SDA
            TWI_write_data(led);      // Function to write data in slave
            TWI_stop(); // Function to send stop condition
            _delay_ms(10); // Delay of 10 mili second
        }

    }

  }
}
return 0;
}
```

Code : HMI ECU(CONTD)

```
s32 LCD_ReadInput(){
    s8 Local_u8PressedKey = 0xFF;
    s32 Local_input = 0;
    u8 counter = 0;
    while(1){
        Local_u8PressedKey = KeyPad_getPressedKey();
        if(Local_u8PressedKey != 0xFF){
            if(Local_u8PressedKey == 'k' && counter != 0)
            {
                break;
            }
            else if(Local_u8PressedKey == -1)
            {
                return -1;
            }
            else if(Local_u8PressedKey != 'k' && Local_u8PressedKey != 'd')
            {
                if(counter != 0)
                {
                    Local_input *= 10;
                    Local_input += Local_u8PressedKey;
                }
                else
                {
                    Local_input = Local_u8PressedKey;
                }
                counter++;
                LCD_intgerToString(Local_u8PressedKey);
            }
            else if(counter != 0 && Local_u8PressedKey == 'd')
            {
                LCD_sendCommand(0x10);
                LCD_displayCharacter(' ');
                LCD_sendCommand(0x10);
                Local_input /= 10;
                counter--;
            }
```

Code : HMI ECU(CONTD)

```
            }
        }
    }
    return Local_input;
}

ISR(INT0_vect)
{
    CLR_BIT(PORTC,PC2);
}

ISR(INT1_vect)
{
    LCD_clearScreen();
    while(GET_BIT(PIND,PD3) == 0)
    {
        LCD_goToRowColumn20(0,0);
        LCD_displayString("Overheating, ");
        LCD_goToRowColumn20(1,0);
        LCD_displayString("Cooling on");
        interrupt = 1;
    }
    LCD_clearScreen();

}
```

Code : Motor control ECU

```c
#include "REG_DEF.h"
#include "BIT_MATH.h"
#include "STD_TYPES.h"
#include "adc.h"
#include "Timer0.h"
#include "Timer1.h"
#include "util/delay.h"
#include "uart.h"
#include "lcd.h"
#include "avr/interrupt.h"

#define Reading_Channel 0
u8 x = 0, y = 0, z = 0;
u8 flag = 0;
u8 fan = 0;

int main(void)
{
	CLR_BIT(DDRD,PD7);
	SET_BIT(PIND,PD7);
	/*Fan PIn*/
	SET_BIT(DDRB,PB0);
	CLR_BIT(PORTB,PB0);

	/*Failure Pin*/
	SET_BIT(DDRD,PD5);
	SET_BIT(PORTD,PD5);
	/*Overheating Pin*/
	SET_BIT(DDRA,PA4);
	SET_BIT(PORTA,PA4);
	////////////////////////////////////
	ADC_init();
	LCD_init();
	timer0_init_PWM();
	timer1_init_CtcMode();
	UART_init();

	CLR_BIT(DDRD,PD3);
```

Code : Motor control ECU(CONTD)

```
u16 Temp_Reading = 0;
u8 speed;

while(1)
{
    /*Read the temprature using ADC*/
    Temp_Reading = ADC_ReadChannel(Reading_Channel);
    Temp_Reading = ((float)Temp_Reading * 500)/ 1023;
    LCD_goToRowColumn(0,0);
    LCD_displayString("Temp ");
    LCD_intgerToString(Temp_Reading);

    if(fan == 1)
    {
        timer0_set_dutyCycle(0);
    }
    /*Check the temperature to control the fan*/
    if(Temp_Reading > 70 && fan == 0)
    {
        SET_BIT(PORTB,PB0);
        CLR_BIT(PORTA,PA4);
        flag = 1;
    }
    else
    {
        CLR_BIT(PORTB,PB0);
        SET_BIT(PORTA,PA4);
        flag = 0;
    }
    if(GET_BIT(PIND,PD3) == 1)
    {
        speed = UART_recieveByte();
        _delay_ms(200);
    }
```

Code : Motor control ECU(CONTD)

```
            /*Control the motor using timer0*/
            if(GET_BIT(PIND,PD7))
            {
                timer0_set_dutyCycle((u8)((float)speed*255/100));
            }
            else
            {
                timer0_set_dutyCycle(0);
            }
            LCD_goToRowColumn(1,0);
            LCD_displayString("Speed ");
            LCD_intgerToString(speed);
            LCD_displayString(" %");

            /*

             */
        }
        return 0;
}

ISR(TIMER1_COMPA_vect)
{
        if(flag == 1)
        {
            x++;
            if(x == 5)   //Should be 5
            {
                CLR_BIT(PORTB,PB0);
                x = 0;
                z++;
                if(z == 2)
                {
                    fan = 1;
                    CLR_BIT(PORTB,PB3);
                    CLR_BIT(PORTD,PD5);
                }
            }
```

Code : Motor control ECU(CONTD)

```
    else if(fan == 1)
    {
        y++;
        if(y == 5)   // Should be 5
        {
            SET_BIT(PORTD,PD5);
            y = 0;
        }
    }
    else
    {
        x = 0;
        y = 0;
    }
}
```

Code : Lighting ECU

```
#include "REG_DEF.h"
#include "BIT_MATH.h"
#include "STD_TYPES.h"
#include "lcd.h"
#include "I2C_Slave.h"
#include "avr/interrupt.h"

#define I_BIT       7

int main(void)
{
    /*****************************interrupts*****************************/
    /*INT0 input  */
    GICR = 1<<INT0;      /* Enable INT0*/
    MCUCR = 1<<ISC01;  /* Trigger INT0 on falling edge */
    SET_BIT(SREG,I_BIT);      /* Enable Global Interrupt */

    CLR_BIT(DDRD,PD2);
    SET_BIT(PIND,PD2);

    /*********************************************************************/
    LCD_init();
    TWI_init_slave(); // Function to initilaize slave
    /*SET PINs PD0 : PD3 output*/
    SET_BIT(DDRD,PD4);
    SET_BIT(DDRD,PD5);
    SET_BIT(DDRD,PD6);
    SET_BIT(DDRD,PD7);

    u16 led;
    while(1)
    {
        /*************************The driver of I2C is from Google*************************************/
        TWI_match_read_slave(); //Function to match the slave address and slave dirction bit(read)
        led = TWI_read_slave(); // Function to read data
        LCD_clearScreen();
        LCD_displayString("option ");
```

Code : Lighting ECU(CONTD)

```
while(1)
{
    /*************************The driver of I2C is from Google*************************************/
    TWI_match_read_slave(); //Function to match the slave address and slave dirction bit(read)
    led = TWI_read_slave(); // Function to read data
    LCD_clearScreen();
    LCD_displayString("option ");
    LCD_intgerToString(led);
    if (led == 11)
    {
        SET_BIT(PORTD,PD4);
        CLR_BIT(PORTD,PD5);
        CLR_BIT(PORTD,PD6);
        CLR_BIT(PORTD,PD7);
    }
    else if (led == 12)
    {
        CLR_BIT(PORTD,PD4);
        SET_BIT(PORTD,PD5);
        CLR_BIT(PORTD,PD6);
        CLR_BIT(PORTD,PD7);
    }
    else if (led == 13)
    {
        CLR_BIT(PORTD,PD4);
        CLR_BIT(PORTD,PD5);
        SET_BIT(PORTD,PD6);
        CLR_BIT(PORTD,PD7);
    }
    else if (led == 14)
    {
        CLR_BIT(PORTD,PD4);
        CLR_BIT(PORTD,PD5);
        CLR_BIT(PORTD,PD6);
        SET_BIT(PORTD,PD7);
    }
```

Code : Lighting ECU(CONTD)

```
            }
            else if (led == 15)
            {
                SET_BIT(PORTD,PD4);
                SET_BIT(PORTD,PD5);
                CLR_BIT(PORTD,PD6);
                CLR_BIT(PORTD,PD7);
            }
            else if (led == 16)
            {
                CLR_BIT(PORTD,PD4);
                CLR_BIT(PORTD,PD5);
                SET_BIT(PORTD,PD6);
                SET_BIT(PORTD,PD7);
            }
        }
        return 0;
}
ISR(INT0_vect)
{
        /*For reseting after Failure*/
        CLR_BIT(PORTD,PD4);
        CLR_BIT(PORTD,PD5);
        CLR_BIT(PORTD,PD6);
        CLR_BIT(PORTD,PD7);

}
```