**CSE 316: Microcontrollers**

Project

Submitted by :

Seif Eldin Mohamed Hussein I.D:16P8109

Mostafa Mahmoud Elrosasy I.D:16P8113

Peter Nabil Zaghloul I.D:16P8100

Mahmoud Fathy Sayed I.D:16P6003

Medhat Ashraf Alhaddad I.D:11P1109

Submitted to :

**Dr. Haytham Azmi**

**Eng. Mostafa Arafa**

# **COURSE PROJECT EVALUATION SHEET**

# 

# 

# **Presentation Date: 28/12/2019**

# 

# **Project Title: Hand-Gesture controlled car**

# 

# **Short Description:**

# **The project is a tiva c controlled car that the user controls using a glove.**

# **The glove contains an imu that detects the motion of the hand and sends the data to the car through bluetooth modules using master/slave architecture.**

# 

# **Team Members:**

# 

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Role in the project** | **Evaluation** |
| Seif Eldin Mohamed | **I2c and Imu** |  |
| Medhat Ashraf Alhaddad | **Main function in the sender** |  |
| Mostafa Mahmoud Elrosasy | **uart** |  |
| Mahmoud Fathy | **pwm** |  |
| Peter Nabil | **Testing uart and pwm** **+ integration** |  |

# Block Diagram

# Receiver Code

**Main.h**

#include "tm4c123gh6pm.h"

#include "stdio.h"

#include "stdlib.h"

#include <stdint.h>

#include "imu.h"

#include "systick.h"

#include "uart.h"

#define HBridge GPIO\_PORTE\_DATA\_R

**Main.c**

#include "tm4c123gh6pm.h"

#include "main.h"

//IN1, PE0 Fowards, RightSide

//IN2, PE1 Backwards, RightSide

//IN3, PE2 Forwards, LeftSide

//IN4, PE3 Backwards, LeftSide

//PWM PinS, PB6.

//IN4,IN3,IN2,IN1

//Right= 0110

//Left= 1001

//Fowards= 0101

//Backwards= 1010

void PWM\_init(int cmp){

SYSCTL\_RCGC0\_R |=(0x100000);

SYSCTL\_RCC\_R |= 0x140000;

GPIO\_PORTB\_AFSEL\_R |=(0x40);

GPIO\_PORTB\_PCTL\_R |=(0x04000000);

PWM0\_0\_CTL\_R &= ~(0x01);

PWM0\_0\_CTL\_R &= ~(0x02);

PWM0\_0\_GENA\_R |= (0xC2);

PWM0\_0\_LOAD\_R = 1000-1;

PWM0\_0\_CMPA\_R = cmp-1;

PWM0\_0\_CTL\_R |=(0x01);

PWM0\_ENABLE\_R |= (PWM\_ENABLE\_PWM0EN);

}

int main()

{

UART\_Init();

SYSCTL\_RCGCGPIO\_R |= SYSCTL\_RCGCGPIO\_R1;

SYSCTL\_RCGCGPIO\_R |= SYSCTL\_RCGCGPIO\_R4;

GPIO\_PORTE\_DEN\_R |= (0xFF);

GPIO\_PORTE\_DIR\_R |= (0xFF);

PWM\_init(1);

uint8\_t x ;

HBridge &= ~(0xFF);

while(1){

x = UART\_InChar();

// printf("%x\n",x);

if(x/16 == 0x8){ //Right

HBridge &= ~(0xFF);

HBridge |=(1<<2) | (1<<0);

if(x%16== 0x1){ //25%

//PWM\_init(750);

PWM0\_0\_CMPA\_R= 750-1;

}

else if(x%16 == 0x02){ //50%

//PWM\_init(500);

PWM0\_0\_CMPA\_R=500-1;

}

else if(x%16== 0x04){ //75%

//PWM\_init(250);

PWM0\_0\_CMPA\_R=250-1;

}

else{

//PWM\_init(1);

PWM0\_0\_CMPA\_R=0;

}

}

else if(x/16 == 0x4){ //Left

HBridge &= ~(0xFF);

HBridge |=(1<<3) | (1<<1);

if(x%16== 0x1){ //25%

//PWM\_init(750);

PWM0\_0\_CMPA\_R=750-1;

}

else if(x%16 == 0x02){

//PWM\_init(500);

PWM0\_0\_CMPA\_R=500-1;

}

else if(x%16== 0x04){

//PWM\_init(250);

PWM0\_0\_CMPA\_R=250-1;

}

else{

//PWM\_init(1);

PWM0\_0\_CMPA\_R=0;

}

}

else if(x/16 == 0x02){ //Fowards

HBridge &= ~(0xFF);

HBridge |=(1<<3) | (1<<0);

if(x%16== 0x1){ //25%

//PWM\_init(750);

PWM0\_0\_CMPA\_R=750-1;

}

else if(x%16 == 0x02){

//PWM\_init(500);

PWM0\_0\_CMPA\_R=500-1;

}

else if(x%16== 0x04){

//PWM\_init(250);

PWM0\_0\_CMPA\_R=250-1;

}

else{

//PWM\_init(1);

PWM0\_0\_CMPA\_R=0;

}

}

else if(x/16 == 0x01){ //Backwards

HBridge &= ~(0xFF);

HBridge |=(1<<2) | (1<<1);

//HBridge &= ~(0x05);

if(x%16== 0x1){ //25%

//PWM\_init(750);

PWM0\_0\_CMPA\_R=750-1;

}

else if(x%16 == 0x02){

//PWM\_init(500);

PWM0\_0\_CMPA\_R=500-1;

}

else if(x%16== 0x04){

//PWM\_init(250);

PWM0\_0\_CMPA\_R=250-1;

}

else{

//PWM\_init(1);

PWM0\_0\_CMPA\_R=0;

}

}

else{ //Free Engine Stop

HBridge &= ~(0xFF);

PWM\_init(1000);

}

delayMs2(10);

}

}

**Uart.h**

#include "tm4c123gh6pm.h"

#include <stdint.h>

#include <stdio.h>

void UART\_Init(void);

uint8\_t UART\_InChar(void);

**Uart.c**

#include "uart.h"

void UART\_Init(void){

SYSCTL\_RCGCUART\_R |= 0x0002; // activate UART1

SYSCTL\_RCGCGPIO\_R |= 0x0002; // activate port B

UART1\_CTL\_R &= ~0x0001; // disable UART

UART1\_IBRD\_R = 104;

// IBRD=int(50000000/(16\*115,200)) = int(27.1267)

UART1\_FBRD\_R = 11;

// FBRD = round(0.1267 \* 64) = 8

UART1\_LCRH\_R = 0x0070; // 8-bit length, enable FIFO

UART1\_CTL\_R = 0x0301; // enable RXE, TXE and UART

GPIO\_PORTB\_AFSEL\_R |= 0x03; // alt funct on PB1-0

GPIO\_PORTB\_PCTL\_R =

(GPIO\_PORTB\_PCTL\_R&0xFFFFFF00)+0x00000011;

GPIO\_PORTB\_DEN\_R |= 0x03; // digital I/O on PB1-0

GPIO\_PORTB\_AMSEL\_R &= ~0x03; // No analog on PB1-0

}

uint8\_t UART\_InChar(void) {

while((UART1\_FR\_R&0x0010) != 0);

// wait until RXFE is 0

return((uint8\_t)(UART1\_DR\_R&0xFF));

}

**Systic.h**

#include "tm4c123gh6pm.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void delayUs(int us);

void delayMs2(int ms);

void delayS(void);

**Systic.c**

#include "systick.h"

void delayUs(int us){

NVIC\_ST\_CTRL\_R=0;

int delay = us \* 15;

NVIC\_ST\_RELOAD\_R = delay;

NVIC\_ST\_CTRL\_R = 5; //7 for interrupts

NVIC\_ST\_CURRENT\_R = 0;

while ((NVIC\_ST\_CTRL\_R & 0x10000) == 0);

}

void delayMs2(int ms){

NVIC\_ST\_CTRL\_R=0;

int delay = ms \* 15999;

NVIC\_ST\_RELOAD\_R = delay;

NVIC\_ST\_CTRL\_R = 5; //7 for interrupts

NVIC\_ST\_CURRENT\_R = 0;

while ((NVIC\_ST\_CTRL\_R & 0x10000) == 0);

}

void delayS(void){

NVIC\_ST\_CTRL\_R=0;

int delay = 15999999;

NVIC\_ST\_RELOAD\_R = delay;

NVIC\_ST\_CTRL\_R = 5; //7 for interrupts

NVIC\_ST\_CURRENT\_R = 0;

while ((NVIC\_ST\_CTRL\_R & 0x10000) == 0);

}

# Sender Code

**Main.h**

#include "tm4c123gh6pm.h"

#include "stdio.h"

#include "stdlib.h"

#include "imu.h"

#include "stdint.h"

#include "uart.h"

void init();

void encoding();

**Main.c**

#include "main.h"

#include "I2C.h"

#define SLAVE\_ADDR 0x68

uint8\_t code1 = 0x0;

uint8\_t code2 = 0x0;

uint8\_t code = 0x00;

double accel\_angle;

float axf, ayf, azf;

float Acc\_angle\_x, Acc\_angle\_y;

extern float Acc\_angle\_error\_x, Acc\_angle\_error\_y;

extern float rad\_to\_deg;

int main()

{

char accel\_data[6];

int16\_t ax, ay, az;

init();

I2C\_Init();

IMU\_Init();

UART\_Init();

while (1)

{

I2C\_Read(SLAVE\_ADDR, 0x3B, 6, accel\_data);

ax = (int16\_t) ((accel\_data[0] << 8) | (accel\_data[1]));

ay = (int16\_t) ((accel\_data[2] << 8) | (accel\_data[3]));

az = (int16\_t) ((accel\_data[4] << 8) | (accel\_data[5]));

//printf("ax= %d \n",ax);

//printf("ay= %d \n",ay);

//printf("azf= %f \n",azf);

axf = (((float) ax) / 4096.0);

ayf = (((float) ay) / 4096.0);

azf = (((float) az) / 4096.0);

//printf("axf= %f \n",axf);

//printf("ayf= %f \n",ayf);

//printf("azf= %f \n",azf);

/\*---X---\*/

Acc\_angle\_x = (atan((ayf)/sqrt(pow((axf),2) + pow((azf),2)))\*rad\_to\_deg);

/\*---Y---\*/

Acc\_angle\_y = (atan(-1\*(axf)/sqrt(pow((ayf),2) + pow((azf),2)))\*rad\_to\_deg);

//printf("xangle= %f \n",Acc\_angle\_x);

//printf("yangle= %f \n",Acc\_angle\_y);

encoding();

//printf("code= %x \n",code);

uart\_send(code);

}

}

void encoding(){

// dir

if( (Acc\_angle\_x <= 15 && Acc\_angle\_x >= -15) && (Acc\_angle\_y <= 15 && Acc\_angle\_y >= -15) ) // not moving

{

code1 = 0x0;

code2 = 0x0;

}

else if(Acc\_angle\_y > 15) // forward

code1 = 0x20;

else if(Acc\_angle\_y < -15) // backward

code1 = 0x10;

else if(Acc\_angle\_x > 15) // right

code1 = 0x80;

else if(Acc\_angle\_x < -15) // left

code1 = 0x40;

// speed -ve

if( (Acc\_angle\_x < -15 && Acc\_angle\_x > -30) || (Acc\_angle\_y < -15 && Acc\_angle\_y > -30) ) // 25% pwm

code2 = 0x1;

else if( (Acc\_angle\_x < -30 && Acc\_angle\_x > -60) || (Acc\_angle\_y < -30 && Acc\_angle\_y > -60) ) // 50% pwm

code2 = 0x2;

else if( (Acc\_angle\_x < -60 && Acc\_angle\_x > -90) || (Acc\_angle\_y < -60 && Acc\_angle\_y > -90) ) // 75% pwm

code2 = 0x4;

else if( (Acc\_angle\_x < -90) || (Acc\_angle\_y < -90) ) // 100% pwm

code2 = 0x8;

if( (Acc\_angle\_x > 15 && Acc\_angle\_x < 30) || (Acc\_angle\_y > 15 && Acc\_angle\_y < 30) ) // 25% pwm

code2 = 0x1;

else if( (Acc\_angle\_x > 30 && Acc\_angle\_x < 60) || (Acc\_angle\_y > 30 && Acc\_angle\_y < 60) ) // 50% pwm

code2 = 0x2;

else if( (Acc\_angle\_x > 60 && Acc\_angle\_x < 90) || (Acc\_angle\_y > 60 && Acc\_angle\_y < 90) ) // 75% pwm

code2 = 0x4;

else if( (Acc\_angle\_x > 90) || (Acc\_angle\_y > 90) ) // 100% pwm

code2 = 0x8;

code = code1 + code2;

}

void init(){

SYSCTL\_RCGCGPIO\_R |= 0x08; /\* enable clock to GPIOD \*/

/\* PORTD 1, 0 for I2C3 \*/

GPIO\_PORTD\_AFSEL\_R |= 0x03; /\* PORTD 1, 0 for I2C3 \*/ //D0=scl D1=sda

GPIO\_PORTD\_PCTL\_R &= ~0x000000FF; /\* PORTD 1, 0 for I2C3 \*/

GPIO\_PORTD\_PCTL\_R |= 0x00000033;

GPIO\_PORTD\_DEN\_R |= 0x03; /\* PORTD 1, 0 as digital pins \*/

GPIO\_PORTD\_ODR\_R |= 0x02; /\* PORTD 1 as open drain \*/

}

**Uart.h**

#include "tm4c123gh6pm.h"

#include <stdio.h>

#include <stdint.h>

void UART\_Init(void);

int UART\_InChar(void);

void uart\_send(uint8\_t input);

void UART\_outChar(uint8\_t input);

**Uart.c**

**#include "uart.h"**

void UART\_Init(void){

SYSCTL\_RCGCGPIO\_R |= (1<<5);

GPIO\_PORTF\_DEN\_R |= (1<<1);

GPIO\_PORTF\_DEN\_R |= (1<<2);

GPIO\_PORTF\_DEN\_R |= (1<<3);

GPIO\_PORTF\_DIR\_R |= (1<<1);

GPIO\_PORTF\_DIR\_R |= (1<<2);

GPIO\_PORTF\_DIR\_R |= (1<<3);

SYSCTL\_RCGCUART\_R |= 0x0002; // activate UART1

SYSCTL\_RCGCGPIO\_R |= 0x0002; // activate port B

UART1\_CTL\_R &= ~0x0001; // disable UART

UART1\_IBRD\_R = 104; //9600: 104, 11 38400: :26, 3

// IBRD=int(50000000/(16\*115,200)) = int(27.1267)

UART1\_FBRD\_R = 11;

// FBRD = round(0.1267 \* 64) = 8

UART1\_LCRH\_R = 0x0070; // 8-bit length, enable FIFO

GPIO\_PORTB\_AFSEL\_R |= 0x03; // alt funct on PB1-0

GPIO\_PORTB\_PCTL\_R =

(GPIO\_PORTB\_PCTL\_R&0xFFFFFF00)+0x00000011;

GPIO\_PORTB\_DEN\_R |= 0x03; // digital I/O on PB1-0

GPIO\_PORTB\_AMSEL\_R &= ~0x03; // No analog on PB1-0

UART1\_CTL\_R = 0x0301; // enable RXE, TXE and UART

}

int UART\_InChar(void) {

while((UART1\_FR\_R&0x0010) != 0);

// wait until RXFE is 0

return((uint8\_t)(UART1\_DR\_R));

}

void UART\_outChar(uint8\_t data) {

while((UART1\_FR\_R&0x0020) != 0);

// wait until TXFF is 0

UART1\_DR\_R = data;

}

void uart\_send(uint8\_t input)

{

// uint8\_t code = 0;

// char \* atcom[]= “AT+ROLE=1\r\n”;

UART\_outChar(input);

printf("%x",input);

if(input == 1){

GPIO\_PORTF\_DATA\_R |=(1<<1);

}

if(input == '2'){

GPIO\_PORTF\_DATA\_R |=(1<<2);

}

if(input == '3'){

GPIO\_PORTF\_DATA\_R |=(1<<3);

}

volatile int count = 0;

// while(count < 1000000){count++;}

}

**I2C.h**

#ifndef I2C\_H\_

#define I2C\_H\_

#include "tm4c123gh6pm.h"

#include <stdbool.h>

#include <stdint.h>

#include "stdio.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// I2C Driver Functions

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void I2C\_Init();

char I2C\_Write(int slaveAddr, char memAddr, char data);

char I2C\_Read(int slaveAddr, char memAddr, int byteCount, char\* data);

#endif /\* I2C\_H\_ \*/

**I2C.c**

#include "I2C.h"

void I2C\_Init()

{

SYSCTL\_RCGCI2C\_R |= 0x08;

I2C3\_MCR\_R = 0x10;

I2C3\_MTPR\_R = 7;

}

static int I2C\_wait\_till\_done(void)

{

while (I2C3\_MCS\_R & 1);

return I2C3\_MCS\_R & 0xE;

}

char I2C\_Write(int slaveAddr, char memAddr, char data)

{

char error;

I2C3\_MSA\_R = slaveAddr << 1;

I2C3\_MDR\_R = memAddr;

I2C3\_MCS\_R = 3;

error = I2C\_wait\_till\_done();

if (error)

return error;

I2C3\_MDR\_R = data;

I2C3\_MCS\_R = 5;

error = I2C\_wait\_till\_done();

while (I2C3\_MCS\_R & 0x40)

;

error = I2C3\_MCS\_R & 0xE;

if (error)

return error;

return 0;

}

char I2C\_Read(int slaveAddr, char memAddr, int byteCount, char\* data)

{

char error;

if (byteCount <= 0)

return -1;

I2C3\_MSA\_R = slaveAddr << 1;

I2C3\_MDR\_R = memAddr;

I2C3\_MCS\_R = 3;

error = I2C\_wait\_till\_done();

if (error)

return error;

I2C3\_MSA\_R = (slaveAddr << 1) + 1;

if (byteCount == 1)

I2C3\_MCS\_R = 7;

else

I2C3\_MCS\_R = 0xB;

error = I2C\_wait\_till\_done();

if (error)

return error;

\*data++ = I2C3\_MDR\_R;

if (--byteCount == 0)

{

while (I2C3\_MCS\_R & 0x40)

;

return 0;

}

while (byteCount > 1)

{

I2C3\_MCS\_R = 9;

error = I2C\_wait\_till\_done();

if (error)

return error;

byteCount--;

\*data++ = I2C3\_MDR\_R;

}

I2C3\_MCS\_R = 5;

error = I2C\_wait\_till\_done();

\*data = I2C3\_MDR\_R;

while (I2C3\_MCS\_R & 0x40)

;

return 0;

}

**Imu.h**

#ifndef IMU\_H\_

#define IMU\_H\_

#include "tm4c123gh6pm.h"

#include <stdbool.h>

#include <stdint.h>

#include "I2C.h"

#include <math.h>

#include "stdio.h"

#define SLAVE\_ADDR 0x68

void IMU\_Init();

void IMU\_Calibrate();

void IMU\_CalculateAcc();

float IMU\_GetAccX();

float IMU\_GetAccY();

void delayMs(int n);

#endif

**Imu.c**

#include "imu.h"

float Acc\_rawX, Acc\_rawY, Acc\_rawZ;

float Acc\_angle\_error\_x, Acc\_angle\_error\_y;

float rad\_to\_deg = 180/3.141592654;

void IMU\_Init()

{

I2C\_Write(SLAVE\_ADDR, 0x6B, 0x01); // clock 8 mhz pll with x axis gyro reference

delayMs(100);

I2C\_Write(SLAVE\_ADDR, 0x68, 0x06); // signal path reset

delayMs(100);

I2C\_Write(SLAVE\_ADDR, 0x6A, 0x00); // i2c\_if\_dis = 0

delayMs(100);

I2C\_Write(SLAVE\_ADDR, 0x1A, 0x00); //fsync and dlpf disabled

delayMs(100);

I2C\_Write(SLAVE\_ADDR, 0x19, 0x07); //sample rate set to 1 khz

delayMs(100);

I2C\_Write(SLAVE\_ADDR, 0x1B, 0x00); // +/- 250dps gyrometer configuration

delayMs(100);

I2C\_Write(SLAVE\_ADDR, 0x1C, 0x10); // +/- 2g accelerometer configuration

delayMs(100);

IMU\_Calibrate();

}

void IMU\_Calibrate()

{

char accel\_data[6];

int a;

int16\_t ax, ay, az;

for( a=0; a<200; a++)

{

I2C\_Read(SLAVE\_ADDR, 0x3B, 6, accel\_data);

ax = (int16\_t) ((accel\_data[0] << 8) | (accel\_data[1]));

ay = (int16\_t) ((accel\_data[2] << 8) | (accel\_data[3]));

az = (int16\_t) ((accel\_data[4] << 8) | (accel\_data[5]));

Acc\_rawX = (((float) ax) / 4096.0);

Acc\_rawY = (((float) ay) / 4096.0);

Acc\_rawZ = (((float) az) / 4096.0);

/\*---X---\*/

Acc\_angle\_error\_x = Acc\_angle\_error\_x + ((atan((Acc\_rawY)/sqrt(pow((Acc\_rawX),2) + pow((Acc\_rawZ),2)))\*rad\_to\_deg));

/\*---Y---\*/

Acc\_angle\_error\_y = Acc\_angle\_error\_y + ((atan(-1\*(Acc\_rawX)/sqrt(pow((Acc\_rawY),2) + pow((Acc\_rawZ),2)))\*rad\_to\_deg));

if(a==199)

{

Acc\_angle\_error\_x = Acc\_angle\_error\_x/200;

Acc\_angle\_error\_y = Acc\_angle\_error\_y/200;

}

}

}

void delayMs(int n)

{

int i, j;

for (i = 0; i < n; i++)

for (j = 0; j < 3180; j++)

{

} /\* do nothing for 1 ms \*/

}

# Challenges we faced

* Dealing with imu and calculating orientation ranges..
* Understanding i2c communication.
* Understanding how H-bridge works.
* Car assembly.
* Bluetooth module configuration.
* Determining the needed battery voltage.
* The H-bridge needed 5v for logic inputs so we used a voltage level shifter to convert 3.3v from tiva to 5v.

Video Link

https://youtu.be/eCOSvOF2HF8

# 

# 