ID S3

Function and Characterization of an Inertial Sensor Cluster / I2C bus

Required Knowledge

- I2C Interface
- Inertial Quantities: Acceleration, Yaw Rate; Coordinate Systems (Cartesian, Polar, Cylindrical)
- Arduino programming / Arduino IDE
- Python (Anaconda/Spyder)
- Processing (https://processing.org, Visualization)
- Datasheet Inertial Sensor Cluster MPU6050

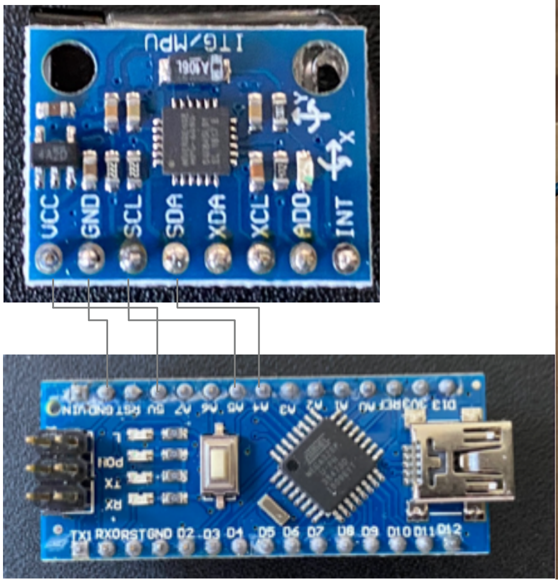
Material

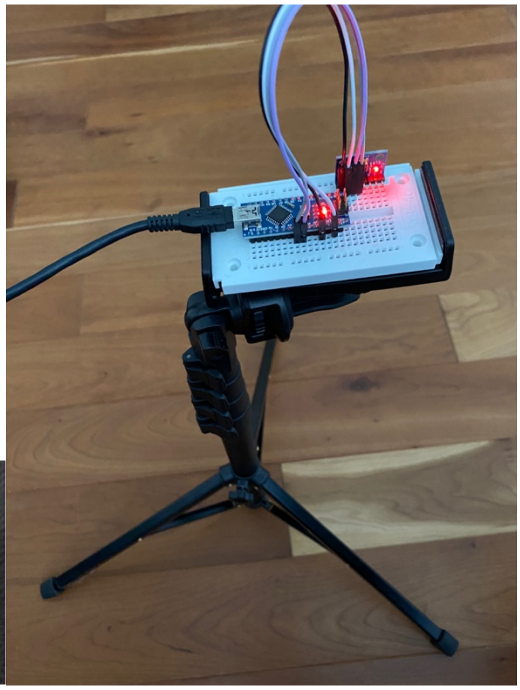
- Arduino Nano
- Inertial Sensor Cluster MPU6050
- Breadboard
- USB-Mini Kabel
- 4 Male-Male Connectors
- Laptop, Arduino-IDE, Python-IDE (Anaconda/Spyder), Terminal-Software (HTerm), Processing (https://processing.org)
- PLEASE INSTALL THE SOFTWARE ON YOUR COMPUTER IN ADVANCE

Time approx. 4h

Setup

Hardware:





Software:

- Arduino IDE
- Anaconda/Spyder
- Processing
- Prepared programs (Arduino, Python, Processing)

Part 1 – Setup

- Setup the Arduino and the MPU 6050 inertial sensor cluster on the breadboard
- Please make sure that you do not create a short circuit between the supply voltages 3V3 or 5V to GND!
- Place the setup in a stable position that is as vibration-free as possible, if available use a tripod
- Four connections are required: 5V, GND, SDA and SCL

- Download the "BasisMPU6050" program, compile it and transfer it to the Arduino
- Check the transmitted data with "Tools/Serial Monitor" or "Tools/Serial Plotter"; make sure that the transmission parameters (COM port and baud rate) are set correctly
- Download the "Basisprogramm_ReadUSB_MPU" program and load it into your editor under Anaconda/Spyder
- Start the program there and check the output values. You may have to install the "pyserial", "numpy" and "matplotlib" libraries in the Anaconda Navigator beforehand
- Make sure that the COM port and baud rate are also set correctly in the program
- Analyze both programs: Which measurement data is recorded and how does the measurement data get to the laptop

Deliverables:

- **Diagram of the data flow including a short written explanation.**

Part 2 – Configuration

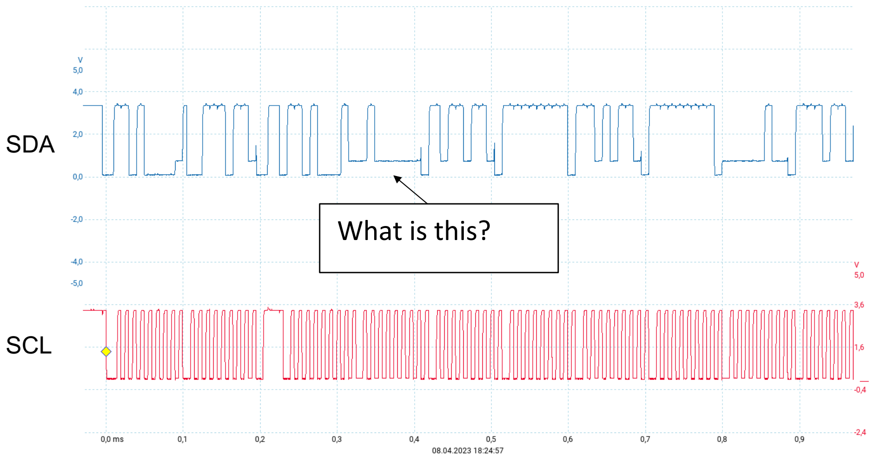
- Find out how to set the sensor cluster to different bandwidths and measurement ranges by analyzing the data sheet (hint: register map at https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf; specifically check the registers 0x1a, 0x1b, 0x1c); make sure to correctly interpret the manufacturers documentation (e.g. big/little endian, where is the LSB/MSB, …)
- 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 (turn sensor element accordingly).
- What is the resolution of each of these measurements?

Deliverables:

- Table of results for 3 different measurement ranges for 1 axis of the sensor including the output value for +1g, 0, -1g
- Explanation – how do the values measured relate to the measurement range selected? What is the respective resolution in g or m/s^2

Part 3 – Oszilloscope measurements on the I2C Bus

- Connect the oscilloscope to your measurement setup. Measure the voltage between SCL and GND with a probe (perform a probe calibration 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).

Deliverables:

- Photo of the setup
- Screen dump of your measurement
- Answers to the following questions:
  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 to the physical layer of the ideal I2C?

Part 4 – Measuring Noise on an 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).

Deliverables:

- Create and compare the histograms for at two different bandwidths and, if applicable, with or w/o filter. Document both mean value, standard deviation and include a plot of the histogram. What are the reasons for the differences?
- Calculate a sensor offset and explain your calculation.
- Document your filter and include source code and explanation into your report.

Part 5 – Measuring Noise on a yaw 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 Arduino or on your PC.

Deliverables

- Create and compare the histograms for at least two different bandwidths and, if applicable, with or w/o filter. Document both mean value, standard deviation and include a plot of the histogram. What are the reasons for the differences?
- How large is the offset of the yaw rate signal in the respective measurements?
- Document the filter and include source code and explanation into your report.

Part 6 (Special Feature!) – Virtualization with "Processing"

- Switch the setup to evaluating the data with Processing – for this you need the program „ArduinoProcessingMPU6050" on the Arduino and „ProcessingMPU6050" within Processing
- 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?
- Analyze the program for Processing. What does this program do? How does it work?
- Try it out: Move the sensor and watch the screen. How do you know that your sensor is not yet perfectly calibrated? How can you demonstrate the limitations of the sensor?

Deliverables:

- Create a data flow diagram for the setup.
- Document your results with a screen dump in your lab report.