Section 1 : General Lab Info /0.5 points

Microprocessors II and Embedded System Design

EECE.4800

Lab 2: Interfacing with a Sensor Device on an Embedded Computer System

Professor Yan Luo

Group number 12

Dushyanth Kadari

November 1, 2017

November 1, 2017

Section 2 : Contributions /1 points

1. Group Member 1 – Aravind Dhulipalla
    - Worked on ADC code, controlling the LED based on the LDR.
    - Worked on PWM code, setting timer for pulse width control, modules setting and operation of motor.

2. Group Member 2 – Zubair Nadaph
    - Worked on Galileo Board, GPIO driver settings such that it can help in communication between board and PIC micro controller.
    - Worked on BUS protocols and strobe code.

3. Group Member 3 – Dushyanth Kadari
    - Worked on Galileo Board, PIC micro controller and GPIO configuration.
    - Worked on chip hardware circuit.

Section 3 : Purpose /0.5 points

The main purpose of the lab is to interface the Galileo Board with the PIC micro-controller in such a way that the LED is controlled by the LDR, that is connection of a sensor with the Embedded computer system. In this, the communication made between the Galileo(Master) and the PIC16F18857(Slave) in such a way that the data is transferred among themselves. The commands that Galileo receives from the user gives instructions to PIC and controls the LED and Motor.

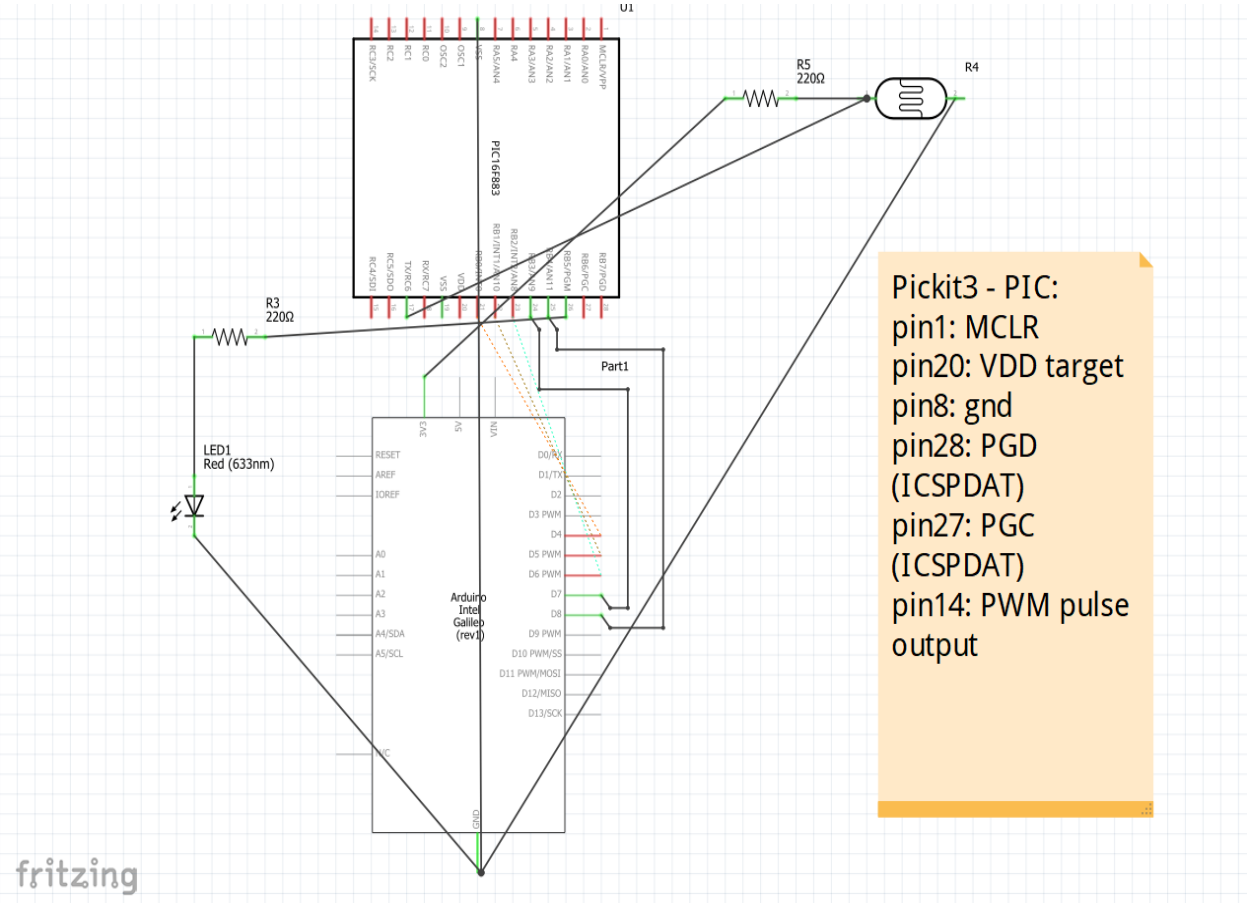
Section 4 : Introduction /0.5 points

In this lab, the Galileo is the Master and the PIC16F18857 as Slave and they are communicated to each other through the communication protocol developed to transfer the data commands among themselves. The Galileo reads the light intensity from the LDR sensor and gives the command to the PIC controller. Based on the light intensity that the Galileo Board receives and the if it is less then certain value the LED glows. It performs four commands MSG_PING, MSG_RESET, MSG_GET and MSG_TURNxxx.

Section 5 : Materials, Devices and Instruments /0.5 points

- Pic microcontroller, PIC16F18857, 3.3 – 5V (operating voltage)
- Servo Motor, SG90, 5 V (operating voltage)
- PICKit 3, 3.3 – 5V (operating voltage)
- MPLAB X IDE, V4.01
- Light Detecting Resistor (LDR)
- FTDI cable
- Breadboard and jumper wires
- 10KΩ and Ω resistors
- Intel Galileo Gen2
- Micro SD card and adaptor
- LED
- XC8 compiler
- Notepad C++ editor
- Analog Discovery2- Xilinx

Section 6 : Schematics /0.5 points



picture_1: Hardware schematic

Section 7 : Lab Methods and Procedure /2 points

Hardware design:

The LDR is connected to the PIC micro controller between pins 6 and 4 using a voltage divider circuit. A 220Ω resistor is used in this circuit, it is chosen because the minimum resistance that LDR gives under maximum light conditions in not less than 220Ω, so that the threshold voltage value can be maintained through this circuit. The output of the circuit is taken form the pin 7 as it is programmed as the

digital output. The servo motor is connected to pin 11 of the micro controller as it is programmed as output to provide the signal to the motor.

Pin 9 and 5 were grounded, pin 1 is connected to the positive voltage through a 10KΩ resistor and the operating voltage is 3.3V. Pin 2 of pickit3 and the pin19 of micro controller is connected to Vcc. Pin 4 and pin 5 of pickit3 are connected to the pin 28 and pin 27 respectively to program the chip.

The D4, D5, D6, D7 and D8 of Galileo board are connected to pins 21, 22, 23, 24 and 25 respectively. D5 and D6 are PWM connections.

Software design:

The GPIO configurations are made as given in the following table:

Galileo GPIO	PIC GPIO	Galileo GPIO	Linux OS
D4	RC0	4	6
D5	RC1	5	0
D6	RC2	6	1
D7	RC3	7	38
D8	RC4	8	40

This GPIO is used for the communication between the PIC and the Galileo board to take commands among themselves.

The GPIO configuration is done in the first place and the code for the PIC micro controller is written the MPLAB IDE as done in the LAB1. The Galileo board is programmed using the Linux commands to access the input and the output pins.

The Linux commands used in this LAB are:

1) echo out > /sys/class/gpio/gpio40/direction

2) echo in > /sys/class/gpio/gpio40/direction

3) This pin will now appear as a new directory i.e inside /sys/class/gpio/

4) It exports the value 1 into the file /sys/class/gpio/export to make it an accessible IO pin.

5) echo -n “1” > /sys/class/gpio/export

These commands are to Export and unexport, the configuration of code for Galileo and the input and the output ports.

Section 8 : Trouble Shooting /1 points

Issue 1:

In the first, the response of the code and circuit build is observed by blinking of the light both in dark and bright conditions. When the ADC values are checked in both the conditions it is observed that the voltages are 2.8V and 1V in dark and bright conditions respectively.

Issue 2:

We were not able to rotate the servomotor to certain given degrees but was able to get the responses when cross checked width the ADC values. This was not solved when the demo is give and could do it later.

Issue 3:

When tried to implement MSG_GET the ADC value supposed to get is not same as the value we got in as response. And was not solved this issue yet.

Section 9 : Results /0.5 points

We could make LED glow correctly with the commands given through the Galileo Board to the PIC micro controller. We were able to get the ADC responses for the SERVO motor control as we supposed to get.

.

Galileo code

```
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define MSG_RESET 0x0
#define MSG_PING 0x1
#define MSG_GET 0x2
#define MSG_TURN30 0x3
#define MSG_TURN90 0x04
#define MSG_TURN120 0x5

void Export()
{
        //export the pin 8 GPIO 40
                system("echo 40 > /sys/class/gpio/export");
                //export the pin 7 GPIO 38
      system("echo 38 > /sys/class/gpio/export");
                //export pin 6 GPIO 1 and SHIFTER GPIO 20
                system("echo 1 > /sys/class/gpio/export");
                system("echo 20 > /sys/class/gpio/export");
                //export pin 5 GPIO 0 and SHIFTER GPIO 18
                system("echo 0 > /sys/class/gpio/export");
                system("echo 18 > /sys/class/gpio/export");
                //export pin 4 GPIO 6 and SHIFTER GPIO 36
                system("echo 6 > /sys/class/gpio/export");
                system("echo 36 > /sys/class/gpio/export");
}
void UnExport()
{
        //export the pin 8 GPIO 40
                system("echo 40 > /sys/class/gpio/unexport");
                //export the pin 7 GPIO 38
      system("echo 38 > /sys/class/gpio/unexport");
                //export pin 6 GPIO 1 and SHIFTER GPIO 20
                system("echo 1 > /sys/class/gpio/unexport");
                system("echo 20 > /sys/class/gpio/unexport");
                //export pin 5 GPIO 0 and SHIFTER GPIO 18
                system("echo 0 > /sys/class/gpio/unexport");
                system("echo 18 > /sys/class/gpio/unexport");
                //export pin 4 GPIO 6 and SHIFTER GPIO 36
                system("echo 6 > /sys/class/gpio/unexport");
                system("echo 36 > /sys/class/gpio/unexport");
}
```

```
void SetGPIO_output()
{
    //setting pin8 as an output
        system("echo out > /sys/class/gpio/gpio40/direction");
        //Setting pin7 as an output
        system("echo out > /sys/class/gpio/gpio38/direction");
        //setting pin6 as an output
        system("echo out > /sys/class/gpio/gpio1/direction");
        system("echo out > /sys/class/gpio/gpio20/direction");
        //setting pin5 as an output
        system("echo out > /sys/class/gpio/gpio0/direction");
        system("echo out > /sys/class/gpio/gpio18/direction");
        //setting pin4 as output
        system("echo out > /sys/class/gpio/gpio6/direction");
        system("echo out > /sys/class/gpio/gpio36/direction");
}

void SetGPIO_Input()
{
        //Setting pin7 as an input
        system("echo in > /sys/class/gpio/gpio38/direction");
        //setting pin6 as an input
        system("echo in > /sys/class/gpio/gpio1/direction");
        system("echo in > /sys/class/gpio/gpio20/direction");
        //setting pin5 as an input
        system("echo in > /sys/class/gpio/gpio0/direction");
        system("echo in > /sys/class/gpio/gpio18/direction");
        //setting pin4 as input
        system("echo in > /sys/class/gpio/gpio6/direction");
        system("echo in > /sys/class/gpio/gpio36/direction");
}

int main()
{
        int msg;
        printf("select the number of the command: \n1.MSG-RESET \n2.MSG-MSG-PING
\n3.MSG-GET \n4.MSG-TURN30 \n5.MSG-TURN90 \n6.MSGTURN120\n");
   scanf("%d",&msg);
  switch(msg)
  {
            case 1:
            Export();
            SetGPIO_output();
             system("echo 0 > /sys/class/gpio/gpio40/value");
                    system("echo 0 > /sys/class/gpio/gpio6/value");
                    system("echo 0 > /sys/class/gpio/gpio0/value");
                    system("echo 0 > /sys/class/gpio/gpio1/value");
                    system("echo 0 > /sys/class/gpio/gpio38/value");
                    system("echo 1 > /sys/class/gpio/gpio40/value");
```

```
usleep(10000);
            system("echo 0 > /sys/class/gpio/gpio40/value");
      UnExport();
            Export();
      SetGPIO_Input();
            system("echo 0 > /sys/class/gpio/gpio40/value");
            system("echo 1 > /sys/class/gpio/gpio40/value");
            system("cat /sys/class/gpio/gpio6/value");
            system("cat /sys/class/gpio/gpio0/value");
            system("cat /sys/class/gpio/gpio1/value");
            system("cat /sys/class/gpio/gpio38/value");
            usleep(10000);
            system("echo 0 > /sys/class/gpio/gpio40/value");
      UnExport();
            break;
            case 2:
            Export();
      SetGPIO_output();
            system("echo 0 > /sys/class/gpio/gpio40/value");
            system("echo 1 > /sys/class/gpio/gpio6/value");
            system("echo 0 > /sys/class/gpio/gpio0/value");
            system("echo 0 > /sys/class/gpio/gpio1/value");
            system("echo 0 > /sys/class/gpio/gpio38/value");
      system("echo 1 > /sys/class/gpio/gpio40/value");
            usleep(10000);
            system("echo 0 > /sys/class/gpio/gpio40/value");
            UnExport();
            Export();
      SetGPIO_Input();
            system("echo 0 > /sys/class/gpio/gpio40/value");
            //usleep(10000);
            system("echo 1 > /sys/class/gpio/gpio40/value");
            system("cat /sys/class/gpio/gpio6/value");
            system("cat /sys/class/gpio/gpio0/value");
            system("cat /sys/class/gpio/gpio1/value");
            system("cat /sys/class/gpio/gpio38/value");
            usleep(10000);
            system("echo 0 > /sys/class/gpio/gpio40/value");
      UnExport();
            break;
            case 3:
            Export();
      SetGPIO_output();
            system("echo 0 > /sys/class/gpio/gpio40/value");
            system("echo 0 > /sys/class/gpio/gpio6/value");
            system("echo 1 > /sys/class/gpio/gpio0/value");
            system("echo 0 > /sys/class/gpio/gpio1/value");
            system("echo 0 > /sys/class/gpio/gpio38/value");
            system("echo 1 > /sys/class/gpio/gpio40/value");
```

```
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        UnExport();
        Export();
SetGPIO_Input();
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 1 > /sys/class/gpio/gpio40/value");
        system("cat /sys/class/gpio/gpio6/value");
        system("cat /sys/class/gpio/gpio0/value");
        system("cat /sys/class/gpio/gpio1/value");
        system("cat /sys/class/gpio/gpio38/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 1 > /sys/class/gpio/gpio40/value");
        system("cat /sys/class/gpio/gpio6/value");
        system("cat /sys/class/gpio/gpio0/value");
        system("cat /sys/class/gpio/gpio1/value");
        system("cat /sys/class/gpio/gpio38/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 1 > /sys/class/gpio/gpio40/value");
        system("cat /sys/class/gpio/gpio6/value");
        system("cat /sys/class/gpio/gpio0/value");
        system("cat /sys/class/gpio/gpio1/value");
        system("cat /sys/class/gpio/gpio38/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 1 > /sys/class/gpio/gpio40/value");
        system("cat /sys/class/gpio/gpio6/value");
        system("cat /sys/class/gpio/gpio0/value");
        system("cat /sys/class/gpio/gpio1/value");
        system("cat /sys/class/gpio/gpio38/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        UnExport();
        break;
        case 4:
                        Export();
SetGPIO_output();
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 1 > /sys/class/gpio/gpio6/value");
        system("echo 1 > /sys/class/gpio/gpio0/value");
        system("echo 0 > /sys/class/gpio/gpio1/value");
        system("echo 0 > /sys/class/gpio/gpio38/value");
        system("echo 1 > /sys/class/gpio/gpio40/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        UnExport();
        Export();
```

```
  SetGPIO_Input();
system("echo 0 > /sys/class/gpio/gpio40/value");
        usleep(10000);
        system("echo 1 > /sys/class/gpio/gpio40/value");
        system("cat /sys/class/gpio/gpio6/value");
        system("cat /sys/class/gpio/gpio0/value");
        system("cat /sys/class/gpio/gpio1/value");
        system("cat /sys/class/gpio/gpio38/value");
        system("echo 0 > /sys/class/gpio/gpio40/value");
  UnExport();
  UnExport();
        break;
        case 5:
                        Export();
  SetGPIO_output();
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 0 > /sys/class/gpio/gpio6/value");
        system("echo 0 > /sys/class/gpio/gpio0/value");
        system("echo 1 > /sys/class/gpio/gpio1/value");
        system("echo 0 > /sys/class/gpio/gpio38/value");
        system("echo 1 > /sys/class/gpio/gpio40/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        UnExport();
        Export();
  SetGPIO_Input();
 system("echo 0 > /sys/class/gpio/gpio40/value");
        usleep(10000);
        system("echo 1 > /sys/class/gpio/gpio40/value");
        system("cat /sys/class/gpio/gpio6/value");
        system("cat /sys/class/gpio/gpio0/value");
        system("cat /sys/class/gpio/gpio1/value");
        system("cat /sys/class/gpio/gpio38/value");
        system("echo 0 > /sys/class/gpio/gpio40/value");
  UnExport();
  UnExport();
        break;
        case 6:
                        Export();
  SetGPIO_output();
        system("echo 0 > /sys/class/gpio/gpio40/value");
        system("echo 1 > /sys/class/gpio/gpio6/value");
        system("echo 0 > /sys/class/gpio/gpio0/value");
        system("echo 1 > /sys/class/gpio/gpio1/value");
        system("echo 0 > /sys/class/gpio/gpio38/value");
  system("echo 1 > /sys/class/gpio/gpio40/value");
        usleep(10000);
        system("echo 0 > /sys/class/gpio/gpio40/value");
        UnExport();
```

```
            Export();
          SetGPIO_Input();
        system("echo 0 > /sys/class/gpio/gpio40/value");
            usleep(10000);
            system("echo 1 > /sys/class/gpio/gpio40/value");
            system("cat /sys/class/gpio/gpio6/value");
            system("cat /sys/class/gpio/gpio0/value");
            system("cat /sys/class/gpio/gpio1/value");
            system("cat /sys/class/gpio/gpio38/value");
            system("echo 0 > /sys/class/gpio/gpio40/value");
          UnExport();
            break;
  }
}
```

Main Code

```
#include <pic16f18857.h>
#include "mcc_generated_files/mcc.h" //default library
#define value 0x0
#define MSG_ACK 0xE
#define MSG_NOTHING 0xF
#define ADC_Temp 0b0000000000

/* Circuit Connections
  Signal STROBE   RC6
  Signal D0       RC2
  Signal D1       RC3
  Signal D2       RC4
  Signal D3       RC5
 */
void Timer2_Init(void)
{
// CCPTMRS0 = 0x01; //SELECTED TIMER 2 FOR PWM
T2CON = 0x80; //CONFIGURED TIMER 2
T2CLKCONbits.CS = 0x01; //clk in relation with osc frequency
T2HLT = 0x00; //TIMER MODE
T2RST = 0x00; //Reset Source
PR2 = 0xFF; //lOAD THE PR2 VALUE
TMR2 = 0x00; //PRESCALE VALUE IS 0
PIR4bits.TMR2IF = 0; // CLEAR THE INTERRUPT FLAG
//T2CONbits.ON = 1; // START THE TIMER
}
void PWM_Init(void)
{
CCP1CONbits.EN = 1; // ENABLING THE
CCP1CONbits.FMT = 0; //RIGHT ALLIGNED FORMAT
```

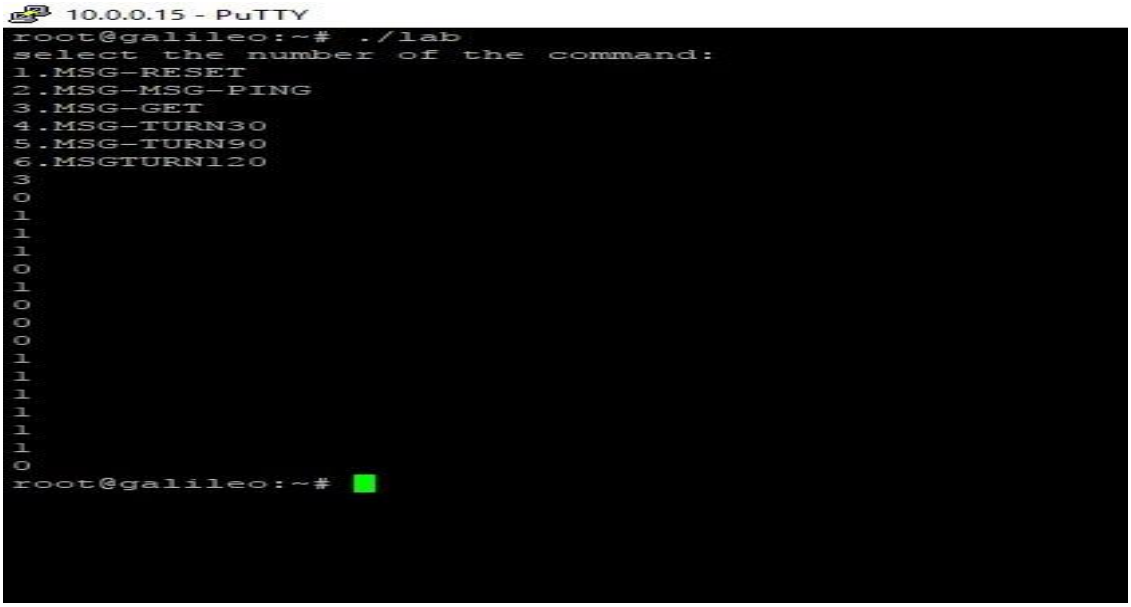
```
CCP1CONbits.MODE = 0xF; // SETTING THE MODE TO PWM
CCPR1H = 0x00; // RH TO 0
CCPR1L = 0X00; //RL TO 0
CCPTMRS0 = 0X01; // SELECTS TIMER2
}
void PWM_signal_out(unsigned int duty)
{
   T2CONbits.ON = 1; // START THE TIMER
PMD3bits.PWM6MD = 0; //PWM 6 is enabled
CCPR1H = duty >>2; // 2 MSB'S IN CCPR1H
CCPR1L = (duty & 0x0003)<<6; //8 LSB'S IN CCPR1L
}
 int ADC_conversion_results()
{
   TRISAbits.TRISA0 = 1; // SETTING PORTA PIN0 TRISTATE REGISTER TO INPUT
   ANSELAbits.ANSA0 = 1; // SETTING PORTA PIN0 AS A ANALOG INPUT
   ADCON0bits.ADON = 1;  // ACTIVATING THE ADC MODULE
   ADCON0bits.GO = 1;    // START CONVERTING
   while(ADCON0bits.ADGO)// WAIT UNTIL THE CONVERSION
   {
   }
   int b = (ADRESH<<8)+(ADRESL); // MAKE THE ADC RESULT IN 10BITS
     ADCON0bits.GO = 0; // STOP CONVERTING
   return(b); // RETURN THE RESULT VALUE
}

void ADC_Init(void)
{
   ADCON1 = 0x00; // setting control register 1 to 0
   ADCON2 = 0x00; // setting control register 2 to 0
   ADCON3 = 0x00; // setting control register 3 to 0
   ADSTAT = 0x00; // setting threshold register and not overflowed  to 0
   ADCAP  = 0x00; // disabling ADC capacitors
   ADACT  = 0x00; // disabling Auto conversion trigger control register
   ADPRE  = 0x00; // setting precharge time control to 0
   ADCLK  = 0x00; // setting ADC clk
   ADREF  = 0x00; // setting ADC positive and negative reference voltages
   ANSELAbits.ANSA0 = 1; // setting ADC analog channel input to 1
   ADCON0 = 0x84; // setting ADCON0 to the required mode.

}

void set_receive()
{
  //1.set RC6 as digital input
  //2.set RC2, RC3, RC4 and RC5 as digital inputs
  //TRISC = 0xFF;
   ANSELB = 0x00;
  TRISBbits.TRISB0=1;
```

```
   TRISBbits.TRISB1=1;
   TRISBbits.TRISB2=1;
   TRISBbits.TRISB3=1;
   TRISBbits.TRISB4=1;
}

void set_send()
{
    ANSELB = 0x00;
    TRISBbits.TRISB0= 0;
    TRISBbits.TRISB1= 0;
    TRISBbits.TRISB2= 0;
    TRISBbits.TRISB3= 0;
}

unsigned char receive_msg()
{
    /* 1.wait strobe high
    2.wait strobe low
    3.read the data
    4.wait strobe high
    5.return the data*/
    set_receive();
    while(PORTBbits.RB4 == 0);
    unsigned char message = 0x00;
     message = ((PORTBbits.RB0)| (PORTBbits.RB1<<1) | (PORTBbits.RB2<<2) |
(PORTBbits.RB3<<3));
    while(PORTBbits.RB4 == 1);
    return message;
}


void Strobe(char message)
{
     ANSELB = 0x00;
    TRISBbits.TRISB0=0;
    TRISBbits.TRISB1=0;
    TRISBbits.TRISB2=0;
    TRISBbits.TRISB3=0;
    TRISBbits.TRISB4 = 1;
    /*PORTBbits.RB0 = 1;
    PORTBbits.RB1 = 1;
    PORTBbits.RB2 = 1;
    PORTBbits.RB3 = 1;
     */
    while(PORTBbits.RB4==1);
    LATBbits.LATB0 = message & 0x01;
    LATBbits.LATB1 = (message>>1)&0x01;
    LATBbits.LATB2 = (message>>2)&0x01;
```

```
    LATBbits.LATB3 = (message>>3)&0x01;
    while(PORTBbits.RB4==0);
}


void SendADC(int ADCValue)
{

    set_send();
    char a = (ADCValue & 0x0F);
    char b = (ADCValue & 0xF0)>>4;
    char c = (ADCValue & 0x300)>>8;
    Strobe(a);
    Strobe(b);
    Strobe(c);
   }
// Main program
void main (void)
{

    int ADC;
    SYSTEM_Initialize();
    ADC_Init();
    Timer2_Init();
    PWM_Init();
    TRISCbits.TRISC2=0;
    unsigned char msg;
//    ANSELC =0;
    while(1)
    {
    msg=receive_msg();
    switch(msg)
    {
        //Reset
        case 0x00:
          Strobe(MSG_ACK);
          __delay_ms(1000);
          SYSTEM_Initialize();
          break;
          //PING
        case 0x01:
            Strobe(MSG_ACK);
            break;
         //Get
        case 0x02:
            Strobe(MSG_ACK);
            ADC = ADC_conversion_results();
            SendADC(ADC);
            break;
```

```
        //TURN 30
        case 0x03:
           PWM_signal_out(100);
           for(int i=0;i<=35;i++)
           {
              PORTCbits.RC2 = 1;
              __delay_ms(3.500);
              PORTCbits.RC2 = 0;
              __delay_ms(16.500);
           }
           break;
        //TURN 90
        case 0x04:
           for(int i=0;i<=35;i++)
           {
              PORTCbits.RC2 = 1;
              __delay_ms(3.500);
              PORTCbits.RC2 = 0;
              __delay_ms(16.500);
           }
           break;
        //TURN 120
        case 0x05:
           for(int i=0;i<=35;i++)
           {
              PORTCbits.RC2 = 1;
              __delay_ms(3.500);
              PORTCbits.RC2 = 0;
              __delay_ms(16.500);
           }
           break;
     }
     }
}
```

Result:


10.0.0.15 - PuTTY
root@galileo:~# ./lab
select the number of the command:
1.MSG-RESET
2.MSG-MSG-PING
3.MSG-GET
4.MSG-TURN30
5.MSG-TURN90
6.MSGTURN120
3
0
1
1
1
0
1
0
0
0
1
1
1
1
1
0
root@galileo:~#