Titel

Team 1: Eggs & Bacon

*Tobias Gilgenreiner, Felix Dollinger, Marco Stoiber,*

*Maximilian Kritzenthaler, Eduard Schröder*

Untertitel

# Sensorik

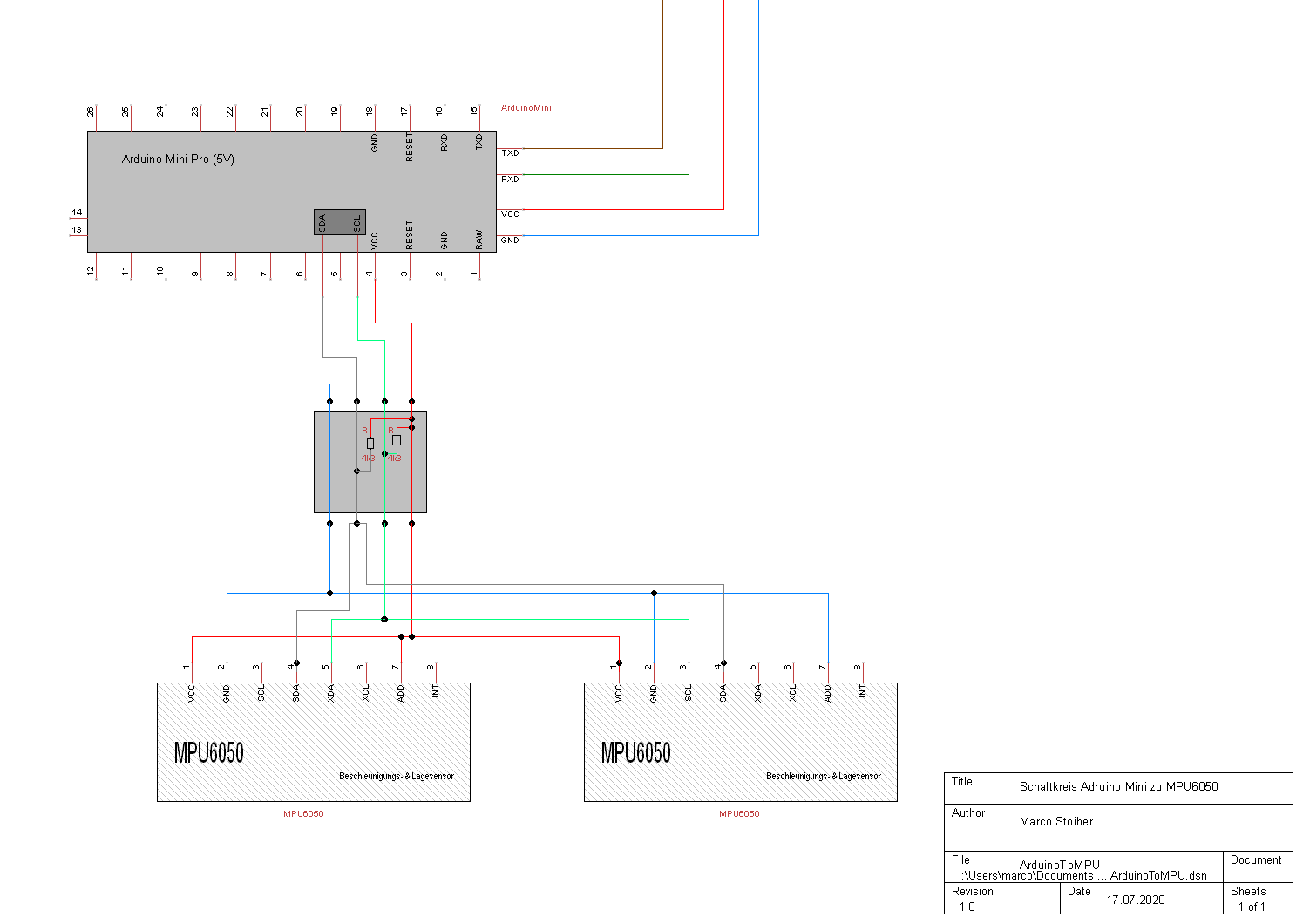
Im Folgenden wird der Aufbau der Sensorik, die Einrichtung der Entwicklungsumgebung und die Entwicklung der Firmware für die Arduino Mini Pros dokumentiert.

## Systemhardware

## Zur Bestimmung der menschlichen Bewegungen werden MPU6050 Einheiten verwendet. Hierbei handelt es sich um 3-Achsen Lage bzw. Beschleunigungssensoren. Dabei misst der Lagesensor die Winkelgeschwindigkeit bei Rotation um die x-/y-/z-Achse unter Verwendung des Corioliseffekts. Der Beschleunigungssensor macht sich die Gravitationskraft zu nutzen und bestimmt so die auf die Achsen wirkende Kraft bei Neigung des Sensors. Beide Sensoren besitzen unterschiedliche Sensitivitätsstufen. Für das Projekt wurden die Standard Einstellungen verwendet, also +/- 250°/s (Lagesensor) bzw. +/- 2g (Beschleunigungssensor). Dabei ist zu beachten, dass die Sensorwerte zunächst Rohdaten ausgeben, die zur tatsächlichen Verwendung erst noch bearbeitet werden müssen. Das geschieht im Backend.

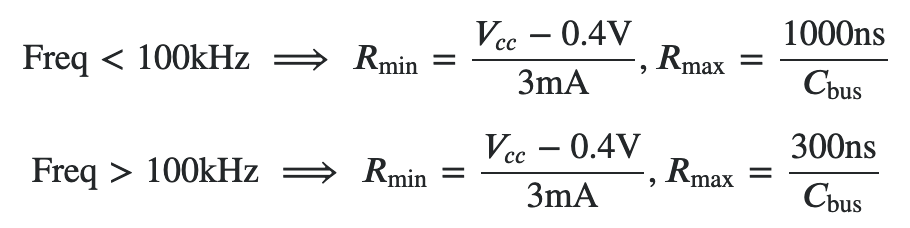
Zum Auslesen der Sensorwerte und Weitergabe der Daten an das Backend werden Arduino Mini Pros verwendet. Diese kommunizieren über einen I2C Bus mit jeweils zwei zugeordneten Sensoren und fragen abwechselnd die Sensordaten ab und senden die Werte über eine UART Verbindung an das Backend.

Hier ist der Schaltkreis, der die Verbindung der MPUs mit dem Arduino Mini beschreibt:

­­­­­

Für den I2C Bus sind externe Pullup Resistoren notwendig, die sowohl die Dataline, als auch die Clockline bei Nichtnutzung des Busses auf High ziehen.

Zur Berechnung des korrekten Widerstands können folgende Formeln verwendet werden:



## Wir haben für den I2C die Standard Übertragungsrate von 100kHz verwendet und liegen damit bei einem Wert zwischen 1,5 und 10kΩ. Letztlich sind 4,3kΩ Pullup Widerstände verbaut worden.

Über den ADD Pin der MPUs kann die Hardwareadresse manipuliert werden. Setzt man den Pin auf High, wird das letzte Bit der Adresse zu 1 bzw. beim Setzen auf Low zu 0. Das ist notwendig um die beiden Sensoren bei der I2C Kommunikation voneinander unterscheiden zu können.

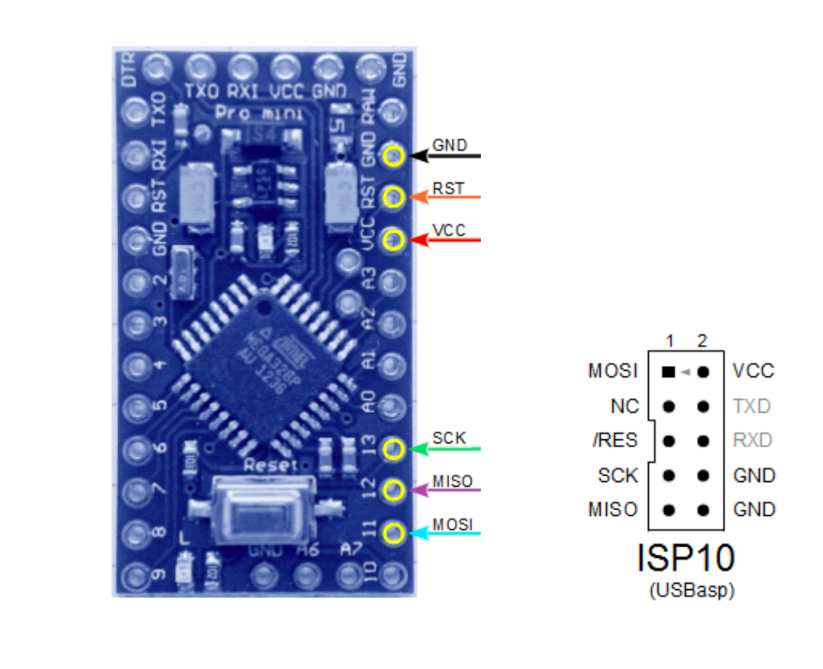
Unserer Erfahrung nach, halten die LEDs der MPUs nicht lange. Daher der Hinweis: Nur weil kein Lämpchen auf dem Sensor leuchtet, heißt es nicht, dass er nicht mehr funktioniert.

**Einrichtung der Entwicklungsumgebung**

Die Firmware ist mit Atmel Studio 7 auf einem Win10 Host entwickelt worden. Zum Flashen der Arduino Minis wurde ein USBasp verwendet. Zur Einrichtung wurde folgende Anleitung verwendet[[1]](#footnote-1). Bei der Anleitung ist zu berücksichtigen, dass es sich bei den Arduino Mini Pros um ein ATmega328P Board handelt.

Weiterhin ein Hinweis zur Installation des USBasp Treibers. Es muss aus Kompatibilitätsgründen der libusb-win32(v1.2.6.0) Treiber installiert werden.



Zum Flashen sind male to female Stecker notwendig, die wie folgt verbunden werden müssen:

**Entwicklung der Firmware**

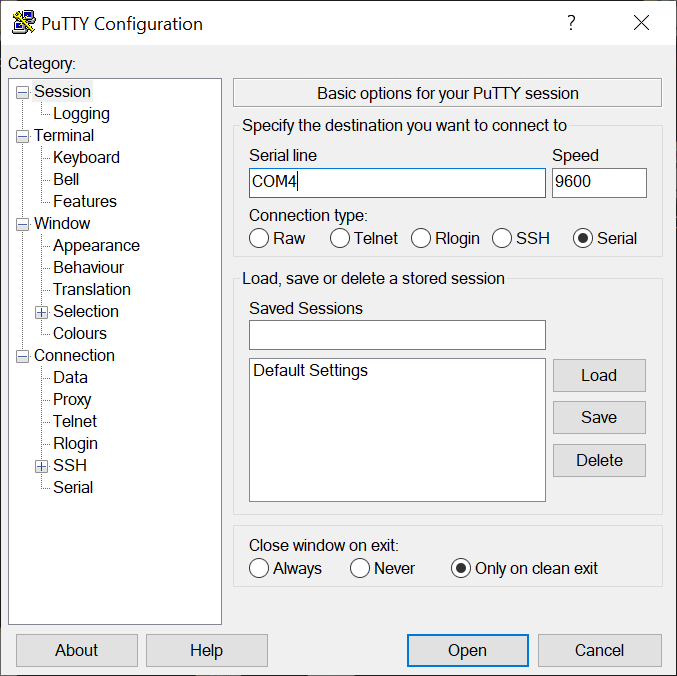
Die Firmware beinhaltet zwei Standard Bibliotheken für die I2C bzw. die UART Kommunikation. Weiterhin wurde eine Bibliothek für die MPUs entwickelt. Diese ermöglicht die Initialisierung der Sensoren und das Abfragen von Lage- und Beschleunigungswerten. Die Initialisierung besteht im Grunde darin, die Sensoren aus dem Ruhezustand aufzuwecken. Zusätzlich zu den Sensordaten wird der jeweilige Zeitpunkt der Abfrage übergeben. Dafür wird der Timer1, des ATmega328P Boards verwendet, der die verstrichene Zeit in Millisekunden genau angibt. Um die Datenpakete voneinander zu unterscheiden wird eine ID mitversendet. Diese muss manuell in der Firmware angepasst werden, da sich die Sensoren zwar innerhalb eines Messpaars unterscheiden, aber auf das ganze System betrachtet mehrfach die gleiche Hardwareadresse auftritt.

Ein exemplarisches Datenpaket sieht so aus:

id 0 t 150 ax -8932 ay -204 az 12912 gx 3035 gy- 4083 gz -179

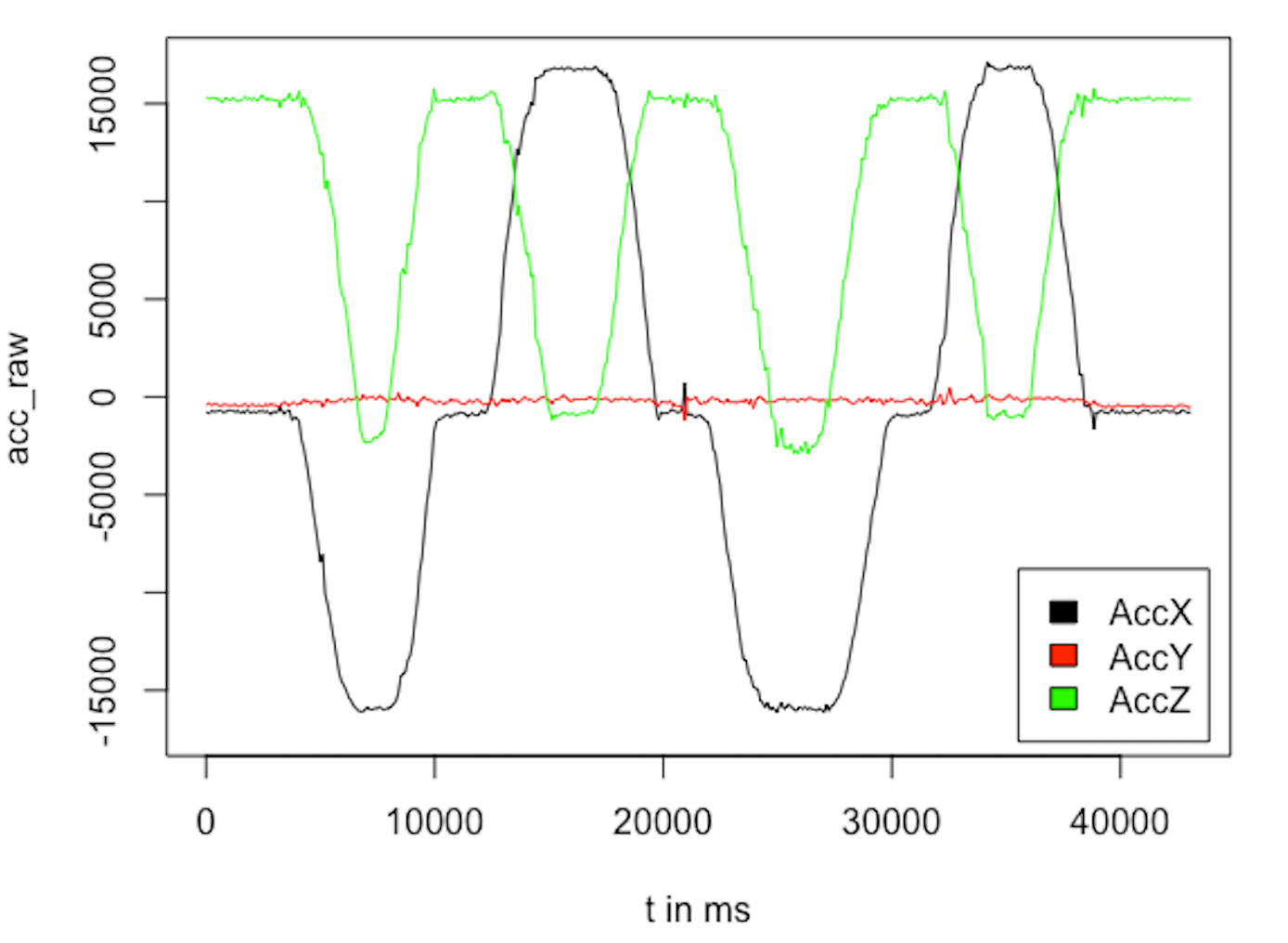
Die Abkürzungen a bzw. g stehen für accelrometer (= Beschleunigungssensor) und gyrometer (= Lagesensor).

Essentiell für die Entwicklung des Codes war das Debuggen mit Putty. Dafür ist aber eine funktionierende UART Kommunikation notwendig. Dafür wurde eine vorgefertigte Bibliothek verwendet.

****

In Putty muss die Serielle Schnittstelle angepasst werden. Dafür muss im Geräte-Manager überprüft werden welchen Port der UART zu USB Wandler vom Betriebssystem erhalten hat. Zusätzlich muss die verwendete Baudrate angegeben werden. In diesem Fall liegt diese bei 9600. Zur verbesserten Übersichtlichkeit sollten Verzögerungen in Form von waits im Code eingebaut werden.

Ein weiterer Hilfreicher Punkt ist es, vorab die Sensordaten mit Hilfe von Plots zu visualisieren, um ein Gefühl für die Werte zu bekommen.



Hier sieht man das Verhalten des Beschleunigungssensors bei abwechselnder rechts bzw. links Rotation. Man erkennt das grundsätzliche Verhalten der Achsen und stellt weiterhin fest, dass die Werte mit einem Rauschen behaftet sind.

Stundentabelle Maximilian Kritzenthaler:

|  |  |
| --- | --- |
| **Aktivität** | **Stunden** |
| Wöchentliche Meetings | 35 |
| Sprints | 50 |
| Entwicklungsumgebung einrichten + Flashen | 5 |
| Einlesen ATmega328P, MPU6050, I2C, UART | 10 |
| Schaltplan auf Breadboard umsetzen | 1 |
| Sensordaten Visualisierung | 2 |
| Firmware implementieren + Dataframe entwickeln | 25 |
| Debugging | 7 |
| Testen | 7 |
| Gesamt | 142 |

1. <https://medium.com/manash-en-blog/setting-up-atmel-studio-for-usbasp-and-avr-programming-802bb4dcf1e9> [↑](#footnote-ref-1)