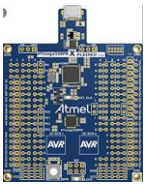
CPE301 – SPRING 2019

Design Assignment 6

Student Name: Meral Abu-Jaser
Student #: 5003137888
Student Email: abujaser@unlv.nevada.edu
Primary Github address: https://github.com/MeralAbuJaser/Submission_da.git
Directory: https://github.com/MeralAbuJaser/Submission_da/tree/master/DA6

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

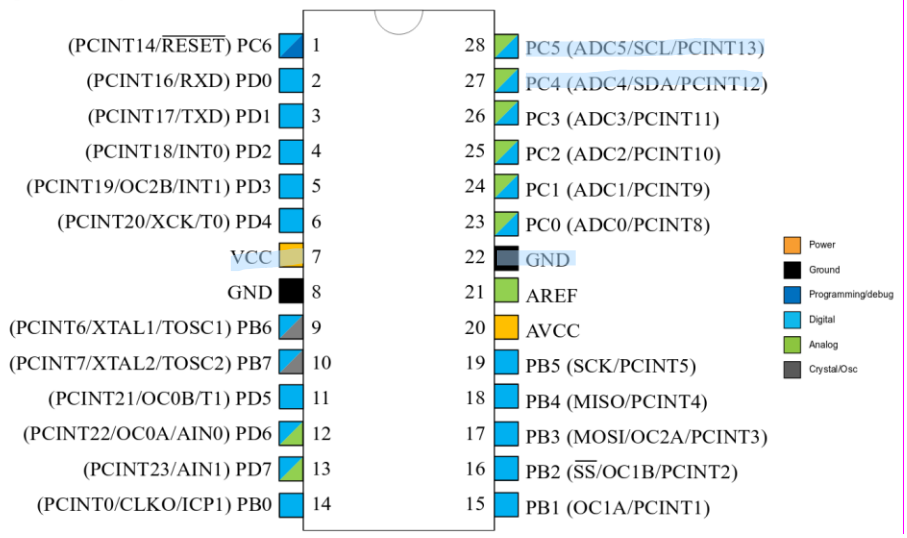


Atmega 328pb

Atmel Studio 7.0
-debugger
-simulator
-assembler
-terminal window

Pin-out

Figure 5-1. 28-pin PDIP



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

```
/*
 * DA6_TASK1.c
 *
 * Created: 5/6/2020 2:52:45 AM
 * Author : Meral
 */
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <util/twi.h>

#define SCL_CLK 100000UL
#define BAUD (((F_CPU / (BAUDRATE * 16UL))) - 1)
#define BITRATE ((((F_CPU / SCL_CLK) / 1) - 16 ) / 2)
float Acc_x, Acc_y, Acc_z, Temp_out, Gyro_x, Gyro_y, Gyro_z;
void USART_Init(unsigned long BAUDRATE)     {
	UCSR0B |= (1 << RXEN0) | (1 << TXEN0);      //enable USART transmitter and receiver
	UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);//write USCRC and 1 stop bit
	UBRR0L = BAUD;                                          //load UBRRL
	UBRR0H = (BAUD >> 8);                              //load UBRRH
}
char USART_Rx(){
	while (!(UCSR0A & (1 << RXC0)));//Wait for new data
	return(UDR0);  //return received data
}
void USART_Tx(char data){
	UDR0 = data;                                  //Write data
	while (!(UCSR0A & (1<<UDRE0)));     //data transmit? buffer set to empty
}
void USART_SendString(char *str){
	int i=0;
	while (str[i]!=0){
		USART_Tx(str[i]);
		i++;
	}
}
void I2C_Init(void){
	TWBR0 = (uint8_t)BITRATE;
}
uint8_t I2C_Start(uint8_t slave_write_address){
	uint8_t status; /* Declare variable */
	TWCR0 = 0;/*reset TWI */
	TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);// transmit START condition
	while( !(TWCR0 & (1<<TWINT)) );/* Enable TWI, generate start condition and clear interrupt flag */
	if((TWSR0 & 0xF8) != TW_START)
		return 1;

	TWDR0  = slave_write_address; /* If yes then write SLA+W in TWI data register */
	TWCR0 = (1<<TWEN)|(1<<TWINT);/* Enable TWI and clear interrupt flag */

	while( !(TWCR0 & (1<<TWINT)) );     /* Wait until TWI finish its current job (Write operation) */
		status = TWSR0 & 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 & nackreceived or not? */
		return 1;/* If yes then return 1 to indicate nackreceived */
	else
		return 2;
}
```

```
uint8_t I2C_write(uint8_t data){
	uint8_t status;
	TWDR0  = data;
	TWCR0=(1<<TWEN)|(1<<TWINT);
	while( !(TWCR0 & (1<<TWINT)));
		status = TWSR0 & 0xF8;
	if(status==0x28)
			return 0;
	if(status == 0x28)
			return 0;
	if(status == 0x30)
		return 1;
	else
		return 2;
}
/* I2C read ack function */
uint8_t i2c_read_ack(void){
	TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);/* Enable TWI, generation of ack and clear interrupt flag */
	while( !(TWCR0& (1<<TWINT)) );/* Wait until TWI finish its current job (read operation) */
	return TWDR0;  /* Return received data */
}
/* I2C read nack function */
uint8_t i2c_read_nack(void){
	TWCR0 = (1<<TWINT) | (1<<TWEN);      /* Enable TWI and clear interrupt flag */
	while( !(TWCR0 & (1<<TWINT)) );      /* Wait until TWI finish its current job (read operation) */
	return TWDR0;  /* Return received data */
}

void I2C_stop(void){
	TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);/* Enable TWI, generate stop condition and clear interrupt
flag */
}

void MPU6050_Init(void){
	_delay_ms(150);       /* Power up time >100ms */
	I2C_Start(0xD0); /* Start with device write address */
	I2C_write(0x19); /* Write to sample rate register */
	I2C_write(0x07); /* 1KHz sample rate */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x6B); /* Write to power management register */
	I2C_write(0x01); /* X axis gyroscope reference frequency */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x1A);/* Write to Configuration register */
	I2C_write(0x00);      /* Fs = 8KHz */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x1B); /* Write to Gyro configuration register */
	I2C_write(0x18);             /* Full scale range +/-2000 degree/C */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x38);
	I2C_write(0x01);
	I2C_stop();
}
```

```
void MPU_Start_Loc(){
	I2C_Start(0xD0);
	I2C_write(0x3B);
	I2C_stop();
	I2C_Start(0xD1);
}
void Read_RawValue(void){
	MPU_Start_Loc();
	//take in X,y,z accelerometer value and Xg,Yg,Zg gyro value
	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_ack());
	I2C_stop();
}
int main(void){
	char buffer[20],float_[10];
	float Xa, Ya ,Za;
	float Xg =0, Yg = 0, Zg =0;
	I2C_Init();/* Initialize I2C */
	MPU6050_Init();/* Initialize MPU6050 */
	USART_Init(9600); /* Initialize USART with 9600 baud rate */

	while(1){
		Read_RawValue();
		Xa = Acc_x/16384.0;   //divide raw value by sensitivity scale factor to get actual values
		Ya = Acc_y/16384.0;   //divide raw value by sensitivity scale factor to get actual values
		Za = Acc_z/16384.0;   //divide raw value by sensitivity scale factor to get actual values
		Xg = Gyro_x/16.4;     //divide raw value by sensitivity scale factor to get actual values
		Yg = Gyro_y/16.4;     //divide raw value by sensitivity scale factor to get actual values
		Zg = Gyro_z/16.4;     //divide raw value by sensitivity scale factor to get actual values

		USART_SendString("\n");
		dtostrf( Xa, 3, 2, float_ );
		//Take values in buffer to send all parameters over USART
		sprintf(buffer,"Xa = %s g, ",float_);
		USART_SendString(buffer);
		dtostrf(Ya, 3, 2, float_ );
		//Take values in buffer to send all parameters over USART
		sprintf(buffer,"Ya = %s g, ",float_);
		USART_SendString(buffer);
		dtostrf( Za, 3, 2, float_ );
		//Take values in buffer to send all parameters over USART
		sprintf(buffer,"Za = %s g\n\n",float_);
		USART_SendString(buffer);
		dtostrf(Xg, 3, 2, float_ );
		//Take values in buffer to send all parameters over USART
		sprintf(buffer,"Xg = %s degrees, ",float_);
		USART_SendString(buffer);
		dtostrf(Yg, 3, 2, float_ );
		//Take values in buffer to send all parameters over USART
		sprintf(buffer,"Yg = %s degrees, ",float_);
		USART_SendString(buffer);
		dtostrf( Zg, 3, 2, float_ );
		//Take values in buffer to send all parameters over USART
		sprintf(buffer,"Zg = %s degrees",float_);
		USART_SendString(buffer);
		USART_SendString("\n\n");
		_delay_ms(1000);     //1s delay for display purposes
	}
	return 0;
}
```

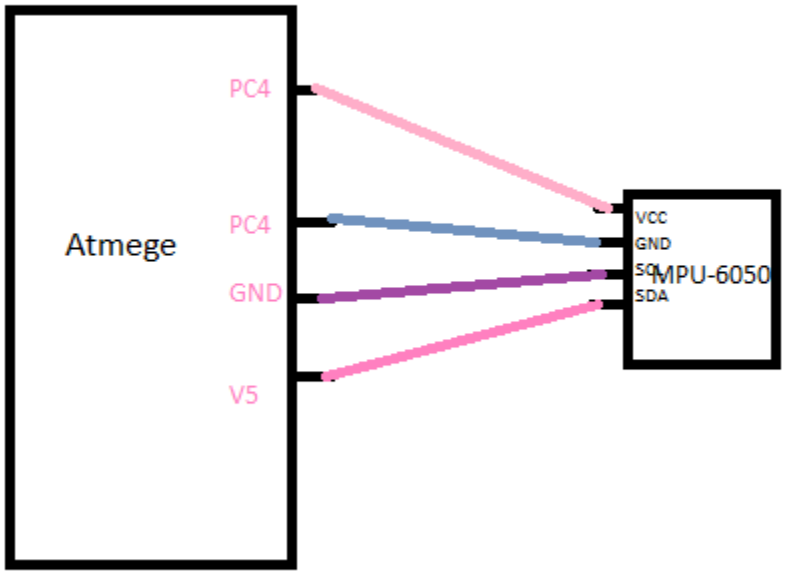
3. DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

```
/*
 * DA6_TASK2.c
 *
 * Created: 5/6/2020 12:49:21 PM
 * Author : Meral
 */
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <util/twi.h>
#define SCL_CLK 100000UL
#define BITRATE ((((F_CPU / SCL_CLK) / 1) - 16 ) / 2)
#define BAUD (((F_CPU / (BAUDRATE * 16UL))) - 1)   //prescale value
float Acc_x, Acc_y, Acc_z, Temp_out, Gyro_x, Gyro_y, Gyro_z;
void USART_Init(unsigned long BAUDRATE)     {
       UCSR0B |= (1 << RXEN0) | (1 << TXEN0);      //enable USART transmitter and receiver
       UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);//write USCRC and 1 stop bit
       UBRR0L = BAUD;                                          //load UBRRL
       UBRR0H = (BAUD >> 8);                              //load UBRRH
}
char USART_Rx(){
       while (!(UCSR0A & (1 << RXC0)));//Wait for new data
       return(UDR0);  //return received data
}
void USART_Tx(char data){
       UDR0 = data;                                  //Write data
       while (!(UCSR0A & (1<<UDRE0)));     //data transmit? buffer set to empty
}
void USART_SendString(char *str){
       int i=0;
       while (str[i]!=0){
              USART_Tx(str[i]);
              i++;
       }
}
void I2C_Init(void){
       TWBR0 = (uint8_t)BITRATE;
}
uint8_t I2C_Start(uint8_t slave_write_address){
       uint8_t status; /* Declare variable */
       TWCR0 = 0;/*reset TWI */
       TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);// transmit START condition
       while( !(TWCR0 & (1<<TWINT)) );/* Enable TWI, generate start condition and clear interrupt flag */
       if((TWSR0 & 0xF8) != TW_START)
              return 1;

       TWDR0  = slave_write_address; /* If yes then write SLA+W in TWI data register */
       TWCR0 = (1<<TWEN)|(1<<TWINT);/* Enable TWI and clear interrupt flag */

       while( !(TWCR0 & (1<<TWINT)) );     /* Wait until TWI finish its current job (Write operation) */
              status = TWSR0 & 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 & nackreceived or not? */
              return 1;/* If yes then return 1 to indicate nackreceived */
       else
              return 2;
}
```

```
uint8_t I2C_write(uint8_t data){
	uint8_t status;
	TWDR0  = data;
	TWCR0=(1<<TWEN)|(1<<TWINT);
	while( !(TWCR0 & (1<<TWINT)));
		status = TWSR0 & 0xF8;
	if(status==0x28)
			return 0;
	if(status == 0x28)
			return 0;
	if(status == 0x30)
		return 1;
	else
		return 2;
}
/* I2C read ack function */
uint8_t i2c_read_ack(void){
	TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);/* Enable TWI, generation of ack and clear interrupt flag */
	while( !(TWCR0& (1<<TWINT)) );/* Wait until TWI finish its current job (read operation) */
	return TWDR0;  /* Return received data */
}
/* I2C read nack function */
uint8_t i2c_read_nack(void){
	TWCR0 = (1<<TWINT) | (1<<TWEN);      /* Enable TWI and clear interrupt flag */
	while( !(TWCR0 & (1<<TWINT)) );      /* Wait until TWI finish its current job (read operation) */
	return TWDR0;  /* Return received data */
}
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); // UART TX (Transmit - senden) einschalten
	UCSR0C |= (1<<USBS0) | (3<<UCSZ00); //Modus Asynchron 8N1 (8 Datenbits, No Parity, 1 Stopbit)
}
void I2C_stop(void){
	TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);/* Enable TWI, generate stop condition and clear interrupt
flag */
}
void MPU6050_Init(void){
	_delay_ms(150);       /* Power up time >100ms */
	I2C_Start(0xD0); /* Start with device write address */
	I2C_write(0x19); /* Write to sample rate register */
	I2C_write(0x07); /* 1KHz sample rate */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x6B); /* Write to power management register */
	I2C_write(0x01); /* X axis gyroscope reference frequency */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x1A);/* Write to Configuration register */
	I2C_write(0x00);      /* Fs = 8KHz */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x1B); /* Write to Gyro configuration register */
	I2C_write(0x18);             /* Full scale range +/-2000 degree/C */
	I2C_stop();

	I2C_Start(0xD0);
	I2C_write(0x38);
	I2C_write(0x01);
	I2C_stop();
}
```

```
void MPU_Start_Loc(){
	I2C_Start(0xD0);
	I2C_write(0x3B);
	I2C_stop();
	I2C_Start(0xD1);
}
void Read_RawValue(void){
	MPU_Start_Loc();
	//take in X,y,z accelerometer value and Xg,Yg,Zg gyro value
	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_ack());
	I2C_stop();
}
int main(void){
	char buffer[30], float_[30];
	float Xa, Ya;
	I2C_Init();/* Initialize I2C */
	MPU6050_Init();/* Initialize MPU6050 */
	USART_Init(9600); /* Initialize USART with 9600 baud rate */

	while(1){
		Read_RawValue();
		Xa = (Acc_x/16384.0)*90;    //divide raw value by sensitivity scale factor to get actual
values
		Ya = (Acc_y/16384.0)*90;    //divide raw value by sensitivity scale factor to get actual
values


		dtostrf( Ya, 3, 2, float_ );
		sprintf(buffer,"filtered pitch = %s degrees   ",float_);
		USART_SendString(buffer);

		dtostrf( Xa, 3, 2, float_ );
		sprintf(buffer,"Roll Angle = %s degrees    ",float_);
		USART_SendString(buffer);

		_delay_ms(1000);
	}
	return 0;
}
```

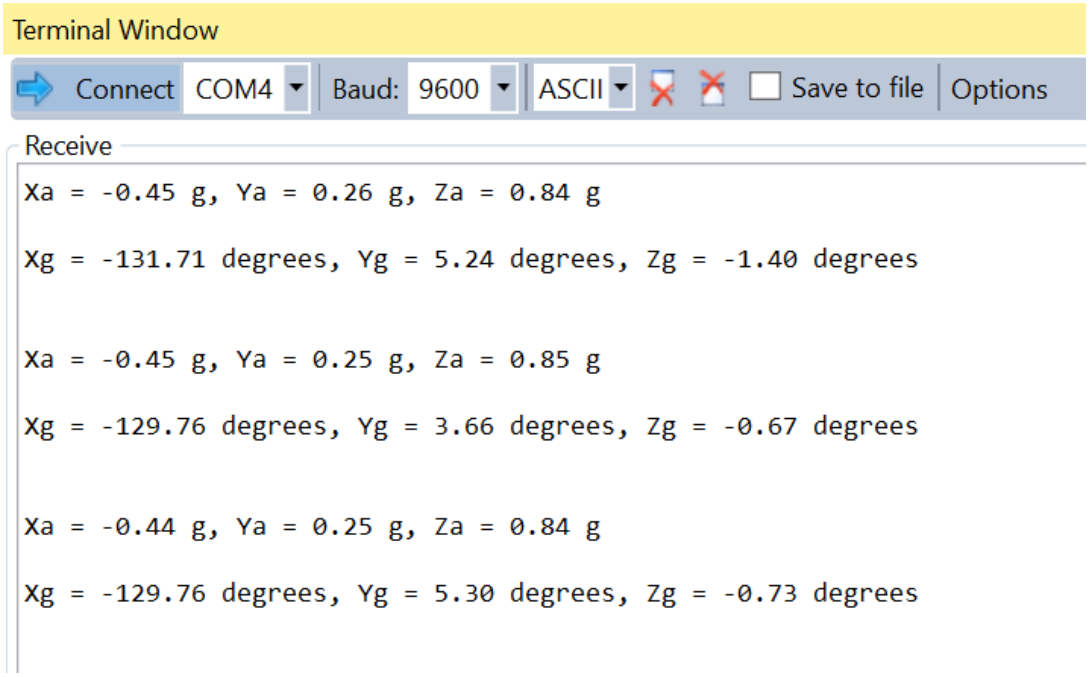
4. SCHEMATICS



5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1

Terminal window

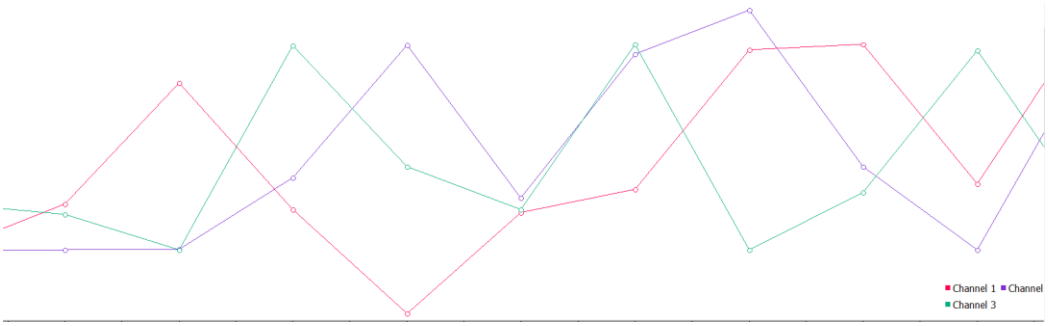


Simulation

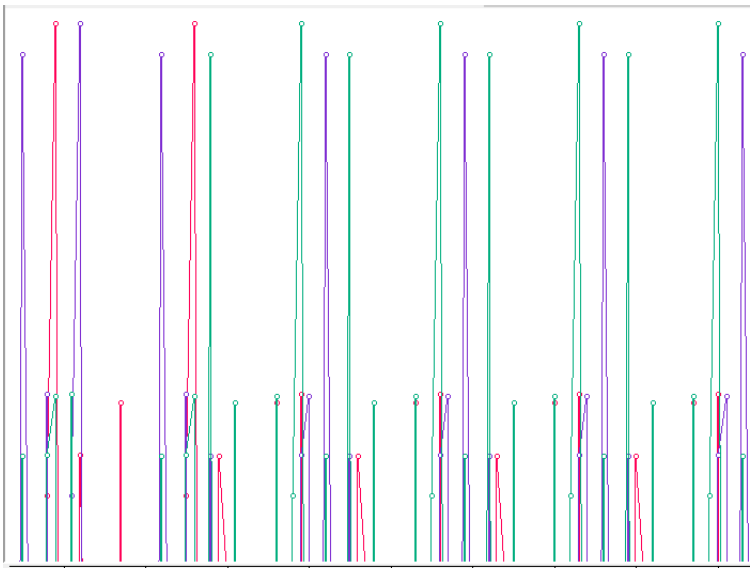
```
Done building target "CoreBuild" in project "DA6_TASK1.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "C:\Users\Meral\Documents\Atı
Done building target "Build" in project "DA6_TASK1.cproj".
Done building project "DA6_TASK1.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Serial plot

Zoom in



Zoom out



Task 2

Terminal Window

Connect COM4 Baud: 9600 ASCII Save to file Options

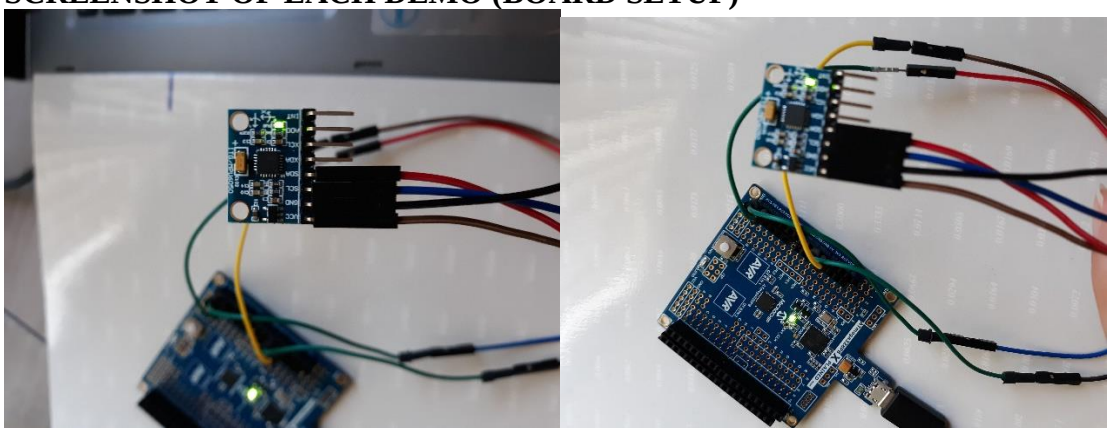
Receive

```
filtered pitch = -4.13 degrees   Roll Angle = -6.20 degrees
filtered pitch = -3.71 degrees   Roll Angle = -5.76 degrees
filtered pitch = -3.78 degrees   Roll Angle = -6.92 degrees
filtered pitch = -3.21 degrees   Roll Angle = -6.50 degrees
filtered pitch = -4.15 degrees   Roll Angle = -5.89 degrees
filtered pitch = -3.71 degrees   Roll Angle = -6.11 degrees
filtered pitch = -4.37 degrees   Roll Angle = -6.57 degrees
filtered pitch = -3.03 degrees   Roll Angle = -6.75 degrees
filtered pitch = -3.58 degrees   Roll Angle = -6.57 degrees
filtered pitch = -3.85 degrees   Roll Angle = -6.48 degrees
```

Simulation

```
Done building target "CoreBuild" in project "DA6_TASK2.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\meral\Docume
Done building target "Build" in project "DA6_TASK2.cproj".
Done building project "DA6_TASK2.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



7. VIDEO LINKS OF EACH DEMO

Task 1

https://www.youtube.com/watch?v=Q32YZvRXYg4

task 2

https://www.youtube.com/watch?v=HVnnypCjBhA

8. GITHUB LINK OF THIS DA

https://github.com/MeralAbuJaser/Submission_da/tree/master/DA6

"*This assignment submission is my own, original work*".

Meral Abu-Jaser