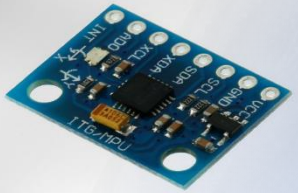
Sensorik (IMU)

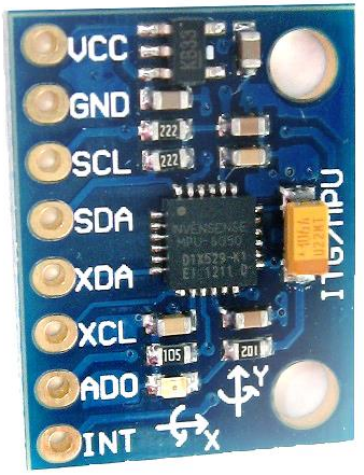
Inertial Measurement Unit
IMU3000 Combo

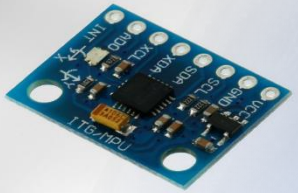


Inhalt

Umfang: ca. 1-3 Zeitstunden

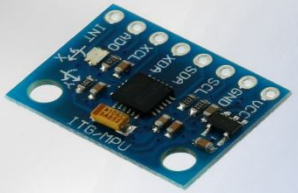
- Einführung Quadrokopter-Labor
- IMU (Intertial Measurement Unit)
- TWI (Two Wire Interface)
- EMQ Framework
- Aufgaben





Einführung Quadrokopter-Labor

- Programmierung der Flugsteuerung (Software) eines Quadrokopters
- Schrittweise Entwicklung der Software Komponenten
- Die Übungen bauen aufeinander auf
- Was gehört zur Flugsteuerung?
  - Sensorik (IMU)
  - Signalverarbeitung (Filter, Umwandlung in Quaternion, etc.)
  - Regelung (Attitude Control, Yaw Control, 3 DOF Control)
  - Steuerung (Automation, Kommandierung)
  - Kommunikation (Telemetrie, Telecommandos)



Einführung Quadrokopter-Labor

Sensorik (IMU)

Signalverarbeitung

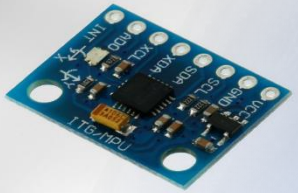
Regelung

Aktuatorik

Telemetrie

Telekommandos

Bodenstation
(PC, Tablet)

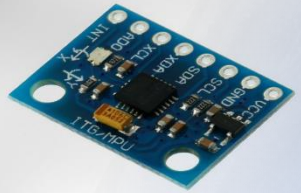


Einführung Quadrokopter-Labor

Überblick Quadrokopter-Labor

• 1. Woche: Einführung USART	Grundlagen, Serielle Schnittstelle
• 2. Woche: Einführung DIP	Grundlagen, Ansteuern des Displays
• **3. Woche: Sensorik (IMU)**	Grundlagen Framework, Ansteuern der Sensoren, Kalibrieren
• 4. Woche: Signalverarbeitung	Orientierungsdarstellung als Quaternion
• 5. Woche: Telemetrie	Empfang und Anzeige von Daten an der Bodenstation
• 6. Woche: Telecommandos	Senden von Kommandos an den Quadrokopter
• 7. Woche: Aktuatorik und Lageregelung	Motoren ansteuern und Regelung einer horizontalen Achse (1 DOF)
• 8. Woche: Yawregelung	Regelung der Gierachse und Superpositon zweier Achsen (2 DOF)
• 9. Woche: 3DOF-Regelung	Regelung aller 3 Freiheitsgerade
• 10. Woche: Automation	Automatisierung: Automatisch gesteuertes Flugmanöver ausführen

DOF = Degree of Freedom (Freiheitsgrad)



IMU (Intertial Measurement Unit)

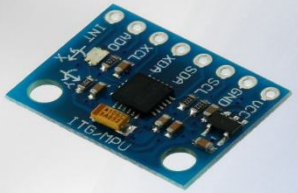
Definition:
Eine IMU ist die Kombination mehrerer Inertialsensoren (Trägheitssensoren). Üblich ist ein triaxialer Aufbau aus Beschleunigungs- (Accelerometer) und Drehratensensoren (Gyroskope). Die IMU ist die Sensoreinheit eines Inertialen Navigationssystemes, welches Positions- und Orientierungsänderung erfasst. Dies kann als Input der Orientierungs-darstellung zur Verwendung in einer Lageregelung hergenommen werden.

Positionsbestimmung (1D, vereinfacht):

$$P = \int \int a$$

Orientierungsbestimmung (1D, vereinfacht):

$$\varphi = \int \omega$$

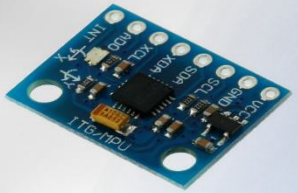


IMU (Intertial Measurement Unit)

IMU 3000:

- IMU3000 (InvenSense) + ADXL345 (Analog Devices)
- Gyroskope (IMU3000) + Accelerometer (ADXL345)
- Ansteuerung per TWI
- Datenblätter:
  - ADXL345.pdf
  - IMU3000.pdf
- Bevor der Sensor verwendet werden kann,

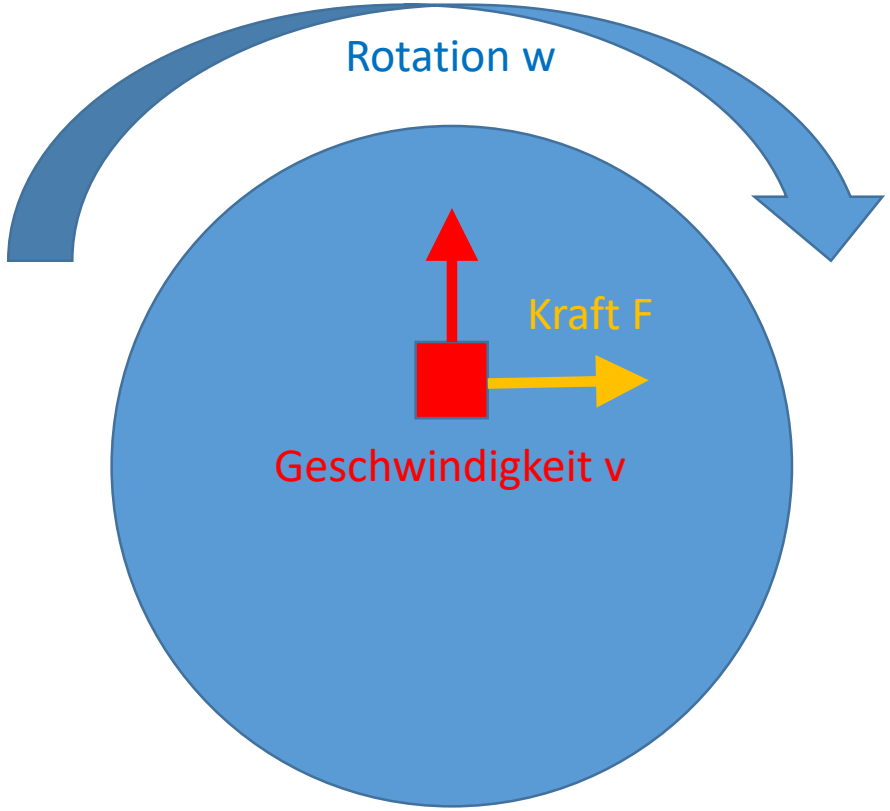
  muss er konfiguriert und kalibriert werden

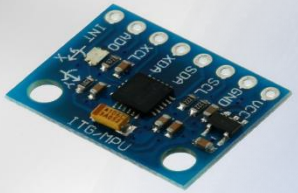


IMU (Intertial Measurement Unit)

Funktionsprinzip Gyroskop (Gyro):

- Messprinzip Corioliskraft (analog zur Lorenzkraft):
  - Bewegte Teilchen (v) (Ursache)
  - gleichzeitige Rotation (w) (Vermittlung)
    -> Wirkung: Kraft (F) senkrecht zu Ursache und Vermittlung
  - Kraft steht Senkrecht auf Drehachse und Bewegung
  - Tritt auf in rotierenden Bezugssystemen (w) bei gleichzeitiger Bewegung (v)
  - F = - 2 · m · ω · v
- MEMS: Mikro-Elektro-Mechanisch
- Triaxial (x,y,z = 3 DOF)
- Drehratensensor: Misst Drehgeschwindigkeiten
- Messung: Kapazitiv

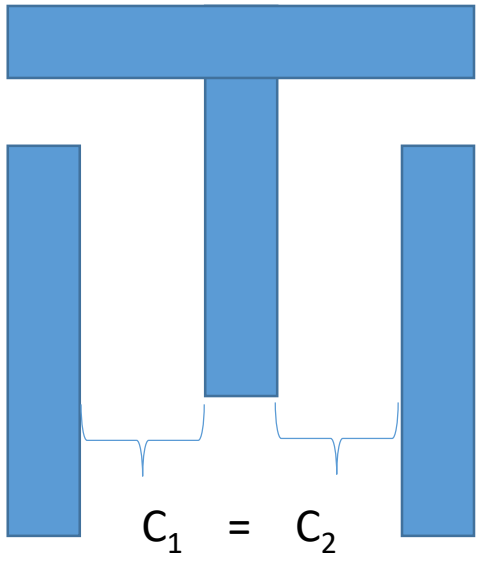


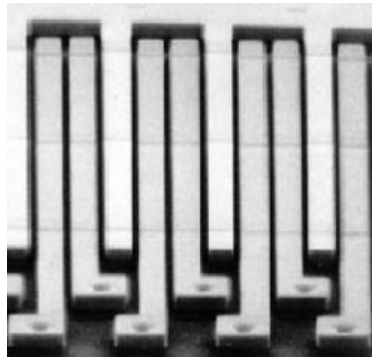


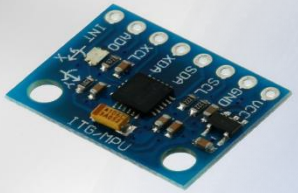
IMU (Intertial Measurement Unit)

<u>Funktionsprinzip Accelerometer (Acc)</u>:

- Messprinzip Feder-Masse Prinzip
  - Zwei Kammreihen
  - Aufgrund von Trägheit ändern sich die Abstände
  - Messung der Abstandsänderung kapazitiv
- MEMS: Mikro-Elektro-Mechanisch
- Triaxial (x,y,z = 3 DOF)
- Beschleunigungssensor: Misst Beschleunigungen (translatorisch)



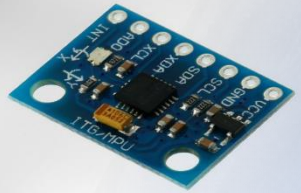




TWI (Two Wire Interface)

TWI

- Serieller Master-Slave-Bus
- Auch I²C oder I2C (inter-Integrated Circuit) genannt
- 2 Busleitungen: SCL (serial clock), SDA (serial data line)
- Vorteile: günstig, wenige Verdrahtungen
- Nachteile: störanfällig, geringe Leitungslänge
- Taktmodi: Standard bis 100 kHz, Fast Mode bis 400 kHz
- Übertragungsrate: max. 3,4 Mbit/s (relativ langsam)



TWI (Two Wire Interface)

TWI Datenaustausch

- Startsignal
- 1 Byte Adresse (7bit) + R/W (1bit = 1 für Write)
- ACK (vom Slave)
- Byteweise Datenpakete je quittiert mit ACK
- Stopsignal

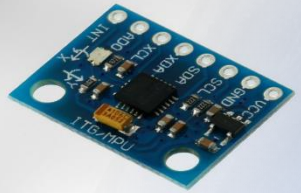
- Die genannte Adresse ist die I^2C/TWI Slave Adresse des angesprochen Teilnehmers.

Implementierung:

- Verwende TWI Framework

Datenpakete:

- Beim AVR Framework mind. 2 Bytes
- 1. Byte **Kommando** bzw. Adresse, für R/W
- 2. Bytes + Weitere sind **Daten** (R/W)
- Bei Burstmode werden mehrere Bytes von einer Anfangsadresse angefangen übertragen.

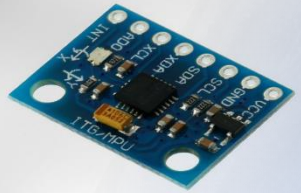


TWI (Two Wire Interface)

TWI Framework

- Verwendet Struktur twi_package_t
- chip = TWI Slave Adresse
- addr = **Kommando**
- addr_length = Länge des Kommandos in Bytes
  Üblicherweise = 1
- buffer = Pointer auf Speicherbereich der **Daten**
  - Send / Write (W): Diese Daten werden versendet
  - Receive / Read (R): Hierhin werden die empfangenen Daten geschrieben
- length = Anzahl Bytes die gesendet / empfangen werden

```
typedef struct
{
  char chip;// TWI Slave Address

  unsigned int addr;// Command Byte to be sent

  int addr_length;// Amount of Command Bytes

  void *buffer;// Buffer for Data (Send/Receive)

  unsigned int length;// Amount of Data Bytes
} twi_package_t;
```



TWI (Two Wire Interface)

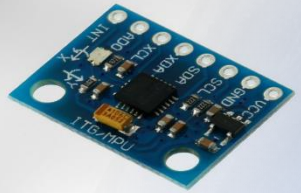
TWI Framework Beispiel

```
char data_received_mpu_acc[6];
twi_package_t packet_mpu6000_acc;

packet_mpu6000_acc.chip = 0x69;
packet_mpu6000_acc.addr_length = 1;
packet_mpu6000_acc.length = 6;
packet_mpu6000_acc.addr = 0x3B;
packet_mpu6000_acc.buffer = data_received_mpu_acc;

read_from_twi(&packet_mpu6000_acc, 0);
```

Danach stehen die 6 Daten-Bytes aus dem Sensorspeicher beginnend mit Adresse 0x3B in data_received_mpu_acc

Bei einem Fehler wird eine Nachricht ausgebeben. Anhand des zweiten Parameters, hier 0, kann diese Fehlernachricht zugeordnet werden.

```
Lesen / Daten empfangen:
read_from_twi(const twi_package_t *twi, int error_id);

Schreiben /  Daten senden:
write_to_twi(const twi_package_t *twi);
```

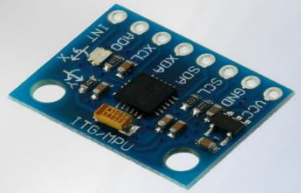


TWI (Two Wire Interface)

IMU konfigurieren

- Wie die Sensoren konfiguriert werden kann, steht im Datenblatt
- Es werden 6 Bytes (pro Sensor 3) zur Konfiguration geschrieben (empfohlen siehe Tabelle)
- Schreibe also 2x 3x 1Byte entsprechend der Tabelle
- Beachte: buffer ist ein Pointer auf die unten stehenden Daten
- Eine alternative TWI-Adresse für die IMU3000 lautet: 0x69 (Alternative zu 0x68 (Standard) abhängig vom verbauten Sensorboard)

Sensor TWI-Adresse (chip)	Konfigurations-Kommando (addr)	Konfigurations-Daten (buffer)	Funktion
0x68	0x16	0x18	IMU 3000: DLPF, Scale
0x68	0x3E	0x01	IMU 3000: PM, x Gyro Ref.
0x68	0x15	0x07	IMU 3000: Sample Rate
0x53	0x31	0x09	ADXL: Data Format
0x53	0x2D	0x08	ADXL: Power Ctrl
0x53	0x2C	0x0D	ADXL: 800 Hz Frequenz



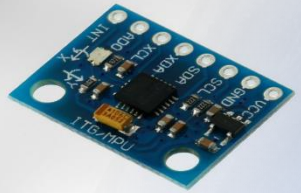
EMQ Framework

EMQ Framework

- Mitgeliefert als Library mit Open-Source-Teil
- Extra für das Quadrokoter-Labor entwickelt
- Enthält Treiber mit Funktionen für
  - Hardware: CPU, IRQ, TC, Remote, ADC, USART, TWI, Delay, Motoren, LED
  - Software:
    - Grundlagen zum Einfachen starten
    - Funktionsvorlagen zum Ausfüllen
- Details siehe Softwaredokumentation: EMQ_Framework.pdf
- Das EMQ Framework verwendet kein Betriebssystem
- Der gesamte Code läuft in einer Endlosschleife (While)

EMQ Datentypen

- In der Datei EMQ_Interface_Data.h stehen einige, verfügbare EMQ Datentypen
- Die Datentypen mit dummy sollen später mit sinnvollen Inhalt gefüllt werden
- Alle anderen Datentypen **NICHT** modifizieren.
- Die Systemzeit steht in der Variable tc_ticks [ms].



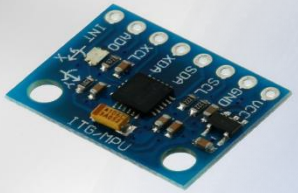
Aufgaben

Notwendige Hardware:

- EMQ3000
- Mini USB Kabel für Strom und zum Flashen
- USART Kabel für Kommunikation: RS232 auf PC (RS232 oder USB)
- Sensor IMU300 Combo + Extra-Kabel

Notwendige Software:

- AVR Studio 32 installiert (mit Tool Chain und FLIP Treiber)
- EMQ Framework (Code)
- Dokumente:
  - EMQ_Framework.pdf
  - ADXL345.pdf
  - IMU3000.pdf



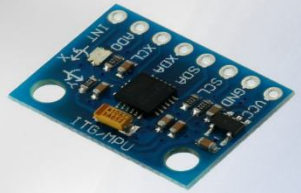
Aufgaben

Aufgabe 1a:
Mache dich mit dem EMQ Framework vertraut. Lies die Dokumentation und gucke dir alle Quellcode-Dateien an, bevor du anfängst.

Aufgabe 1b:
Konfiguriere die beiden Sensoren. Fülle dazu die Funktion my_imu_init() aus. Verwende die Werte aus der Tabelle in Folie 14 sowie das Framework aus Folie 12 und 13.

Aufgabe 1c:
Lies und konditioniere die Sensoren. Dazu muss du im Datenblatt nachschlagen, an welcher Registeradresse die Werte stehen. Konditionieren meint, dass die Rohdaten der Sensoren in korrekt skalierte Werte umgewandelt werden. Jeder Sensor liefert 6 Bytes, d.h. jeweils ein High- und ein Low-Byte pro Achse.

Die Skalierungsfaktoren findest du in den Datenblättern.



Aufgaben

Aufgabe 2:
Kalibriere die Sensoren, indem über eine im Code leicht einstellbare Anzahl an Messungen ein Mittelwert errechnet wird und dieser zum Bias-Ausgleich herangezogen wird. Der Gyro ist auf 0/0/0 zu kalibrieren, der Acc ist auf den Gravitationsvektor zu kalibrieren.

Aufgabe 3a:
Führe die eindimensionale Integration für die Messungen des Gyro durch, indem du die drei Achsen unabhängig voneinander integrierst. Bei einer 90° Drehung sollten auch 90° herauskommen.

Hinweis:
Der Sensor sollte hierzu auf dem Tisch liegen und mittels Extra-Kabel verbunden werden.

Aufgabe 3b:
Im Display sollen die drei Winkel angezeigt werden. Modifiziere dazu die renew_display() Funktion.