CPE301 – SPRING 2020

Design Assignment 6

Student Name: Xianjie Cao

Student #: 5004222179

Student Email: [caox2@unlv.nevada.edu](mailto:caox2@unlv.nevada.edu)

Primary Github address: <https://github.com/c1029324620/Mocha.git>

Directory: Mocha/DesignAssignments/Lab6

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

Atmel Studio 7: Assembler, debugger and simulator

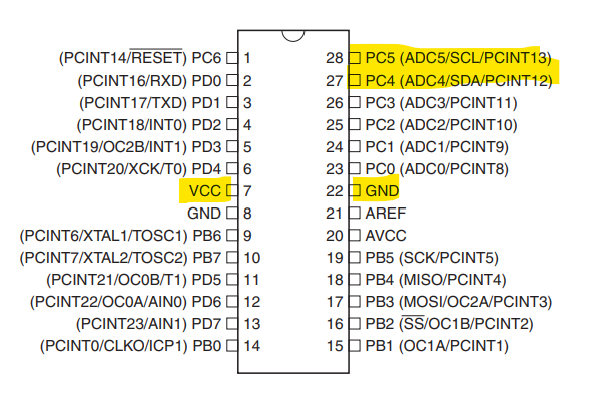
Atmega328pb-Xmini

Jumper wires

Breadboard

MPU6050

Multifunctional Shield



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

#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 <inttypes.h> /\* Include integer type header file \*/

#include <stdlib.h> /\* Include standard library file \*/

#include <stdio.h>

#include <math.h> /\* Include standard library file \*/

#include "MPU6050\_def.h" /\* Include MPU6050 register define file \*/

#include "i2c\_master.h" /\* Include I2C Master header file \*/

#include "uart.h" /\* Include USART header file \*/

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

void MPU6050\_Init() /\* Gyro initialization function \*/

{

*\_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());

Temperature = (((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()

{

//Variable Declarations

char buffer[20];

char float\_val[10];

float X\_a,Y\_a,Z\_a; //Acceleration Values X-Y-Z

float X\_g=0,Y\_g=0,Z\_g=0; //Gyro Values X-Y-Z

I2C\_Init(); //Initialize I2C

MPU6050\_Init(); //Initialize MPU6050

USART\_Init(9600); //Initialize UART

while(1)

{

Read\_RawValue(); //Call Read\_RawValue Function

X\_a = Acc\_x/16384.0; //Acceleration-X

Y\_a = Acc\_y/16384.0; //Acceleration-Y

Z\_a = Acc\_z/16384.0; //Acceleration-Z

X\_g = Gyro\_x/16.4; //Gyro-X

Y\_g = Gyro\_y/16.4; //Gyro-Y

Z\_g = Gyro\_z/16.4; //Gyro-Z

//Read/Print Acceleration-X

*dtostrf*( X\_a, 3, 2, float\_val );

*sprintf*(buffer," Ax = %s g\t",float\_val);

USART\_SendString(buffer);

//Read/Print Acceleration-Y

*dtostrf*( Y\_a, 3, 2, float\_val );

*sprintf*(buffer," Ay = %s g\t",float\_val);

USART\_SendString(buffer);

//Read/Print Acceleration-Z

*dtostrf*( Z\_a, 3, 2, float\_val );

*sprintf*(buffer," Az = %s g\t",float\_val);

USART\_SendString(buffer);

//Read/Print Gyro-X

*dtostrf*( X\_g, 3, 2, float\_val );

*sprintf*(buffer," Gx = %s%c/s\t",float\_val,0xF8);

USART\_SendString(buffer);

//Read/Print Gyro-Y

*dtostrf*( Y\_g, 3, 2, float\_val );

*sprintf*(buffer," Gy = %s%c/s\t",float\_val,0xF8);

USART\_SendString(buffer);

//Read/Print Gyro-Z

*dtostrf*( Z\_g, 3, 2, float\_val);

*sprintf*(buffer," Gz = %s%c/s\r\n",float\_val,0xF8);

USART\_SendString(buffer);

*\_delay\_ms*(1000);

}

}//End Main

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

#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 <inttypes.h> /\* Include integer type header file \*/

#include <stdlib.h> /\* Include standard library file \*/

#include <stdio.h>

#include <math.h> /\* Include standard library file \*/

#include "MPU6050\_def.h" /\* Include MPU6050 register define file \*/

#include "i2c\_master.h" /\* Include I2C Master header file \*/

#include "uart.h" /\* Include USART header file \*/

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

#define ACCELEROMETER\_SENSITIVITY 16384.0

#define GYROSCOPE\_SENSITIVITY 16.4

#define dt 0.01 // 10 ms sample rate!

void MPU6050\_Init() /\* Gyro initialization function \*/

{

*\_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());

Temperature = (((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();

}

void getAngle(int Ax, int Ay, int Az)

{

double x = Ax;

double y = Ay;

double z = Az;

pitch = *atan*(x / *sqrt*((y\*y) + (z\*z))); //pitch calculation

roll = *atan*(y/ *sqrt*((x\*x) + (z\*z))); //roll calculation

//converting radians into degrees

pitch = pitch \* (180.0 / 3.14);

roll = roll \* (180.0 / 3.14);

}

void ComplementaryFilter()

{

float pitchAcc, rollAcc;

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

pitch += ((float)Gyro\_x / GYROSCOPE\_SENSITIVITY) \* dt;

// Angle around the X-axis

roll -= ((float)Gyro\_y / GYROSCOPE\_SENSITIVITY) \* dt;

// Angle around the Y-axis

// Compensate for drift with accelerometer data if !bullshit

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

int forceMagnitudeApprox = *abs*(Acc\_x) + *abs*(Acc\_y) + *abs*(Acc\_z);

if (forceMagnitudeApprox > 8192 && forceMagnitudeApprox < 32768)

{

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

pitchAcc = *atan2f*((float)Acc\_y, (float)Acc\_z) \* 180 / *M\_PI*;

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

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

rollAcc = *atan2f*((float)Acc\_x, (float)Acc\_z) \* 180 / *M\_PI*;

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

}

}

int main()

{

//Variable Declarations

char buffer[20];

char float\_val[10];

//float X\_a,Y\_a,Z\_a; //Acceleration Values X-Y-Z

//float X\_g=0,Y\_g=0,Z\_g=0; //Gyro Values X-Y-Z

I2C\_Init(); //Initialize I2C

MPU6050\_Init(); //Initialize MPU6050

USART\_Init(9600); //Initialize UART

//10ms sample

TCCR1B = (1<< CS10); //prescaler of 8

TCNT1 = 0;

while(1)

{

if(TCNT1 == 20000)

{

Read\_RawValue(); //Call Read\_RawValue Function

/\* X\_a = Acc\_x/16384.0; //Acceleration-X

Y\_a = Acc\_y/16384.0; //Acceleration-Y

Z\_a = Acc\_z/16384.0; //Acceleration-Z

X\_g = Gyro\_x/16.4; //Gyro-X

Y\_g = Gyro\_y/16.4; //Gyro-Y

Z\_g = Gyro\_z/16.4; //Gyro-Z\*/

getAngle(Acc\_x, Acc\_y, Acc\_z);

ComplementaryFilter();

//Read/Print pitch

*dtostrf*(pitch, 3, 2, float\_val );

*sprintf*(buffer," Pitch = %s\t",float\_val);

USART\_SendString(buffer);

//Read/Print Gyro-Z

*dtostrf*(roll, 3, 2, float\_val);

*sprintf*(buffer," Roll = %s\n",float\_val);

USART\_SendString(buffer);

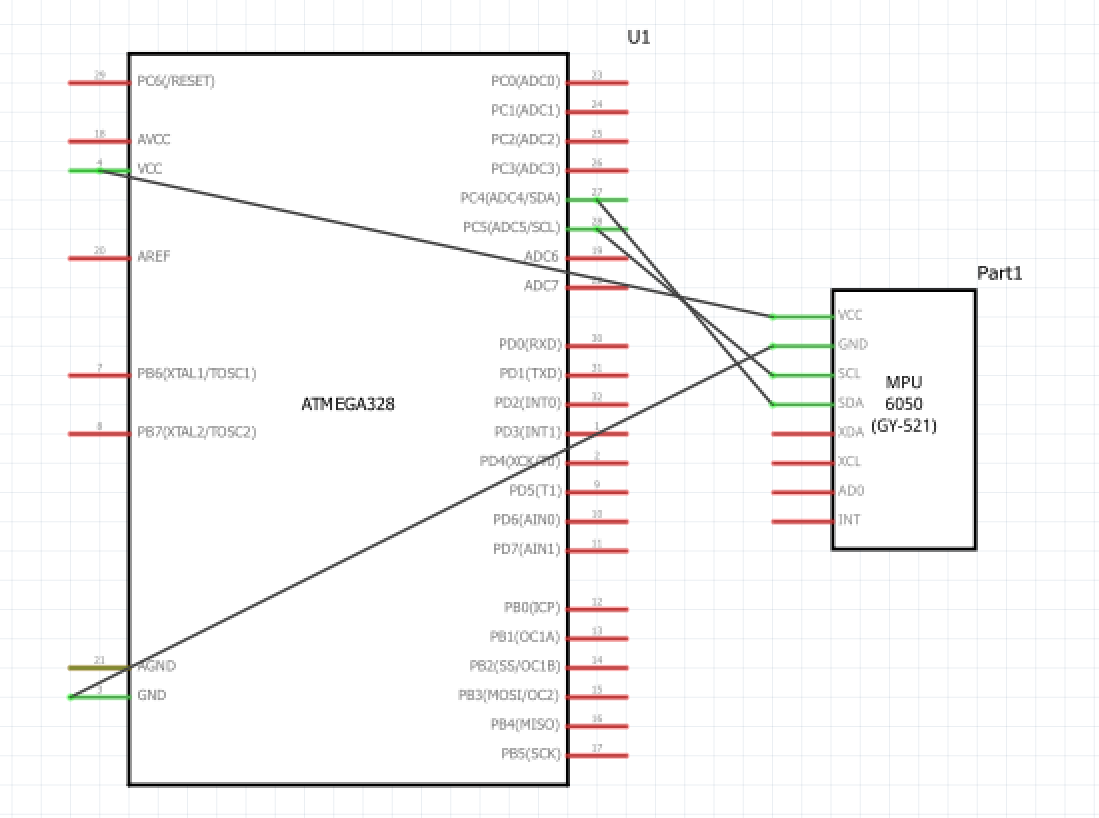
TCNT1 = 0; //reset timer counter

}

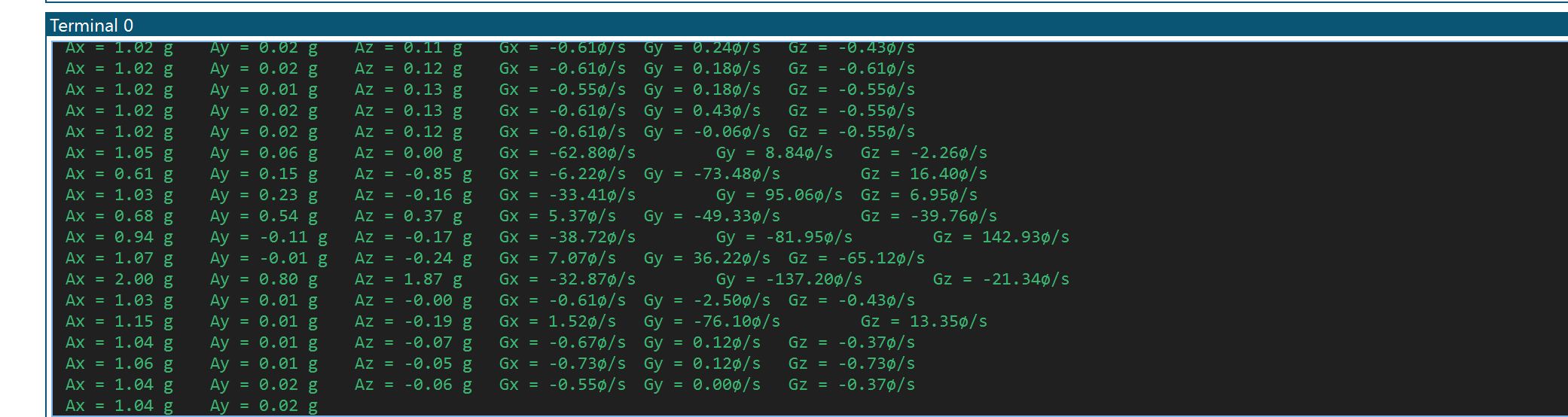
}

}//End Main

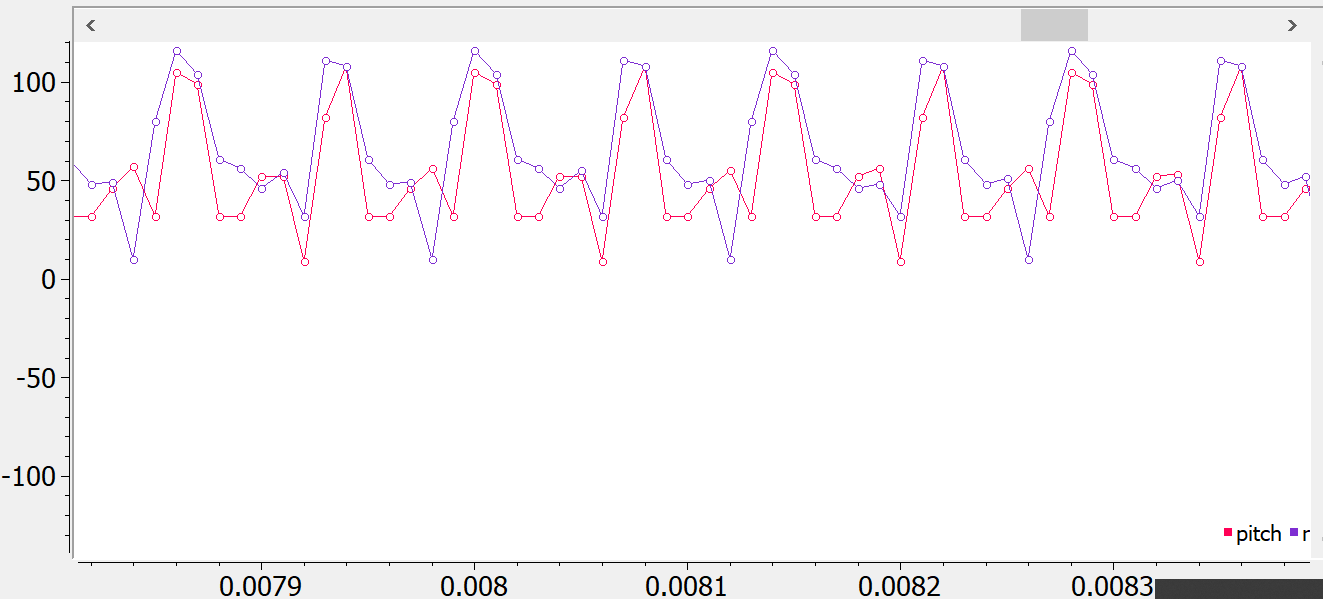
1. **SCHEMATICS**



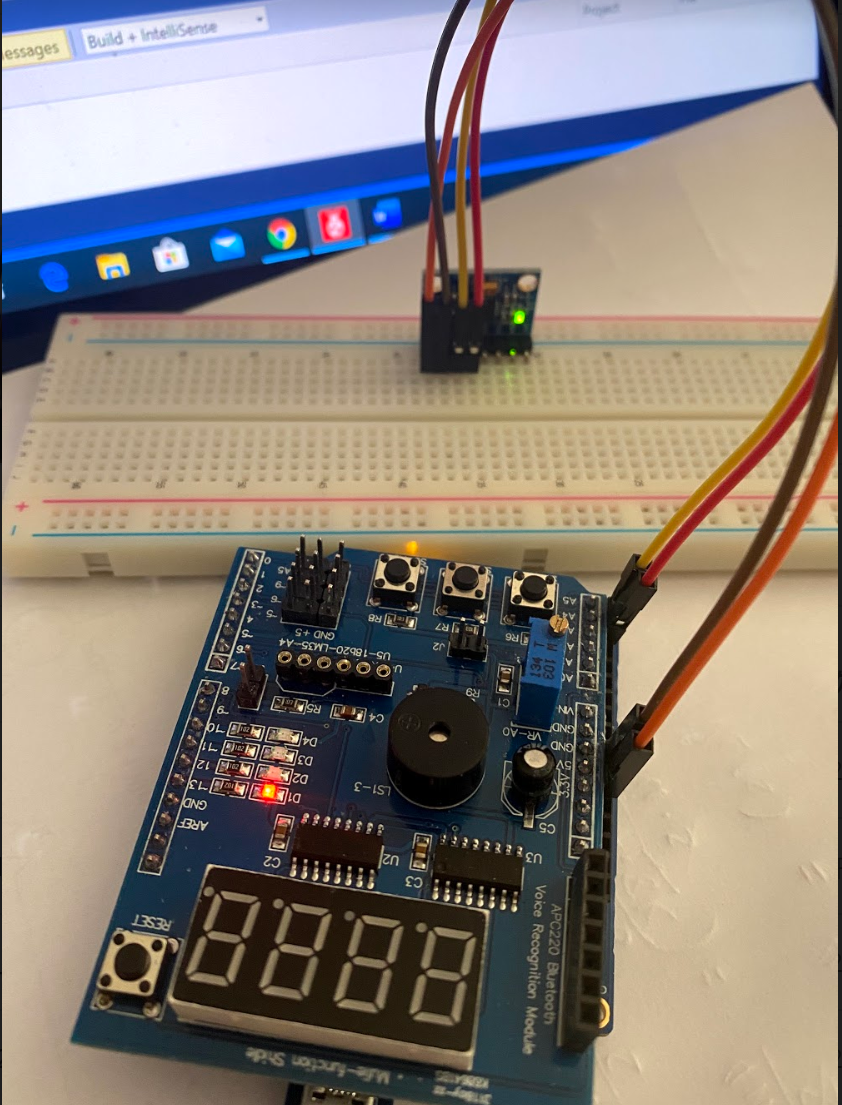
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

**Task 1:**

**Task 2:**



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



1. **VIDEO LINKS OF EACH DEMO**

Task 1: <https://youtu.be/Uiu1zLGxVh8>

Task 2: <https://youtu.be/01EThdNeow0>

1. **GITHUB LINK OF THIS DA**

**Student Academic Misconduct Policy**

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

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

Xianjie Cao