CPE301 – SPRING 2019

Design Assignment DA6

Student Name: Cody McDonald

Student #: 5000382538

Student Email: mcdonc4@unlv.nevada.edu

Primary Github address: https://github.com/elev8rProcrastinator/submission\_da.git

Directory: [https://github.com/elev8rProcrastinator/submission\_da/tree/master/DA6](https://github.com/elev8rProcrastinator/submission_da/tree/master/DA65)

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

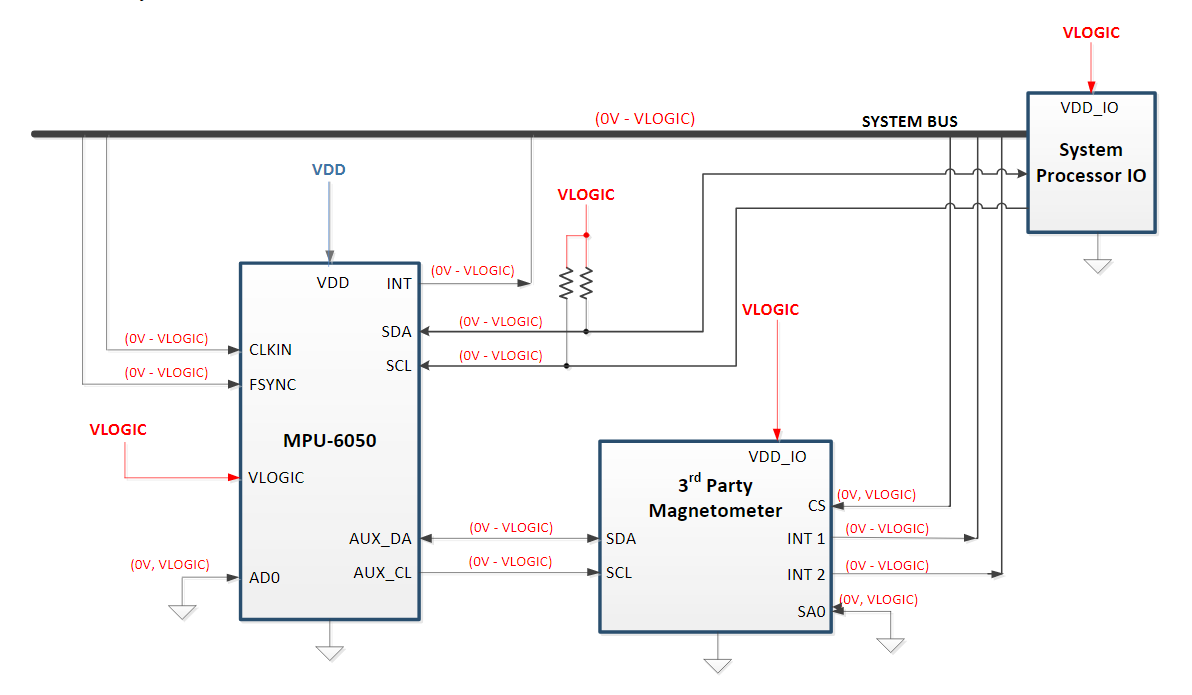
1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

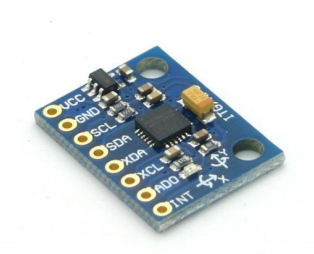
Atmini Xplained

LM35

FTDI USB to serial converter

MPU-6050:





1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

//include standard libraries

#include <avr/io.h>

#include <stdlib.h>

#include <stdio.h>

#include <util/delay.h>

#include <math.h>

//include custom libraries

#include "MPU6050\_def.h"

#include "i2c\_master.h"

#include "uart.h"

//set bits for reading and writing with the mpu6050

#define MPU6050\_WRITE 0xD0

#define MPU6050\_READ 0xD1

//Global variables

float Acc\_x, Acc\_y, Acc\_z;

float Gyro\_x, Gyro\_y, Gyro\_z;

float temp;

//Function declarations

void getValues(void);

void uart\_char(unsigned char c);

void uart\_string(char \*s);

int main(void){

char buffer[20];

char floatVal[10];

float Ax, Ay, Az;

float Gx, Gy, Gz;

//initialize our communication modes and gyroscope

i2c\_init();

init\_MPU6050();

init\_uart(9600);

while(1){

getValues(); //get raw values

//covert each raw value with their correct divisor

Ax = Acc\_x/16384.0;

Ay = Acc\_y/16384.0;

Az = Acc\_z/16384.0;

Gx = Gyro\_x/16.4;

Gy = Gyro\_y/16.4;

Gz = Gyro\_z/16.4;

USART\_SendString("\n-----------------------\n");//Spacer line

//output Ax value

*dtostrf*( Ax, 3, 2, floatVal );

*sprintf*(buffer,"Ax = %s g, ",floatVal);

USART\_SendString(buffer);

//output Ay value

*dtostrf*( Ay, 3, 2, floatVal );

*sprintf*(buffer,"Ay = %s g, ",floatVal);

USART\_SendString(buffer);

//output Az value

*dtostrf*( Az, 3, 2, floatVal );

*sprintf*(buffer,"Az = %s g\n\n",floatVal);

USART\_SendString(buffer);

//output Gx value

*dtostrf*( Gx, 3, 2, floatVal );

*sprintf*(buffer,"Gx = %s degrees/s, ",floatVal);

USART\_SendString(buffer);

//output Gy value

*dtostrf*( Gy, 3, 2, floatVal );

*sprintf*(buffer,"Gy = %s degrees/s, ",floatVal);

USART\_SendString(buffer);

//output Gz value

*dtostrf*( Gz, 3, 2, floatVal );

*sprintf*(buffer,"Gz = %s degrees/s",floatVal);

USART\_SendString(buffer);

USART\_SendString("\n-----------------------\n"); //end line break

*\_delay\_ms*(1000);

}

return 0;

}

void init\_uart(*uint16\_t* baudrate){

*uint16\_t* UBRR\_val = (F\_CPU/16)/(baudrate-1);

UBRR0H = UBRR\_val >> 8;

UBRR0L = UBRR\_val;

UCSR0B |= (1<<TXEN0) | (1<<RXEN0) | (1<<RXCIE0);

UCSR0C |= (1<<USBS0) | (3<<UCSZ00);

}

void uart\_char(unsigned char c){

while(!(UCSR0A & (1<<UDRE0))); // wait

UDR0 = c; // output character

}

void uart\_string(char \*s){

while(\*s){

uart\_char(\*s);

s++;

}

}

void init\_MPU6050(void){

*\_delay\_ms*(150);

i2c\_start(MPU6050\_WRITE); /\* Start with device write address \*/

i2c\_write(SMPLRT\_DIV); /\* Write to sample rate register \*/

i2c\_write(0x07); /\* 1KHz sample rate \*/

i2c\_stop();

i2c\_start(MPU6050\_WRITE);

i2c\_write(PWR\_MGMT\_1); /\* Write to power management register \*/

i2c\_write(0x01); /\* X axis gyroscope reference frequency \*/

i2c\_stop();

i2c\_start(MPU6050\_WRITE);

i2c\_write(CONFIG); /\* Write to Configuration register \*/

i2c\_write(0x00); /\* Fs = 8KHz \*/

i2c\_stop();

i2c\_start(MPU6050\_WRITE);

i2c\_write(GYRO\_CONFIG); /\* Write to Gyro configuration register \*/

i2c\_write(0x18); /\* Full scale range +/- 2000 degree/C \*/

i2c\_stop();

i2c\_start(MPU6050\_WRITE);

i2c\_write(INT\_ENABLE); /\* Write to interrupt enable register \*/

i2c\_write(0x01);

i2c\_stop();

}

void getValues(void){

//Start system by setting cursor

i2c\_start(MPU6050\_WRITE);

i2c\_write(ACCEL\_XOUT\_H);

i2c\_stop();

//Start reading process for each value

i2c\_start(MPU6050\_READ);

Acc\_x = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

Acc\_y = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

Acc\_z = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

temp = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

Gyro\_x = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

Gyro\_y = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

Gyro\_z = (((int)i2c\_read\_ack()<<8) | (int)i2c\_read\_ack());

//Stop reading

i2c\_stop();

}

1. **DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A**

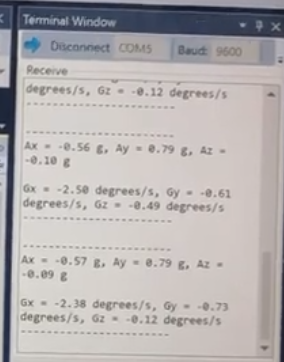
N/A

1. **SCHEMATICS**

Refer to board demos for connections of the motors and drivers to the pins on the microcontroller

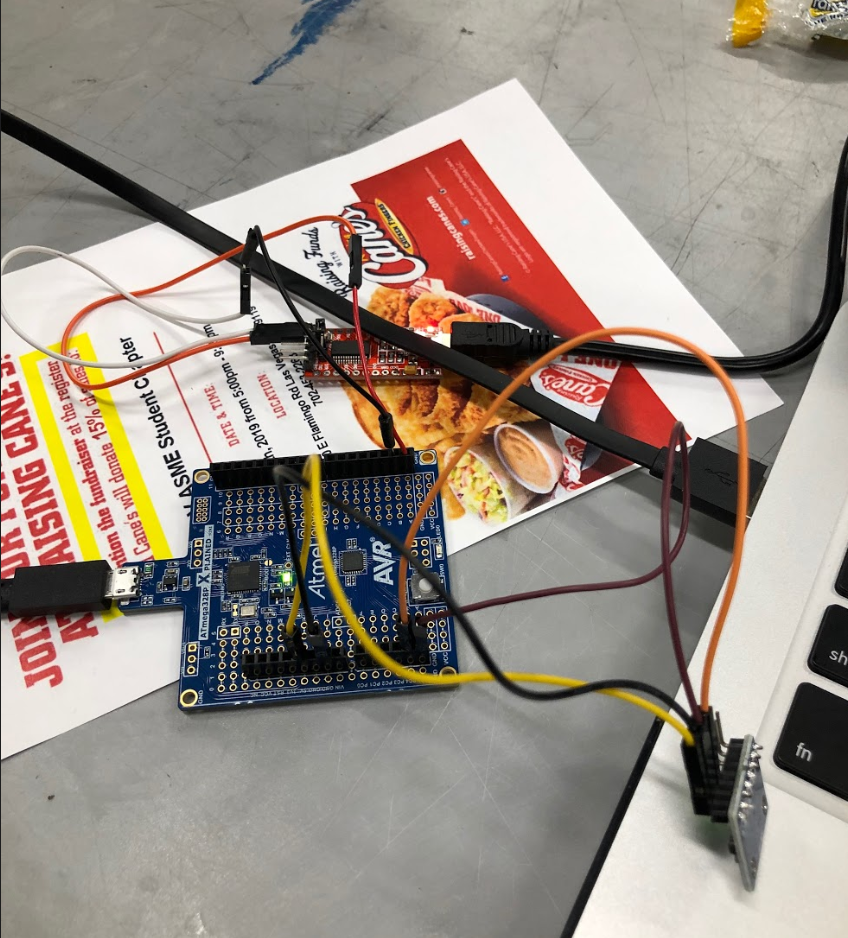
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

Terminal window of my UART communication



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**

This is the set-up of my controller setup



1. **VIDEO LINKS OF EACH DEMO**

[**https://youtu.be/YTTyRZ8qZz4**](https://youtu.be/YTTyRZ8qZz4)

1. **GITHUB LINK OF THIS DA**

https://github.com/elev8rProcrastinator/submission\_da/tree/master/DA6

**Student Academic Misconduct Policy**

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

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

Cody McDonald