CPE301 – SPRING 2019

Design Assignment 6A

Student Name: Saul Alejandro Mendoza Guzman

Student #: 2000540481

Student Email: [mendos1@unlv.nevada.edu](mailto:mendos1@unlv.nevada.edu)

Primary Github address: githbub.com/mendos1/

Directory: DA6A

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**

In this DA I used the Atmega328p, the MPU 6050, and a usb cable, and the Atmel Studio 7 software.

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

MAIN.C

#define F\_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <inttypes.h>

#include <stdlib.h>

#include <stdio.h>

#include "MPU6050\_res\_define.h"

#include "I2C\_Master\_H\_file.h"

#include "USART\_RS232\_H\_file.h"

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

void MPU6050\_Init()

{

\_delay\_ms(150); /\* Power up time >100ms \*/

I2C\_Start\_Wait(0xD0); /\* Start with device write address \*/

I2C\_Write(SMPLRT\_DIV); /\* Write to sample rate register \*/

I2C\_Write(0x07); /\* 1KHz sample rate \*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(PWR\_MGMT\_1); /\* Write to power management register \*/

I2C\_Write(0x01); /\* X axis gyroscope reference frequency \*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(CONFIG); /\* Write to Configuration register \*/

I2C\_Write(0x00); /\* Fs = 8KHz \*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(GYRO\_CONFIG); /\* Write to Gyro configuration register \*/

I2C\_Write(0x18); /\* Full scale range +/- 2000 degree/C \*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(INT\_ENABLE); /\* Write to interrupt enable register \*/

I2C\_Write(0x01);

I2C\_Stop();

}

void MPU\_Start\_Loc()

{

I2C\_Start\_Wait(0xD0); /\* I2C start with device write address \*/

I2C\_Write(ACCEL\_XOUT\_H); /\* Write start location address from where to read \*/

I2C\_Repeated\_Start(0xD1); /\* I2C start with device read address \*/

}

void Read\_RawValue()

{

MPU\_Start\_Loc(); /\* Read Gyro values \*/

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());

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\_Nack());

I2C\_Stop();

}

int main()

{

char buffer[20], float\_[10];

float Xa,Ya,Za,t;

float Xg=0,Yg=0,Zg=0;

I2C\_Init();

MPU6050\_Init();

USART\_Init(9600);

while(1)

{

Read\_RawValue();

Xa = Acc\_x/16384.0; /\* Divide raw value by sensitivity scale factor to get real values \*/

Ya = Acc\_y/16384.0;

Za = Acc\_z/16384.0;

Xg = Gyro\_x/16.4;

Yg = Gyro\_y/16.4;

Zg = Gyro\_z/16.4;

dtostrf( Xa, 3, 2, float\_ ); /\* Takes the first parameter and formts is with size 3 precission 2 and puts it in float\_ \*/

sprintf(buffer," Ax = %s g\t",float\_); /\* This is where the value is being put as a string onto the buffer variable \*/

USART\_SendString(buffer); /\* This is where it is being displayed on the serial terminal \*/

dtostrf( Ya, 3, 2, float\_ ); /\* Takes the first parameter and formts is with size 3 precission 2 and puts it in float\_v \*/

sprintf(buffer," Ay = %s g\t",float\_); /\* This is where the value is being put as a string onto the buffer variable \*/

USART\_SendString(buffer); /\* This is where it is being displayed on the serial terminal \*/

dtostrf( Za, 3, 2, float\_ ); /\* Takes the first parameter and formts is with size 3 precission 2 and puts it in float\_ \*/

sprintf(buffer," Az = %s g\t",float\_); /\* This is where the value is being put as a string onto the buffer variable \*/

USART\_SendString(buffer); /\* This is where it is being displayed on the serial terminal \*/

dtostrf( Xg, 3, 2, float\_ ); /\* Takes the first parameter and formts is with size 3 precission 2 and puts it in float\_ \*/

sprintf(buffer," Gx = %s%c/s\t",float\_,0xF8); /\* This is where the value is being put as a string onto the buffer variable \*/

USART\_SendString(buffer); /\* This is where it is being displayed on the serial terminal \*/

dtostrf( Yg, 3, 2, float\_ ); /\* Takes the first parameter and formts is with size 3 precission 2 and puts it in float\_ \*/

sprintf(buffer," Gy = %s%c/s\t",float\_,0xF8); /\* This is where the value is being put as a string onto the buffer variable \*/

USART\_SendString(buffer); /\* This is where it is being displayed on the serial terminal \*/

dtostrf( Zg, 3, 2, float\_ ); /\* Takes the first parameter and formts is with size 3 precission 2 and puts it in float\_ \*/

sprintf(buffer," Gz = %s%c/s\r\n",float\_,0xF8); /\* This is where the value is being put as a string onto the buffer variable \*/

USART\_SendString(buffer); /\* This is where it is being displayed on the serial terminal \*/

\_delay\_ms(1000); /\* This just be a delaylay for the thingys \*/

}

}

I2C\_Master\_H\_file.h

#ifndef I2C\_MASTER\_H\_FILE\_H\_ /\* Define library H file if not defined \*/

#define I2C\_MASTER\_H\_FILE\_H\_

#define F\_CPU 16000000UL /\* Define CPU clock Frequency e.g. here its 8MHz \*/

#include <avr/io.h> /\* Include AVR std. library file \*/

#include <util/delay.h> /\* Include delay header file \*/

#include <math.h> /\* Include math function \*/

#define SCL\_CLK 100000L /\* Define SCL clock frequency \*/

/\* | 7| 6| 5| 4| 3| 2| 1| 0| \*/

/\* TWI Status Register(TWSR) = |TWS7|TWS6|TWS5|TWS4|TWS3|----|TWPS1|TWPS0| \*/

/\* TWI STATUS(TWISR[7:3]) SHOW STATUS OF TWI CONTROL AND BUS \*/

/\* TWI STATUS(TWISR[1:0]) CONTROL BIT RATE PRESCALAR \*/

#define BITRATE(TWSR) ((F\_CPU/SCL\_CLK)-16)/(2\*pow(4,(TWSR&((1<<TWPS0)|(1<<TWPS1))))) /\* Define bit rate \*/

void I2C\_Init(); /\* I2C initialize function \*/

uint8\_t I2C\_Start(char); /\* I2C start function \*/

uint8\_t I2C\_Repeated\_Start(char); /\* I2C repeated start function \*/

void I2C\_Stop(); /\* I2C stop function \*/

void I2C\_Start\_Wait(char); /\* I2C start wait function \*/

uint8\_t I2C\_Write(char); /\* I2C write function \*/

char I2C\_Read\_Ack(); /\* I2C read ack function \*/

char I2C\_Read\_Nack(); /\* I2C read nack function \*/

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

I2C\_Master\_H\_file.C

#include "I2C\_Master\_H\_file.h" /\* Include I2C header file \*/

void I2C\_Init() /\* I2C initialize function \*/

{

TWBR = BITRATE(TWSR = 0x00); /\* Get bit rate register value by formula \*/

}

uint8\_t I2C\_Start(char slave\_write\_address) /\* I2C start function \*/

{

uint8\_t status; /\* Declare variable \*/

TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT); /\* Enable TWI, generate start condition and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (start condition) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status != 0x08) /\* Check weather start condition transmitted successfully or not? \*/

return 0; /\* If not then return 0 to indicate start condition fail \*/

TWDR = slave\_write\_address; /\* If yes then write SLA+W in TWI data register \*/

TWCR = (1<<TWEN)|(1<<TWINT); /\* Enable TWI and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (Write operation) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status == 0x18) /\* Check weather SLA+W transmitted & ack received or not? \*/

return 1; /\* If yes then return 1 to indicate ack received i.e. ready to accept data byte \*/

if (status == 0x20) /\* Check weather SLA+W transmitted & nack received or not? \*/

return 2; /\* If yes then return 2 to indicate nack received i.e. device is busy \*/

else

return 3; /\* Else return 3 to indicate SLA+W failed \*/

}

uint8\_t I2C\_Repeated\_Start(char slave\_read\_address) /\* I2C repeated start function \*/

{

uint8\_t status; /\* Declare variable \*/

TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT); /\* Enable TWI, generate start condition and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (start condition) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status != 0x10) /\* Check weather repeated start condition transmitted successfully or not? \*/

return 0; /\* If no then return 0 to indicate repeated start condition fail \*/

TWDR = slave\_read\_address; /\* If yes then write SLA+R in TWI data register \*/

TWCR = (1<<TWEN)|(1<<TWINT); /\* Enable TWI and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (Write operation) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status == 0x40) /\* Check weather SLA+R transmitted & ack received or not? \*/

return 1; /\* If yes then return 1 to indicate ack received \*/

if (status == 0x20) /\* Check weather SLA+R transmitted & nack received or not? \*/

return 2; /\* If yes then return 2 to indicate nack received i.e. device is busy \*/

else

return 3; /\* Else return 3 to indicate SLA+W failed \*/

}

void I2C\_Stop() /\* I2C stop function \*/

{

TWCR=(1<<TWSTO)|(1<<TWINT)|(1<<TWEN); /\* Enable TWI, generate stop condition and clear interrupt flag \*/

while(TWCR & (1<<TWSTO)); /\* Wait until stop condition execution \*/

}

void I2C\_Start\_Wait(char slave\_write\_address) /\* I2C start wait function \*/

{

uint8\_t status; /\* Declare variable \*/

while (1)

{

TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT); /\* Enable TWI, generate start condition and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (start condition) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status != 0x08) /\* Check weather start condition transmitted successfully or not? \*/

continue; /\* If no then continue with start loop again \*/

TWDR = slave\_write\_address; /\* If yes then write SLA+W in TWI data register \*/

TWCR = (1<<TWEN)|(1<<TWINT); /\* Enable TWI and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (Write operation) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status != 0x18 ) /\* Check weather SLA+W transmitted & ack received or not? \*/

{

I2C\_Stop(); /\* If not then generate stop condition \*/

continue; /\* continue with start loop again \*/

}

break; /\* If yes then break loop \*/

}

}

uint8\_t I2C\_Write(char data) /\* I2C write function \*/

{

uint8\_t status; /\* Declare variable \*/

TWDR = data; /\* Copy data in TWI data register \*/

TWCR = (1<<TWEN)|(1<<TWINT); /\* Enable TWI and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (Write operation) \*/

status = TWSR & 0xF8; /\* Read TWI status register with masking lower three bits \*/

if (status == 0x28) /\* Check weather data transmitted & ack received or not? \*/

return 0; /\* If yes then return 0 to indicate ack received \*/

if (status == 0x30) /\* Check weather data transmitted & nack received or not? \*/

return 1; /\* If yes then return 1 to indicate nack received \*/

else

return 2; /\* Else return 2 to indicate data transmission failed \*/

}

char I2C\_Read\_Ack() /\* I2C read ack function \*/

{

TWCR=(1<<TWEN)|(1<<TWINT)|(1<<TWEA); /\* Enable TWI, generation of ack and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (read operation) \*/

return TWDR; /\* Return received data \*/

}

char I2C\_Read\_Nack() /\* I2C read nack function \*/

{

TWCR=(1<<TWEN)|(1<<TWINT); /\* Enable TWI and clear interrupt flag \*/

while (!(TWCR & (1<<TWINT))); /\* Wait until TWI finish its current job (read operation) \*/

return TWDR; /\* Return received data \*/

}

MPU6050\_RES\_DEFINE.H

#ifndef MPU6050\_RES\_DEFINE\_H\_

#define MPU6050\_RES\_DEFINE\_H\_

#include <avr/io.h>

#define ACCEL\_CONFIG 0x1C

#define ACCEL\_XOUT\_H 0x3B

#define ACCEL\_XOUT\_L 0x3C

#define ACCEL\_YOUT\_H 0x3D

#define ACCEL\_YOUT\_L 0x3E

#define ACCEL\_ZOUT\_H 0x3F

#define ACCEL\_ZOUT\_L 0x40

#define BANK\_SEL 0x6D

#define CONFIG 0x1A

#define DMP\_CFG\_1 0x70

#define DMP\_CFG\_2 0x71

#define DMP\_INT\_STATUS 0x39

#define EXT\_SENS\_DATA\_00 0x49

#define EXT\_SENS\_DATA\_01 0x4A

#define EXT\_SENS\_DATA\_02 0x4B

#define EXT\_SENS\_DATA\_03 0x4C

#define EXT\_SENS\_DATA\_04 0x4D

#define EXT\_SENS\_DATA\_05 0x4E

#define EXT\_SENS\_DATA\_06 0x4F

#define EXT\_SENS\_DATA\_07 0x50

#define EXT\_SENS\_DATA\_08 0x51

#define EXT\_SENS\_DATA\_09 0x52

#define EXT\_SENS\_DATA\_10 0x53

#define EXT\_SENS\_DATA\_11 0x54

#define EXT\_SENS\_DATA\_12 0x55

#define EXT\_SENS\_DATA\_13 0x56

#define EXT\_SENS\_DATA\_14 0x57

#define EXT\_SENS\_DATA\_15 0x58

#define EXT\_SENS\_DATA\_16 0x59

#define EXT\_SENS\_DATA\_17 0x5A

#define EXT\_SENS\_DATA\_18 0x5B

#define EXT\_SENS\_DATA\_19 0x5C

#define EXT\_SENS\_DATA\_20 0x5D

#define EXT\_SENS\_DATA\_21 0x5E

#define EXT\_SENS\_DATA\_22 0x5F

#define EXT\_SENS\_DATA\_23 0x60

#define FF\_DUR 0x1E

#define FF\_THR 0x1D

#define FIFO\_COUNTH 0x72

#define FIFO\_COUNTL 0x73

#define FIFO\_EN 0x23

#define FIFO\_R\_W 0x74

#define GYRO\_CONFIG 0x1B

#define GYRO\_XOUT\_H 0x43

#define GYRO\_XOUT\_L 0x44

#define GYRO\_YOUT\_H 0x45

#define GYRO\_YOUT\_L 0x46

#define GYRO\_ZOUT\_H 0x47

#define GYRO\_ZOUT\_L 0x48

#define I2C\_MST\_CTRL 0x24

#define I2C\_MST\_DELAY\_CTRL 0x67

#define I2C\_MST\_STATUS 0x36

#define I2C\_SLV0\_ADDR 0x25

#define I2C\_SLV0\_CTRL 0x27

#define I2C\_SLV0\_DO 0x63

#define I2C\_SLV0\_REG 0x26

#define I2C\_SLV1\_ADDR 0x28

#define I2C\_SLV1\_CTRL 0x2A

#define I2C\_SLV1\_DO 0x64

#define I2C\_SLV1\_REG 0x29

#define I2C\_SLV2\_ADDR 0x2B

#define I2C\_SLV2\_CTRL 0x2D

#define I2C\_SLV2\_DO 0x65

#define I2C\_SLV2\_REG 0x2C

#define I2C\_SLV3\_ADDR 0x2E

#define I2C\_SLV3\_CTRL 0x30

#define I2C\_SLV3\_DO 0x66

#define I2C\_SLV3\_REG 0x2F

#define I2C\_SLV4\_ADDR 0x31

#define I2C\_SLV4\_CTRL 0x34

#define I2C\_SLV4\_DI 0x35

#define I2C\_SLV4\_DO 0x33

#define I2C\_SLV4\_REG 0x32

#define INT\_ENABLE 0x38

#define INT\_PIN\_CFG 0x37

#define INT\_STATUS 0x3A

#define MEM\_R\_W 0x6F

#define MEM\_START\_ADDR 0x6E

#define MOT\_DETECT\_CTRL 0x69

#define MOT\_DETECT\_STATUS 0x61

#define MOT\_DUR 0x20

#define MOT\_THR 0x1F

#define PWR\_MGMT\_1 0x6B

#define PWR\_MGMT\_2 0x6C

#define SIGNAL\_PATH\_RESET 0x68

#define SMPLRT\_DIV 0x19

#define TEMP\_OUT\_H 0x41

#define TEMP\_OUT\_L 0x42

#define USER\_CTRL 0x6A

#define WHO\_AM\_I 0x75

#define XA\_OFFS\_H 0x06

#define XA\_OFFS\_L\_TC 0x07

#define XG\_OFFS\_TC 0x00

#define XG\_OFFS\_USRH 0x13

#define XG\_OFFS\_USRL 0x14

#define X\_FINE\_GAIN 0x03

#define YA\_OFFS\_H 0x08

#define YA\_OFFS\_L\_TC 0x09

#define YG\_OFFS\_TC 0x01

#define YG\_OFFS\_USRH 0x15

#define YG\_OFFS\_USRL 0x16

#define Y\_FINE\_GAIN 0x04

#define ZA\_OFFS\_H 0x0A

#define ZA\_OFFS\_L\_TC 0x0B

#define ZG\_OFFS\_TC 0x02

#define ZG\_OFFS\_USRH 0x17

#define ZG\_OFFS\_USRL 0x18

#define ZRMOT\_DUR 0x22

#define ZRMOT\_THR 0x21

#define Z\_FINE\_GAIN 0x05

#endif /\* MPU6050\_RES\_DEFINE\_H\_ \*/

USART\_RS232\_H\_file.h

#ifndef USART\_RS232\_H\_FILE\_H\_ /\* Define library H file if not defined \*/

#define USART\_RS232\_H\_FILE\_H\_

#define F\_CPU 16000000UL /\* Define CPU clock Frequency e.g. here its 8MHz \*/

#include <avr/io.h> /\* Include AVR std. library file \*/

#define BAUD\_PRESCALE (((F\_CPU / (BAUDRATE \* 16UL))) - 1) /\* Define prescale value \*/

void USART\_Init(unsigned long); /\* USART initialize function \*/

char USART\_RxChar(); /\* Data receiving function \*/

void USART\_TxChar(char); /\* Data transmitting function \*/

void USART\_SendString(char\*); /\* Send string of USART data function \*/

#endif /\* USART\_RS232\_H\_FILE\_H\_ \*/

USART\_RS232\_H\_file.C

#include "USART\_RS232\_H\_file.h" /\* Include USART header file \*/

void USART\_Init(unsigned long BAUDRATE) /\* USART initialize function \*/

{

UCSR0B = (1<<RXEN0)|(1<<TXEN0); /\* Enable USART transmitter and receiver \*/

/\* Write USCRC for 8 bit data and 1 stop bit \*/

UBRR0L = BAUD\_PRESCALE; /\* Load UBRRL with lower 8 bit of prescale value \*/

UBRR0H = (BAUD\_PRESCALE >> 8); /\* Load UBRRH with upper 8 bit of prescale value \*/

}

char USART\_RxChar() /\* Data receiving function \*/

{

while (!(UCSR0A & (1 << RXC0))); /\* Wait until new data receive \*/

return(UDR0); /\* Get and return received data \*/

}

void USART\_TxChar(char data) /\* Data transmitting function \*/

{

UDR0 = data; /\* Write data to be transmitting in UDR \*/

while (!(UCSR0A & (1<<UDRE0))); /\* Wait until data transmit and buffer get empty \*/

}

void USART\_SendString(char \*str) /\* Send string of USART data function \*/

{

int i=0;

while (str[i]!=0)

{

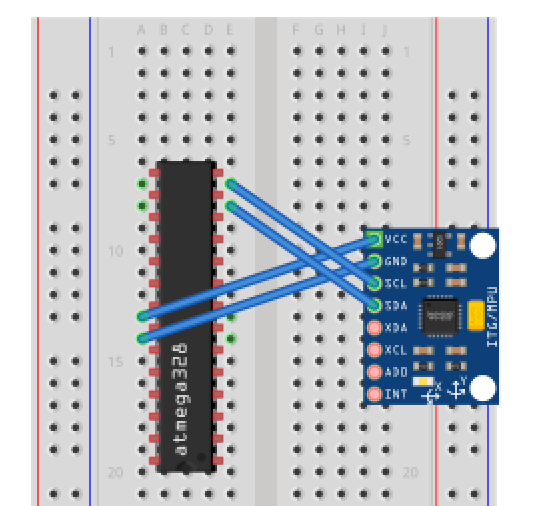
USART\_TxChar(str[i]); /\* Send each char of string till the NULL \*/

i++;

}

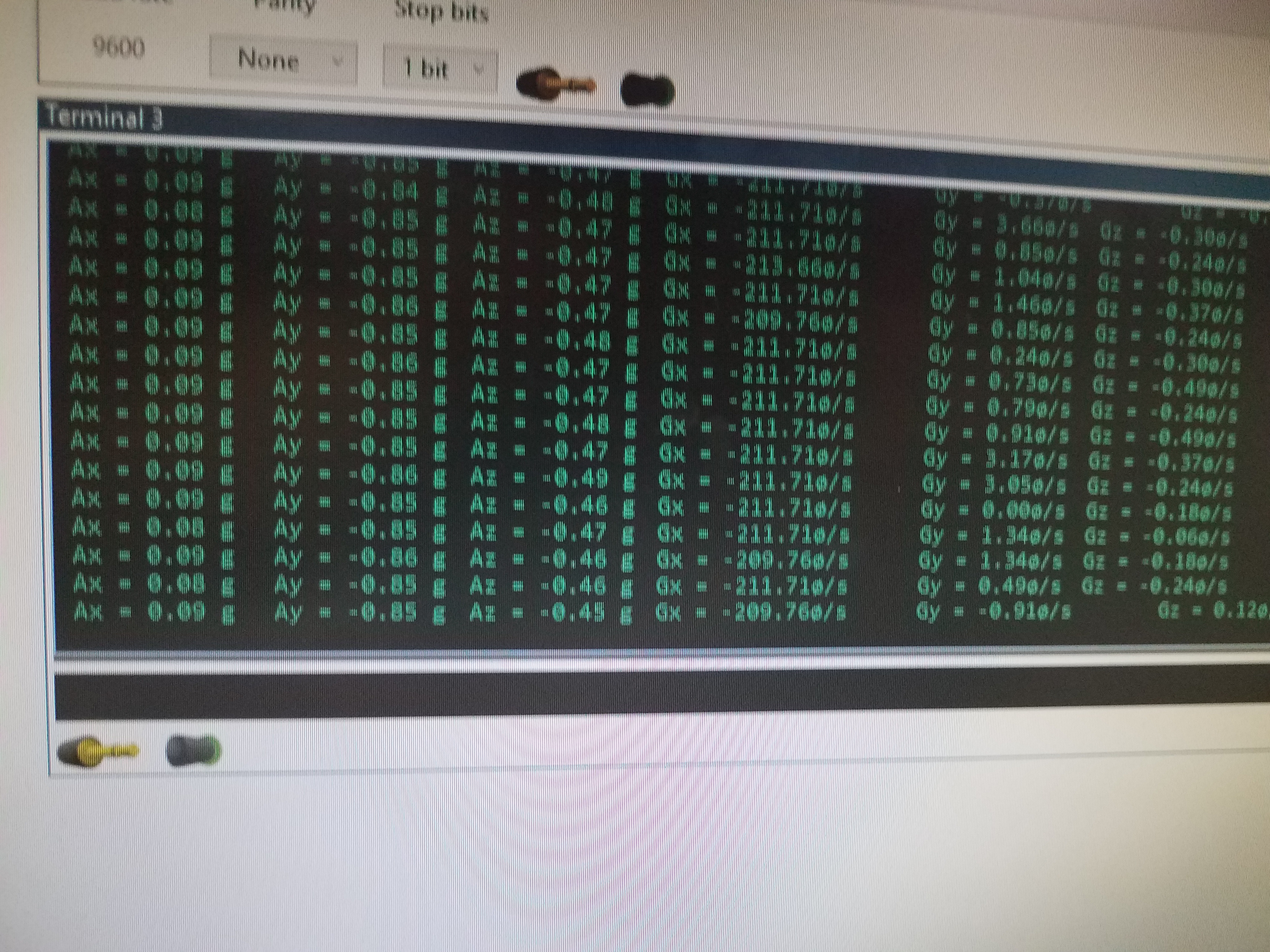
}

1. **SCHEMATICS**



Use fritzing.org

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



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**
2. **VIDEO LINKS OF EACH DEMO**

<https://www.youtube.com/watch?v=xRggt_VyhUQ>

1. **GITHUB LINK OF THIS DA**

<https://github.com/mendos1/subnission_da>

**Student Academic Misconduct Policy**

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

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

NAME OF THE STUDENT