RALPH MAGO

CPE301 – SPRING 2016

Design Assignment FINAL

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 0. | GOAL OF THE PROJECT | Y |  |
| 1. | DELIVERABLES | Y |  |
| 2. | LITERATURE SURVEY | Y |  |
| 3. | COMPONENTS | Y |  |
| 4. | SCHEMATICS | Y |  |
| 5. | INITIAL PCB | Y |  |
| 6. | IMPLEMENTATION | Y |  |
| 7. | SCREENSHOTS AND VIDEO | Y |  |
| 8. | CODE | Y |  |
| 9. | REFERENCES | Y |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | GOAL OF THE PROJECT |  |  |

* Capture gyro/accelerometer data from muscle movements in the Z-axis, rotation around the X-axis, and rotation around the Y-axis.
* Display these movement patterns using a serial terminal with values transmitted wirelessly via Bluetooth.
* Because wireless implementation is possible, this device has an application in tracking muscle twitching patterns by storing data into a mobile device.
* Fully developed version of this circuit will be similar to a surface electrode which transmits muscle movements to a serial terminal wirelessly.

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | DELIVERABLES |  |  |

This project is expected to deliver data from the gyro/accelerometer into a graphical output:

* Use **Bluetooth Terminal/Graphics** for **Android** for serial communication
  + Position: Z-axis gyro/accelerometer data (Graph 1)
  + Pitch : Measurement of rotation around X-axis (Graph 2)
  + Roll: Measurement of rotation around Y-axis (Graph 3)
* AVR MAIN C code + referenced .c and header files
* Schematics, PCB
* Demonstration of BT serial communication

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | LITERATURE SURVEY |  |  |

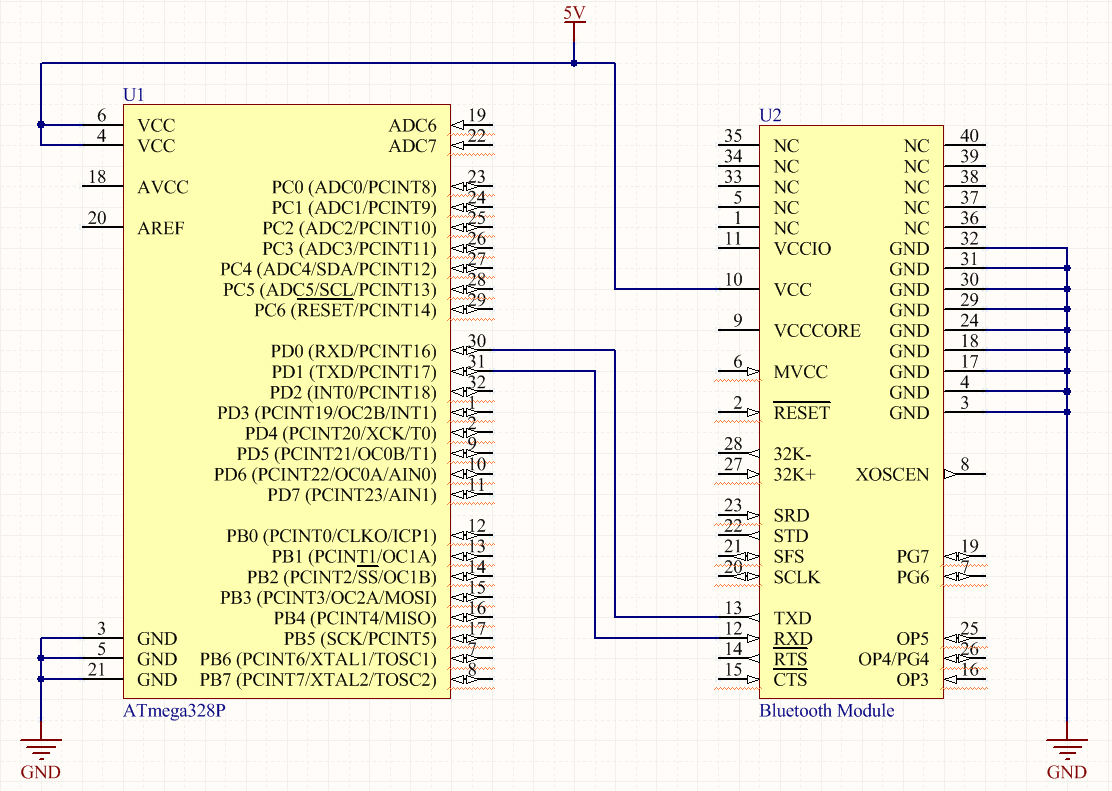
Muscle movement detection is used for finding patterns in movement for animals (species research) and humans (muscle disease). In a research study [3], diaphragm contraction of alligators was measured in a graphical output. Muscle twitching patterns in humans [4] was observed in identifying muscle contraction and relaxation for normal human subjects and subjects with tetanus.

The current state of the project allows a functional graphical representation of gyro/accelerometer movement by the user, by wrist or ankle movement. Wireless transmission using Bluetooth to send the captured accelerometer data will be useful in observing movement data over time and finding rotational movement patterns along the X, Y, and Z axes.

|  |  |  |  |
| --- | --- | --- | --- |
| 3. | COMPONENTS |  |  |

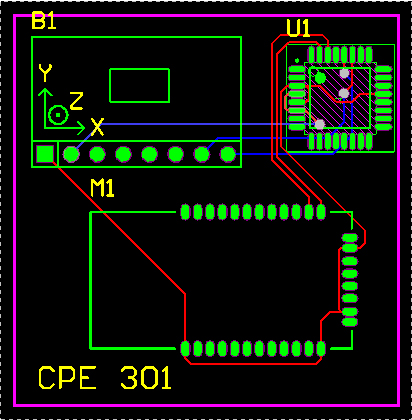
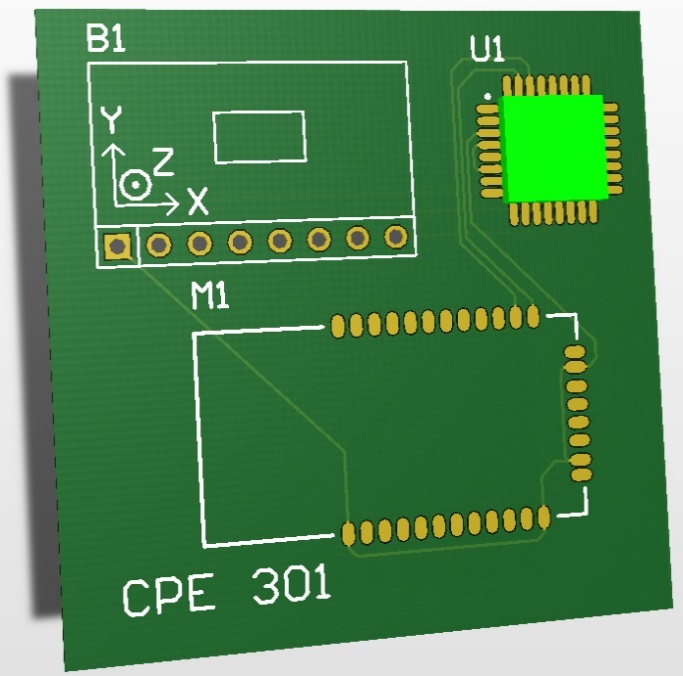
|  |  |
| --- | --- |
| Atmel Xplained Mini (328P) | 1 |
| MPU6050 Gyro/Accelerometer | 1 |
| HC-06 Bluetooth Module | 1 |
| Power Supply (3.3V, 5V) | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | SCHEMATICS |  |  |



* ATMega328P has PC4 connected to SDA and PC5 connected to SCL on MPU-6050 (gyro/accelerator)
* ATMega328P has RX and TX connected with TX and RX with HC-06 (BT Module)
* ATMega328P and MPU-6050 connected to 5V power
* BT Module connected to 3.3V power

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | INITIAL PCB |  |  |

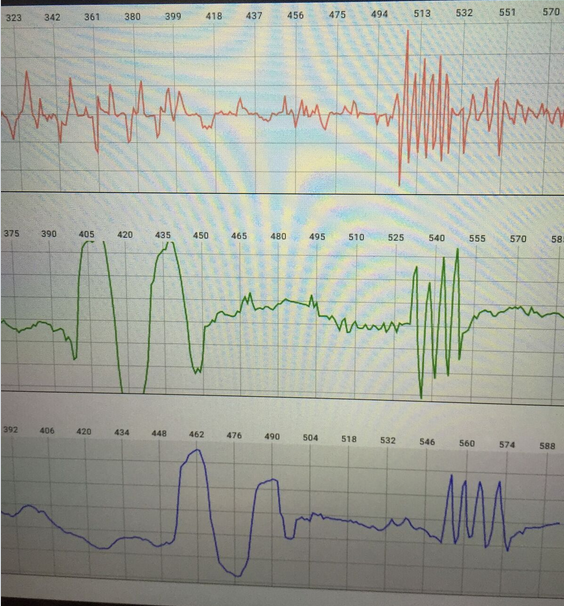
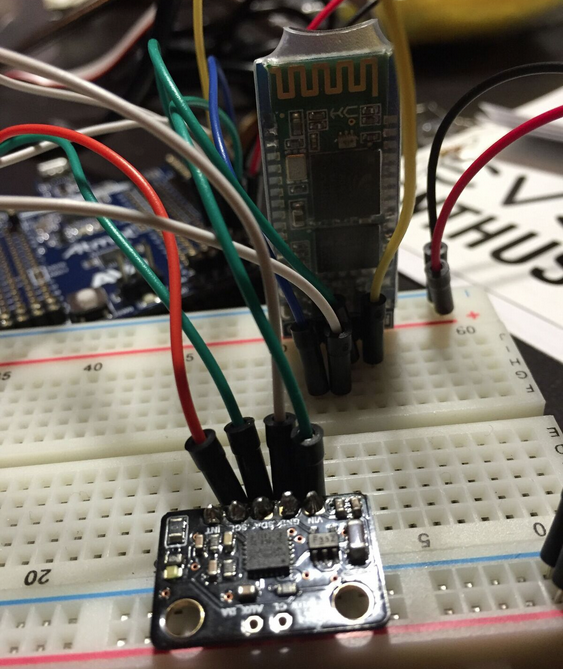
 

* **UI** represents ATMega328P
* **M1** represents HC-06 Bluetooth Module
* **B1** represents MPU-6050 Gyro/Accelerometer

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | IMPLEMENTATION |  |  |

* Prepare a Bluetooth capable device (Android) that can run a serial terminal to transmit and receive between the HC-06 Bluetooth Module and the microcontroller
* Make necessary conversions to gyro/accelerometer captured data in order for the serial terminal to get a correct output.
* Establish proper wireless connection through Bluetooth
* Enable graphs for Z-axis, Pitch, and Roll calculations
* Rotate accelerometer around and observe changes in output data.
* Interfaces:
  + ATMega328P
  + UART – **TX** and **RX** between HC-06 and Serial Terminal
    - Serial Terminal used is **Bluetooth Terminal/Graphics** on **Android**
  + I2C – HC-06 is the **slave** device

|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS AND VIDEO |  |  |



* Position along Z-axis (red)
  + Graph spikes up when moving left
  + Graph spikes down when moving right
* Side-to-side rotation about X-axis aka Pitch (green)
  + Turning around X-axis results in a vector on the Y-axis
  + Graph spikes up rolling right
  + Graph spikes down rolling left
* Up-and-down rotation about Y-axis aka Roll (blue)
  + Turning around Y-axis results in a vector on the X-axis
  + Graph spikes up pitching upwards
  + Graph spikes down pitching downwards
* Muscle Movement Detection Unit – YouTube
  + <https://www.youtube.com/watch?v=6vSMingkX18>

|  |  |  |  |
| --- | --- | --- | --- |
| 8. | CODE |  |  |

* The header files are refenced from [2]
  + **I2c.h** / **i2.c -> define functions for i2c operation**
  + **Mpu6050\_reg.h -> defines register addresses of MPU6050**
  + **Mpu6050.h / mpu6050.c -> define functions to operate mpu6050**
  + **Uart.h / uart.c -> define functions to operate UART interface for serial comm.**

/\*

\* DA7.c

\*

\* Created: 5/4/2016 12:48:55 AM

\* Author : r

\*/

// Muscle Movement Detection Unit

//

// Captures data from Gyro/Accelerometer

// Implements UART and I2C using BT

#define F\_CPU 16000000UL //clock frequency (Fosc)

#define BAUD 9600 // Baud Rate 9600, matching

#define GYRO\_SENSITIVITY 65.536 // referenced from [1]

#define M\_PI 3.14159265359 //pi

#include <avr/io.h> //AVR input/output

#include <util/setbaud.h> //requires that entry values already defined for F\_CPU and BAUD

#include <math.h>

#include <inttypes.h> //necessary to use uint types from stdint.h, and printf functions

#include <util/delay.h> //\_delay\_ms procedure for millisecond delay

#include <avr/interrupt.h> //interrupt enable for serial terminal

#include "i2c.h"

#include "mpu6050.h"

#include "mpu6050\_reg.h"

#include "uart.h"

void timer\_setup(); // referenced from [2]

void get\_time(double\* dt);

volatile double count;

const double unit\_t = 8/16000000;

int main(void){ //variable declarations and procedures referenced from [2]

sei();

uart\_init();

i2c\_init();

DDRB |= \_BV(5);

uint8\_t ret;

int16\_t accel\_buff[3], gyro\_buff[3];

double accelX, accelY, accelZ;

double gyroX, gyroY, gyroZ;

double dt;

float pitch; //angle around x axis

float roll; //angle around y axis

float pitch\_accel, roll\_accel;

float pitch\_count = 0;

float roll\_count = 0;

int count\_previous\_x = 0;

int count\_previous\_y = 0;

// initialize & test MPU5060 availability

ret = i2c\_start(MPU6050\_ADDRESS+I2C\_WRITE);

if(~ret){

PORTB |= \_BV(5);

\_delay\_ms(200);

PORTB &= ~(\_BV(5));

}

mpu6050\_init();

timer\_setup();

mpu6050\_read\_accel\_ALL(accel\_buff);

while(1){

get\_time(&dt);

mpu6050\_read\_accel\_ALL(accel\_buff);

mpu6050\_read\_gyro\_ALL(gyro\_buff);

// acceleration (m/s^2)

accelX = accel\_buff[0]\*9.8\*2/32768;

accelY = accel\_buff[1]\*9.8\*2/32768;

accelZ = accel\_buff[2]\*9.8\*2/32768;

// gyro rate (degrees/s)

gyroX = gyro\_buff[0]\*(180/M\_PI)\*1000/32768;

gyroY = gyro\_buff[1]\*(180/M\_PI)\*1000/32768;

gyroZ = gyro\_buff[2]\*(180/M\_PI)\*1000/32768;

/////////////////////////////////////////

// Complementary Filter referenced from [1]

/////////////////////////////////////////

// Integrate the gyroscope data -> int(angularSpeed) = angle

pitch = pitch + ((float)gyro\_buff[0] / GYRO\_SENSITIVITY) \* dt;

roll = roll - ((float)gyro\_buff[1] / GYRO\_SENSITIVITY) \* dt;

// Compensate for drift with accelerometer data if !bull

// Sensitivity = -2 to 2 G at 16Bit -> 2G = 32768 && 0.5G = 8192

int approx\_force\_magnitude = abs(accel\_buff[0]) + abs(accel\_buff[1]) + abs(accel\_buff[2]);

if(approx\_force\_magnitude > 8192 && approx\_force\_magnitude < 32768)

{

// Turning around the X axis results in a vector on the Y-axis

// Also known as "Pitch"

pitch\_accel = atan2f((float)accel\_buff[1], (float)accel\_buff[2]) \* 180 / M\_PI; //in degrees

pitch = pitch \* 0.98 + pitch\_accel \* 0.02;

if((pitch > 25 || pitch <-25) && count\_previous\_x == 0)

{

pitch\_count += 1;

count\_previous\_x = 1;

}

else if (!(pitch > 25 || pitch <-25)) count\_previous\_x = 0;

// Turning around the Y axis results in a vector on the X-axis

// Also known as "Roll"

roll\_accel = atan2f((float)accel\_buff[0], (float)accel\_buff[2]) \* 180 / M\_PI; //in degrees

roll = roll \* 0.98 + roll\_accel \* 0.02;

if((roll > 25 || roll <-25) && count\_previous\_y == 0)

{

roll\_count += 1;

count\_previous\_y = 1;

}

else if(!(roll > 25 || roll <-25)) count\_previous\_y = 0;

}

//App: Bluetooth Terminal/Graphics (Android)

//The string output to the serial terminal must be in this form:

//"Eval1,val2,val3\n"

//Where val1, val2, ... , are the value variables and are separated by commas.

\_delay\_ms(5);

uart\_putstring("E"); //first character of serial terminal transmission

uart\_putdouble(gyroZ); //display gyro data of Z axis

uart\_putstring(","); //separator

uart\_putdouble(pitch); //display pitch angle

uart\_putstring(","); //separator

uart\_putdouble(roll); //display roll angle

uart\_putstring(","); //separator

uart\_putdouble(gyroX); //OPTIONAL display gyro data of X axis

uart\_putstring(","); //separator

uart\_putdouble(gyroY); //OPTIONAL display gyro data of Y axis

uart\_putstring("\n"); //end of string for input values

uart\_putstring(" X-axis turns: ");

uart\_putdouble(pitch\_count); //number of pitches

uart\_putstring(" Y-axis turns: ");

uart\_putdouble(roll\_count); //number of rolls

uart\_putstring("\n");

}

}//end of main

void timer\_setup(){ //referenced from [2]

TCCR1A = 0x00;

TIMSK1 |= \_BV(TOIE1);

TCCR1B |= \_BV(CS11);

TCCR1B &= ~( \_BV(CS12) | \_BV(CS10)); // prescaler=8

}

void get\_time(double \* dt){ //referenced from [2]

cli();

uint8\_t l = TCNT1L;

uint8\_t h = TCNT1H;

uint16\_t step = h<<8 | l;

\*dt = (double)step\*5e-7 + count\*0.032768;

count = 0;

sei();

}

// timer 1 overflow interrupt handler

SIGNAL(TIMER1\_OVF\_vect){ //referenced from [2]

count += 1;

//TCNT1H = 0x00;

//TCNT1L = 0x00;

}

|  |  |  |  |
| --- | --- | --- | --- |
| 9. | REFERENCES |  |  |

[1]P. Jan. (Apr 2013). Reading a IMU Without Kalman: The Complementary Filter (1st ed.)[Online]. Available: <http://www.pieter-jan.com/node/11>

[2]Y. Jiang. (Dec 2014). MPU6050, GitHub repository [Online].

Available: <https://github.com/YifanJiangPolyU/MPU6050>

[3]T. J. Uriona, C. G. Farmer. (2008). Recruitment of the diaphragmaticus, ischiopubis and other respiratory muscles to control pitch and roll in the American alligator (*Alligator mississippiensis*) (1st ed.)[Online]. Available: <http://jeb.biologists.org/content/211/7/1141>

[4]K. Beitas *et al*. (2001). HUMAN BIOLOGY: Electrophysiological experiment (1st ed.)[Online]. Available: <http://www.pef.uni-lj.si/eprolab/comlab/sttop/sttop-bm/bm-elephy.htm#Index>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

RALPH MAGO