**IE6-BUL S23**

**Function and Characterization of an Inertial Sensor Cluster/ I2C bus**

Authors: Celestine Machuca, Soodeh Mousaviasl.

Contents

[**IE6-BUL S23** 1](#_Toc134004766)

[**Function and Characterization of an Inertial Sensor Cluster/ I2C bus** 1](#_Toc134004767)

[Materials Used 3](#_Toc134004768)

[Connection diagram 3](#_Toc134004769)

[Setup Used 3](#_Toc134004770)

[Lab Tasks 3](#_Toc134004771)

[Part 1 Setup 4](#_Toc134004772)

[What data is being recorded? 6](#_Toc134004773)

[how does the measurement data get to the laptop? 7](#_Toc134004774)

[Part 2 Configuration 7](#_Toc134004775)

[Find out how to set the sensor cluster to different bandwidths and measurement ranges by analyzing the data sheet? 7](#_Toc134004776)

[Try out different configurations for the measuring range of one channel of the accelerometer and measure the digital output values for a = -1 g; 0 ; +1 g. 9](#_Toc134004777)

[What is the resolution of each of these measurements? 10](#_Toc134004778)

[Part 3 Oscilloscope measurements on the I2C bus 10](#_Toc134004779)

[Connect the oscilloscope to your measurement setup. Measure the voltage between SCL and GND with a probe (please adjust the square-wave signal first!) and examine the data line SDA with a second probe. 10](#_Toc134004780)

[Compare your measurement result with the I2C data protocol from the lecture (or e.g. from https://de.wikipedia.org/wiki/I²C). 10](#_Toc134004781)

[Please answer the following questions in your lab report:What is the datarate? How many bits (raw) are transferred per second?Analyze a single I2C telegram based on your oscilloscope measurement.How does your measurement compare tothe physical layer of the ideal I2C? 10](#_Toc134004782)

[Part 4 Measuring Noise on a acceleration sensor 11](#_Toc134004783)

[Analyze the noise performance of one of the three axes of the accelerometer for different bandwidths (e.g. 260 Hz vs. 5 Hz) 11](#_Toc134004784)

[To do this, keep the sensor vibration-free/still and carry out a long-term measurement, e.g. over 1000 values. Check the measurement in the time domain for outliers, filter them out if necessary,using a suitable filter(either on the Arduino or on your computer).Document the filter and include source code and explanation into your report. 11](#_Toc134004785)

[Create and compare the histograms for at least two different bandwidths and, if applicable, with or w/o filter. What are the reasons for the differences? 11](#_Toc134004786)

[Part 5 Determination of the noise behavior of a channel of the angular rate sensor 11](#_Toc134004787)

[Analyze the noise behavior of one of the three axes of the angular rate sensor for different bandwidths (e.g. 260 Hz vs. 5 Hz). 11](#_Toc134004788)

[To do this, keep the sensor vibration-free/still and carry out a long-term measurement, e.g. over 1000 values. Check the measurement in the time domain for outliers, filter them out if necessary,using a suitable filter in the Arduinoor on your PC.Document the filter and include source code and explanation into your report. 11](#_Toc134004789)

[Create and compare the histograms for at least two different bandwidths and, if applicable, with or w/o filter. What are the reasons for the differences? 11](#_Toc134004790)

[How large is the offset of the yaw rate signal in the respective measurements? 11](#_Toc134004791)

[Part 6 Visualization with “Processing” 11](#_Toc134004792)

[Switch the setupto evaluating the data with Processing –for this you need the program „ArduinoProcessingMPU6050“ on the Arduino and „ProcessingMPU6050“withinProcessing 11](#_Toc134004793)

[Analyze the program for the Arduino. How does it work? How do you find out the correction values that need to be entered into the program? 11](#_Toc134004794)

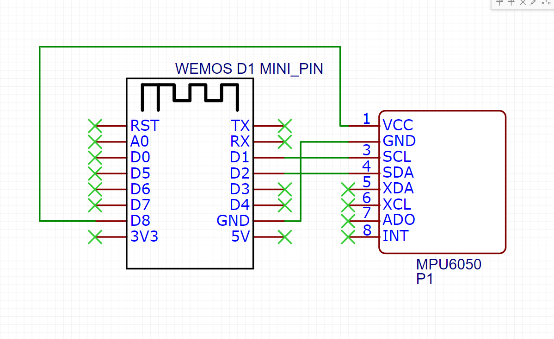
[Analyzed he program for Processing. What does this program do? How does it work? 13](#_Toc134004795)

[References 14](#_Toc134004796)

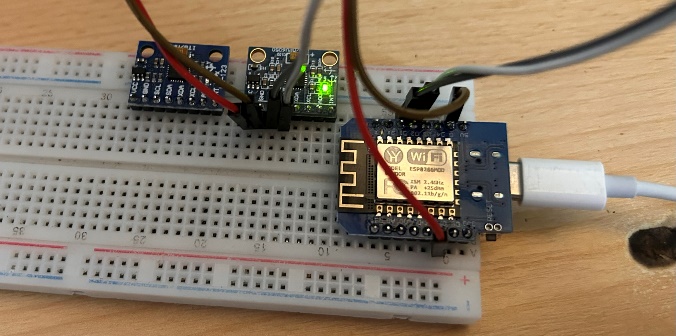
# Materials Used

* [D1 Mini](https://www.wemos.cc/en/latest/d1/d1_mini.html) microcontroller
* [Breadboard](https://www.arduino.cc/en/Tutorial/BuiltInExamples/Breadboard) for prototyping
* [MPU-6050](https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/) inertial sensor

# Connection diagram



# Setup Used



# Lab Tasks

## Part 1 Setup

c code used

#include <Arduino.h>

#include <Wire.h>

#define sensor\_address 0x68

// keywords = ['@filter\_setting', '@accelerometer\_setting', '@gyroscope\_setting']

#define FILTER\_CONFIG\_REG 0x1A

#define SET\_FILTER\_260HZ 0x06

#define SET\_FILTER\_184HZ 0x01

#define SET\_FILTER\_94HZ 0x02

#define SET\_FILTER\_44HZ 0x03

#define SET\_FILTER\_21HZ 0x04

#define SET\_FILTER\_10HZ 0x05

#define SET\_FILTER\_5HZ 0x06

#define GYRO\_CONFIG\_REG 0x1B

#define SET\_GYRO\_250 0x00

#define SET\_GYRO\_500 0x08

#define SET\_GYRO\_1000 0x10

#define SET\_GYRO\_2000 0x18

#define ACC\_CONFIG\_REG 0x1C

#define SET\_ACC\_2G 0x00

#define SET\_ACC\_4G 0x08

#define SET\_ACC\_8G 0x10

#define SET\_ACC\_16G 0x18

void SetConfiguration(byte reg, byte setting)

{

  // Aufruf des MPU6050 Sensor

  Wire.beginTransmission(sensor\_address);

  // Register Aufruf

  Wire.write(reg);

  // Einstellungsbyte für das Register senden

  Wire.write(setting);

  Wire.endTransmission();

}

void setup()

{

  Serial.begin(115200);

  pinMode(D7, OUTPUT);

  digitalWrite(D7, LOW);

  delay(1000);

  digitalWrite(D7, HIGH);

  delay(1000);

  Wire.begin();

  delay(1000);

  // Powermanagement aufrufen

  // Sensor schlafen und Reset, Clock wird zunächst von Gyro-Achse Z verwendet

  // Serial.println("Powermanagement aufrufen - Reset");

  SetConfiguration(0x6B, 0x80);

  // Kurz warten

  delay(500);

  // Powermanagement aufrufen

  // Sleep beenden und Clock von Gyroskopeachse X verwenden

  // Serial.println("Powermanagement aufrufen - Clock festlegen");

  SetConfiguration(0x6B, 0x03);

  delay(500);

  // filter configuration

  SetConfiguration(FILTER\_CONFIG\_REG, @filter\_setting);

  // gyro config

  SetConfiguration(GYRO\_CONFIG\_REG, @gyroscope\_setting);

  // acc config

  SetConfiguration(ACC\_CONFIG\_REG, @accelerometer\_setting);

  delay(500);

}

void loop()

{

  byte result[14];

  result[0] = 0x3B;

  Wire.beginTransmission(sensor\_address);

  Wire.write(result[0]);

  Wire.endTransmission();

  Wire.requestFrom(sensor\_address, 14);

  for (int i = 0; i < 14; i++)

  {

    result[i] = Wire.read();

  }

  // Accelerometer

  int16\_t acc\_X = (((int16\_t)result[0]) << 8) | result[1];

  int16\_t acc\_Y = (((int16\_t)result[2]) << 8) | result[3];

  int16\_t acc\_Z = (((int16\_t)result[4]) << 8) | result[5];

  // Temperature sensor

  int16\_t temp = (((int16\_t)result[6]) << 8) | result[7];

  int16\_t tempC = temp / 340 + 36.53;

  // Gyroscope

  int16\_t gyr\_X = (((int16\_t)result[8]) << 8) | result[9];

  int16\_t gyr\_Y = (((int16\_t)result[10]) << 8) | result[11];

  int16\_t gyr\_Z = (((int16\_t)result[12]) << 8) | result[13];

  // Print data

  // json like format

  Serial.print("{\"acc\_X\":");

  Serial.print(acc\_X);

  Serial.print(",\"acc\_Y\":");

  Serial.print(acc\_Y);

  Serial.print(",\"acc\_Z\":");

  Serial.print(acc\_Z);

  Serial.print(",\"temp\":");

  Serial.print(temp);

  Serial.print(",\"tempC\":");

  Serial.print(tempC);

  Serial.print(",\"gyr\_X\":");

  Serial.print(gyr\_X);

  Serial.print(",\"gyr\_Y\":");

  Serial.print(gyr\_Y);

  Serial.print(",\"gyr\_Z\":");

  Serial.print(gyr\_Z);

  Serial.println("}");

### What data is being recorded?

From the code the data being recorded is:

// Accelerometer

int16\_t acc\_X = (((int16\_t)result[0]) << 8) | result[1];

int16\_t acc\_Y = (((int16\_t)result[2]) << 8) | result[3];

int16\_t acc\_Z = (((int16\_t)result[4]) << 8) | result[5];

// Temperature sensor

int16\_t temp = (((int16\_t)result[6]) << 8) | result[7];

int16\_t tempC = temp / 340 + 36.53;

// Gyroscope

int16\_t gyr\_X = (((int16\_t)result[8]) << 8) | result[9];

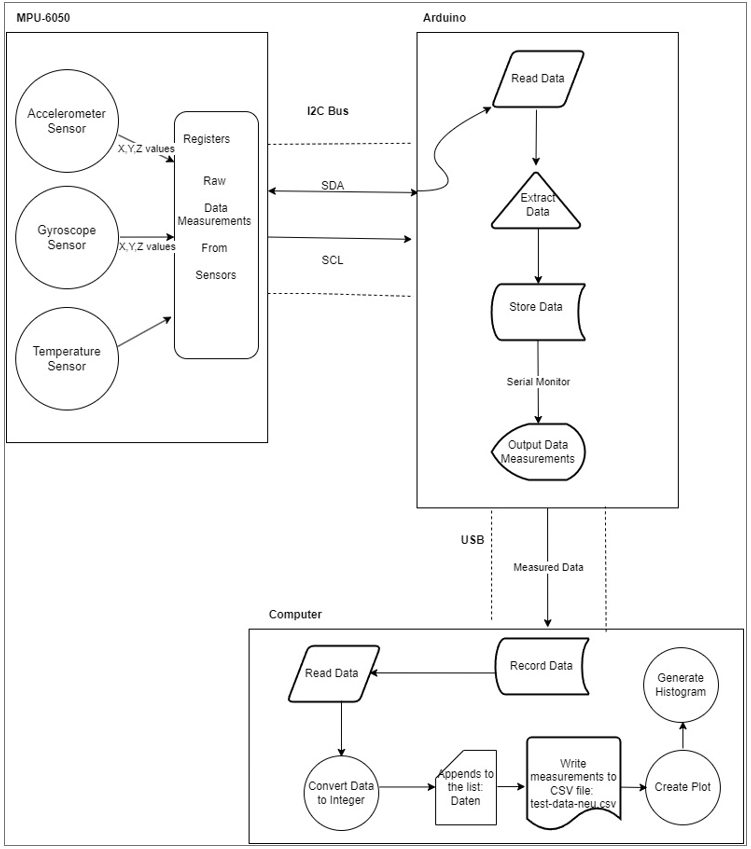
int16\_t gyr\_Y = (((int16\_t)result[10]) << 8) | result[11];

int16\_t gyr\_Z = (((int16\_t)result[12]) << 8) | result[13];

Example output:

{"acc\_X":4210,"acc\_Y":15,"acc\_Z":-1416,"temp":-4341,"tempC":24,"gyr\_X":-83,"gyr\_Y":-34,"gyr\_Z":36}

### how does the measurement data get to the laptop?



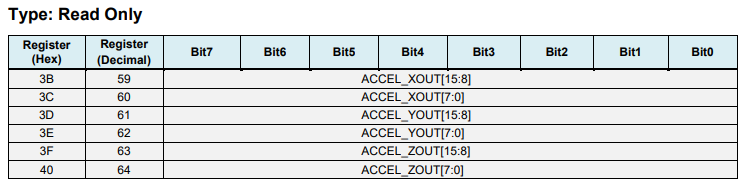
The MPU6050 sends data from its gyro, accelerometer, and temperature sensor to the Arduino via the I2C bus.  
The Arduino processes the received data and formats it for transmission.  
The Arduino sends the formatted data to the laptop via a USB cable using serial communication.  
A serial terminal or custom software on the laptop reads and interprets the received data, allowing you to view and analyze the measurements.

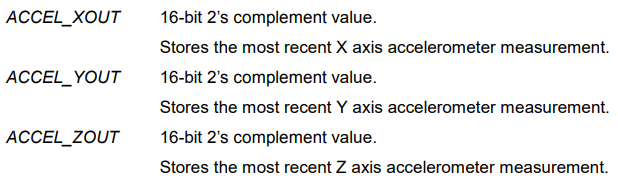
## Part 2 Configuration

### Find out how to set the sensor cluster to different bandwidths and measurement ranges by analyzing the data sheet?

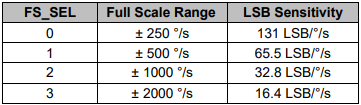
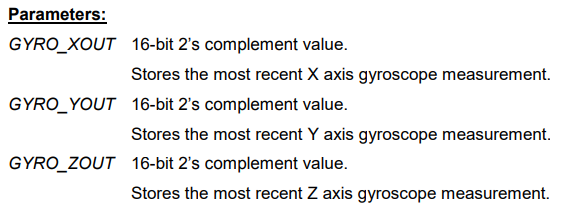
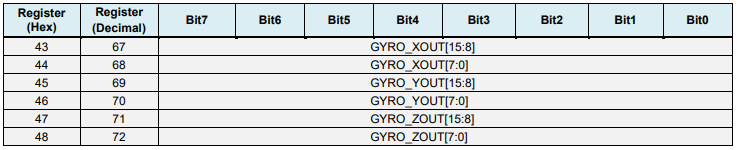
Using the datasheet, the sensor cluster can be set to different bandwidth and measurements through modifying the values of specific registers in the MPU-6050’s register map.  
The 0x1A register – CONFIG – is used to configure the DLPF (Digital Low-Pass Filter) and the sampling rate divider (SMPLRT\_DIV). The 0x1B register - GYRO\_CONFIG – is for configuring the Gyroscope’s full-scale range. And the 0x1C register - ACCEL\_CONFIG – is for configuring the full-scale range of the accelerometer.  
Hence, to set different bandwidth the DLPF\_CFG bits of 0x1A register were modified based on the availble options (look datasheet register map page 13).  
On Page 29 to 31 of the MPU6050 register map, there is a table that shows the different configurations for the accelerometer and gyroscope. The table shows the different bandwidths and measurement range and the corresponding register values.

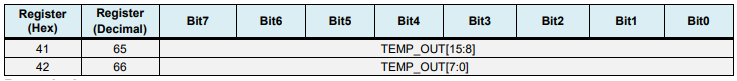
Page 29 Registers 59 to 64 – Accelerometer Measurements



Table

Description automatically generated



Graphical user interface, text

Description automatically generated

### Try out different configurations for the measuring range of one channel of the accelerometer and measure the digital output values for a = -1 g; 0 ; +1 g.

We performed the test over this values = +-2g, +-4g,+-8g, +-16h against earth gravity on the X Accelerometer. The results are shown below.

### What is the resolution of each of these measurements?

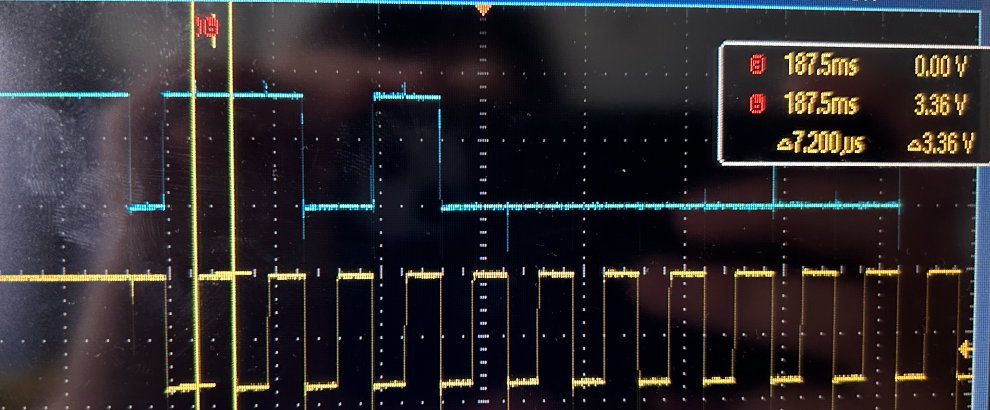
## Part 3 Oscilloscope measurements on the I2C bus

### Connect the oscilloscope to your measurement setup. Measure the voltage between SCL and GND with a probe (please adjust the square-wave signal first!) and examine the data line SDA with a second probe.

### Compare your measurement result with the I2C data protocol from the lecture (or e.g. from [https://de.wikipedia.org/wiki/I²C](https://de.wikipedia.org/wiki/I%C2%B2C)).

### Please answer the following questions in your lab report:What is the datarate? How many bits (raw) are transferred per second?Analyze a single I2C telegram based on your oscilloscope measurement.How does your measurement compare tothe physical layer of the ideal I2C?

From the data collected from the oscilloscope, we can see that given the period of scl(CH1) of delta 7.2 microseconds \* 2 = 14.4 microseconds, 1/14.4 \* 10 ^ 6 = 69.44 kHz. This is the frequency of the clock signal, if we assumed 1 bit per clock cycle, then the bitrate would be 69.44 kbps but that would be without taking into account the overhead from the protocol and other factors.



## Part 4 Measuring Noise on a acceleration sensor

### Analyze the noise performance of one of the three axes of the accelerometer for different bandwidths (e.g. 260 Hz vs. 5 Hz)

### To do this, keep the sensor vibration-free/still and carry out a long-term measurement, e.g. over 1000 values. Check the measurement in the time domain for outliers, filter them out if necessary,using a suitable filter(either on the Arduino or on your computer).Document the filter and include source code and explanation into your report.

### Create and compare the histograms for at least two different bandwidths and, if applicable, with or w/o filter. What are the reasons for the differences?

## Part 5 Determination of the noise behavior of a channel of the angular rate sensor

### Analyze the noise behavior of one of the three axes of the angular rate sensor for different bandwidths (e.g. 260 Hz vs. 5 Hz).

### To do this, keep the sensor vibration-free/still and carry out a long-term measurement, e.g. over 1000 values. Check the measurement in the time domain for outliers, filter them out if necessary,using a suitable filter in the Arduinoor on your PC.Document the filter and include source code and explanation into your report.

### Create and compare the histograms for at least two different bandwidths and, if applicable, with or w/o filter. What are the reasons for the differences?

### How large is the offset of the yaw rate signal in the respective measurements?

### Part 6 Visualization with “Processing”

### Switch the setupto evaluating the data with Processing –for this you need the program „ArduinoProcessingMPU6050“ on the Arduino and „ProcessingMPU6050“withinProcessing

### Analyze the program for the Arduino. How does it work? How do you find out the correction values that need to be entered into the program?

The function calculate imu error capturres 200 samples and calculate the drift on x and y by averaging the samples and obtaining the x and y components of the drift vector.

void calculate\_IMU\_error() {

  // We can call this funtion in the setup section to calculate the accelerometer and gyro data error.

  // From here we will get the error values used in the above equations printed on the Serial Monitor.

  // Note that we should place the IMU flat in order to get the proper values, so that we then can the correct values

  // Read accelerometer values 200 times

  while (c < 200) {

    Wire.beginTransmission(MPU);

    Wire.write(0x3B);

    Wire.endTransmission(false);

    Wire.requestFrom(MPU, 6, true);

    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0 ;

    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0 ;

    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0 ;

    // Sum all readings

    AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) \* 180 / PI));

    AccErrorY = AccErrorY + ((atan(-1 \* (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) \* 180 / PI));

    c++;

  }

  //Divide the sum by 200 to get the error value

  AccErrorX = AccErrorX / 200;

  AccErrorY = AccErrorY / 200;

  c = 0;

  // Read gyro values 200 times

  while (c < 200) {

    Wire.beginTransmission(MPU);

    Wire.write(0x43);

    Wire.endTransmission(false);

    Wire.requestFrom(MPU, 6, true);

    GyroX = Wire.read() << 8 | Wire.read();

    GyroY = Wire.read() << 8 | Wire.read();

    GyroZ = Wire.read() << 8 | Wire.read();

    // Sum all readings

    GyroErrorX = GyroErrorX + (GyroX / 131.0);

    GyroErrorY = GyroErrorY + (GyroY / 131.0);

    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);

    c++;

  }

  //Divide the sum by 200 to get the error value

  GyroErrorX = GyroErrorX / 200;

  GyroErrorY = GyroErrorY / 200;

  GyroErrorZ = GyroErrorZ / 200;

  GyroErrorZ = GyroErrorZ + 0.01;

  // Print the error values on the Serial Monitor

  Serial.print("AccErrorX: ");

  Serial.println(AccErrorX);

  Serial.print("AccErrorY: ");

  Serial.println(AccErrorY);

  Serial.print("GyroErrorX: ");

  Serial.println(GyroErrorX);

  Serial.print("GyroErrorY: ");

  Serial.println(GyroErrorY);

  Serial.print("GyroErrorZ: ");

  Serial.println(GyroErrorZ);

}

the value obtained is later use as a offset to correct the drift on the x and y axis.

  GyroX = GyroX - GyroErrorX ; // GyroErrorX ~(-0.56)

  GyroY = GyroY - GyroErrorY; // GyroErrorY ~(2)

  GyroZ = GyroZ - GyroErrorZ; // GyroErrorZ ~ (-0.8)

### Analyzed he program for Processing. What does this program do? How does it work?

The program for processing reads the data from the serial port, translate the 'cube' and display the yaw, pitch and roll values.

1. initiates the serial port

void setup() {

  size (1024, 768, P3D);

  myPort = new Serial(this, "COM4", 19200); // starts the serial communication

  myPort.bufferUntil('\n');

  // logo = loadImage("UrbanMobilityLab.png");

  // logo2 = loadImage("HAW\_Marke.png");

}

1. reads the data on serial event

void serialEvent (Serial myPort) {

  // reads the data from the Serial Port up to the character '.' and puts it into the String variable "data".

  data = myPort.readStringUntil('\n');

  // if you got any bytes other than the linefeed:

  if (data != null) {

    data = trim(data);

    // split the string at "/"

    String items[] = split(data, '/');

    if (items.length > 1) {

      //--- Roll,Pitch in degrees

      roll = float(items[0]);

      pitch = float(items[1]);

      yaw = float(items[2]);

    }

  }

}

1. draws the cube and the text, call on every frame

void draw() {

  translate(width/2, height/2, 0);

  background(233);

  textSize(22);

  text("Roll: " + int(roll) + "     Pitch: " + int(pitch), -100, 265);

  // Rotate the object

  rotateX(radians(-pitch));

  rotateZ(radians(roll));

  rotateY(radians(yaw));

  // 3D 0bject

  textSize(20);

  fill(0, 76, 153);

  box (500, 40, 200); // Draw box

  textSize(25);

  fill(255, 255, 255);

  text("HAW Hamburg", -183, 10, 101);

  //image(logo,0,-300,400,400);

  //image(logo2,-200,0);

  delay(10);

  //println("ypr:\t" + angleX + "\t" + angleY); // Print the values to check whether we are getting proper values

}

## References

MPU 6050 Register Map <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf>

I2C Data Frame <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>