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

***16.480/552 Microprocessor Design II and Embedded Systems***

***Instructor:*** *Prof. Yan Luo*

*Group-12*

*Due:11/01/17*

*Submitted: 11/01/17*

*By,*

* ***Aravind Dhulipalla***
* *Zubair Sikandar Nadaph*
* *Dushyanth Kadari*

**Contributions:**

**Group Member I: Aravind Dhulipalla –** Deigned the code for PIC side also helped for coding the Galileo part.

**Group Member 2: Zubair Sikandar Nadaph** – Assisted with the connections of GPIO ports and also with the strobe signal also assisted in developing the pic side of the code.

**Group Member 3: Dhushyanth** -

**Purpose:**

The main purpose of this project is to design bus protocol between intel Galileo (master) and PIC (slave). And transmit the ADC value read by PIC to the Galileo board over bus protocol using strobe mechanism. And display those value on the computer screen using putty software.

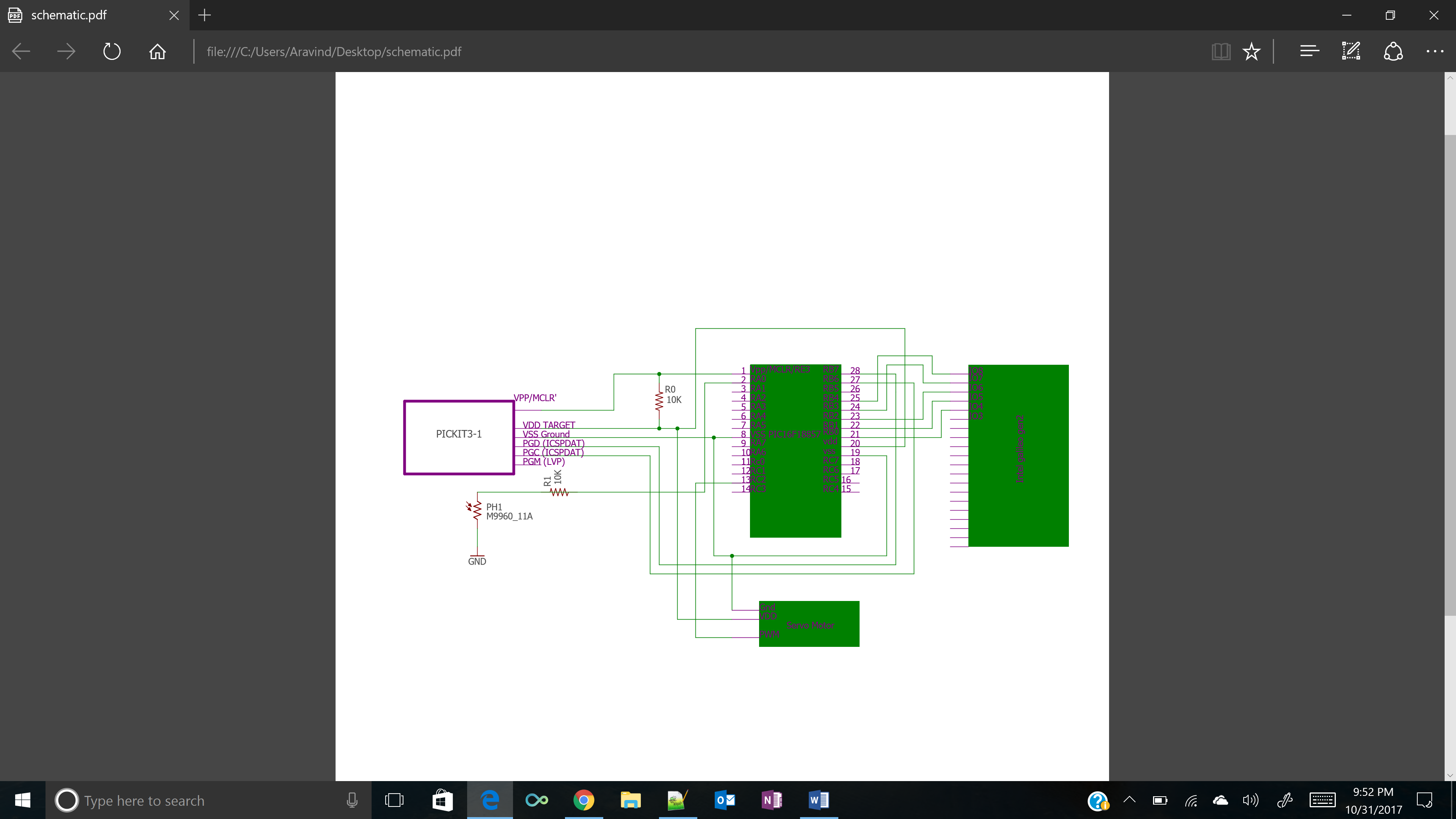
**Introduction:**

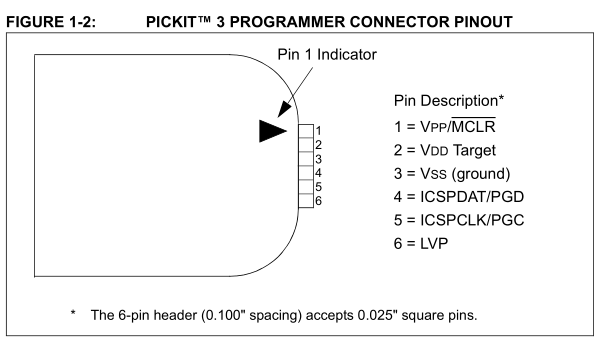
The main objective of this lab is to read the data from a photo resistor through ADC module of the microcontroller PIC16F18857. Send those 10-bit data, over the 4-bit bus using a strobe mechanism, by breaking the value to 4 bits. The Galileo sends one of these commands MSG\_PING, MSG\_RESET, MSG\_GET and MSG\_TURNxxx to the PIC microcontroller and PIC responds to these commands respectively. When the Galileo sends the MSG\_Get command the PIC reads the ADC value and sends it to the Galileo and it prints the value.

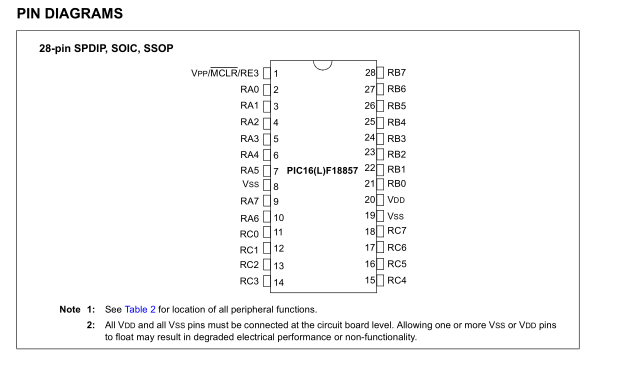
**Materials, devices and Instruments:**

1. Bread board
2. Wires to connect
3. PIC16F18857 microcontroller
4. Pickit3
5. LED
6. Photo resistor
7. one 10k resistor
8. Servo Motor
9. Serial to USB connector
10. Multi-meter
11. Voltage supply (3.3V)
12. Intel Galileo Gen 2 Board
13. Yocto Linux
14. MPLAB IDE
15. Putty Software

**Schematic:**







**Hardware Design:** Initially Pickit3 is connected to the microcontroller. The Galileo GPIO ports are used to used to connect to the PIC microcontroller. The Galileo’s D4,D5,D6,D7 are connected to the PIC’s Port B RB0.RB1,RB2,RB3 repectively and Galileo’s D8 is used as a stobe and it is connected to the PIC’s PORTB pin RB4. The galileo sets the strobe as low saying that there is some data that need to be sent or to be received. Then it sets the stobe to high and puts the command on the data bus and makes the strobe go low saying that data has been sent. PIC will receive the data from the data bus when the strobe signal is high and when it is low it knows that the data sending is completed. Then PIC sees that the strobe is low that means it can send the data and when the stobe goes high it keeps the data on the data bus the galileo reads the data and sets the strobe to low.

The LDR is connected to pin RA0 and ADC is configured to use the PORTA Pin RA0 so that it can read the value from LDR and convert it to digital value using the internal ADC module in the PIC.

Linux commands used for GPIO’s

* echo -n “x” > /sys/class/gpio/export – exports the respective GPIO pinx
* echo in > /sys/class/gpio/gpio$x/direction – sets the direction as input for gpio pin x
* echo out > /sys/class/gpio/gpio$x/direction – sets the direction as output for gpio pinx
* echo -n “x” > /sys/class/gpio/unexport – unexports the respective GPIO pin
* echo -n “0 or 1”> /sys/class/gpio/gpio$x/value – writes 0 or 1 if the direction is out
* cat /sys/class/gpio/gpio38/value – gives the value if the pin is set as an input

To configure a gpio pin on galileo first we need to export the pin package and we should set the direction as input or output and we can read or write to the pin using value. Aslo we need to configure the shifter gpio pins to make the GPIO’s to work.

**Software Design:**

System\_Initialize



Read data from galileo



read data from



NO

NO

NO

NO

NO

value==5

value==4

value==3

value==2

value==0

value==1

value==0

alue==0

Turn 120 & send Ack

Turn 90& send Ack

Turn 30 & send Ack

Send Ack& Send ADC

Send Ack

Send Ack& System\_initialize

**Galileo Flowchart:**

Init



Get Command



read data from



NO

NO

NO

NO

NO

cmd==5

cmd==4

cmd==3

cmd==2

Cmd = 0

cmd==1

value==0

alue==0

no

yes

yes

yes

yes

Yes

Yes

Send Ack& receive and print ADC Value

Turn 120 & wait for ack

Turn 90& swait for ack

Send 3 and wait for ack

Send 1 and wait for ack

Send 0 and wait for ack

**PIC code:**

/\*

\* File: PIC and Galileo communication

\*

\*

\* simple PIC program example

\* for UMass Lowell 16.480/552

\*

\* Author: Aravind, Zubair, Dushyanth

\*

\*/

#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;

}

}

}

**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;

}

}

**Results:**

**A screenshot of a computer

Description generated with high confidence**

As you can see in the putty when we gave a command 3 which is MSG\_Get the Pic send an Ack and an 12 bit ADC value the first 4 bits which are 0 1 1 1 is an ACK message and the next 10 bits is our ADC value.

1. Trouble Shooting

The first is to assure the correctness of the circuit by blinking the LED for few minutes without the sensor data. Then relate the sensor data to ON the led. To check the sensor, you need to measure the voltage across it during blocked and unblocked situations. The measured values are 2.8v during unblocked and 1v during the blocked. To estimate ADC output using the formula. Signal = (sample/1024) \* Reference voltage. For 2.8v, sample = 868. For 1v, sample = 310 with reference voltage = 3.3 v and (2^10- 10bits of ADC value) 1024 base value.

The second is to troubleshoot the PORTB- data pins configurations by blinking the LED at every pin used.

Third is to configure GPIO ports 4,5,6,7,8 of Intel Galileo by blinking LED at each port used.