Prinzipien und Komponenten Eingebetteter System

Wintersemester 2013/2014

Christoph Steup steup@ivs.cs.uni-magdeburg.de
André Dietrich dietrich@ivs.cs.uni-magdeburg.de
Sebastian Zug zug@ivs.cs.uni-magdeburg.de

3. Praktische Aufgabe

Version 0.1

Anmerkungen

Machen Sie sich mit der Dateistruktur der Vorlage vertraut, diese umfasst insbesondere:

- `diverse Arduino Bibliotheken (I2Cdev, HMC5883L, usw.)`
- `Flydurino.cpp, Flydurino.h`
- `vorlage.ino`
- `arduino.mk, Makefile`
- `showOffset.py, determineDelay.py`

Sie können die Vorlage entweder in der Eclipse oder Arduino IDE bzw. als einfaches Makefile Projekt benutzen. Ihren Programmcode fügen Sie bitte in die Datei `vorlage.ino` ein.

Beide Teilaufgaben sollen in einen Programmcode umgesetzt werden. Für den Wechsel zwischen den Teilaufgaben sind die 2 Buttons zu benutzen. Erweitern Sie dazu die Funktion

```
int8_t checkButtons();
```

mit eigenem Programmcode. Übernehmen Sie zunächst Ihre Lösungen aus der Aufgabe 2. Achtung, einige Datentypen wurden angepasst, möglicherweise sind Korrekturen erforderlich.

Teilaufgabe 1

Die Vorlage stellt Ihnen bereits eine Auslesefunktion für den Gyro des Flydurino zur Verfügung. Benutzen Sie diese Information um Winkeländerungen zu detektieren. Integrieren Sie die Drehrate um die Z-Achse über der Zeit und berechnen Sie die aktuelle Richtung des Roboters.

Beginnen Sie damit, dass Sie zunächst den Offset des Gyroskops für einen unbewegten Sensor bestimmen. Zeichnen Sie dafür die Daten auf und berechnen Sie den mittleren Offset. Beachten Sie, dass im Sensor ein Low-Pass-Filter aktiviert werden sollte, um das Ergebnis zu stabilisieren [1]. Das beiliegende Python-Skript `showOffset.py` können Sie für die Analyse benutzen. Abbildung 1 im Anhang zeigt das Ergebnis. Je stärker die zulässige Bandbreite (256 – 5 Hz) limitiert wird, desto stabiler wird der Wert. Für unsere einfache Anwendung ist ein Band von 5Hz ausreichend (MPU6050_DLPF_BW_5). Versuchen Sie nicht den Offset im MPU 6050 zu kompensieren, diese (vom Hersteller undokumentierte) Funktion ist zwar in der Open Source Library enthalten, funktioniert aber nicht! Die Auflösung des AD-Wandlers des MPU 6050 beträgt 16 Bit. Als maximale Drehrate ist im Defaultzustand +-250°/s (für unsere Anwendung mehr als ausreichend!) vorkonfiguiert.

Bestimmen Sie die Zeitkonstante, mit der Sie die Integration durchführen. Zeichnen Sie dazu den Takt der Hauptschleife auf und berechnen Sie eine mittlere Taktdauer. Da die UART-Kommunikation nicht echtzeitfähig ist, muss bei deren

[1] http://invensense.com/mems/gyro/documents/RM-MPU-6000A-00v4.2.pdf , Seite 13

Benutzung ein Rauschen von einigen Millisekunden toleriert werden. Leiten Sie aus den gewonnenen Parametern eine Funktion für die Berechnung der Richtung ab!

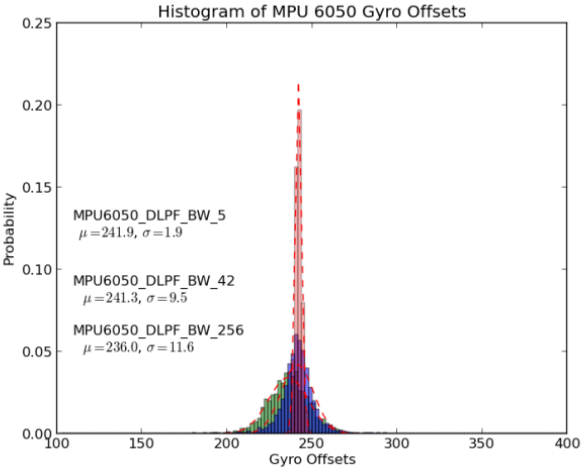

Abbildung 1 – Streuung des Offsets des Gyroskops

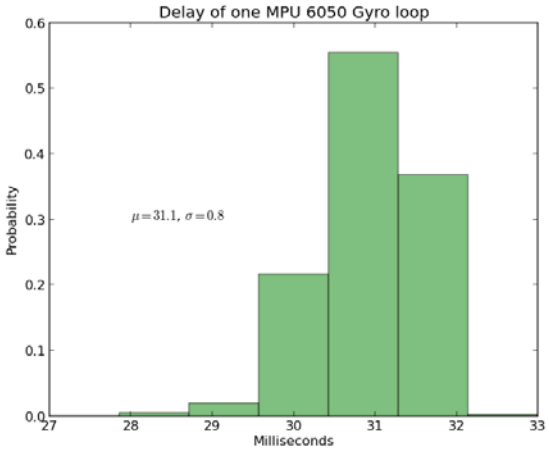

Abbildung 2 – Streuung des Taktzyklus der Hauptschleife

Teilaufgabe 2

Implementieren Sie eine einfache Verfahrfunktion für den Roboter, so dass dieser sich kollisionsfrei auf dem Labortisch bewegt. Sie können dazu neben den 2 obligatorischen Distanzsensoren auch die IMU (zum Beispiel für spannungsunabhängige Drehbewegungen) verwenden.

Die Implementierung der Motoransteuerung soll nicht mit den `analogWrite()`-Funktionen der Arduino Bibliothek erfolgen, sondern allein mit der avrlibc, da die Arduino-Bibliothek an dieser Stelle nur sehr eingeschränkte Konfigurationen zulässt. So kann zum Beispiel die Auflösung des Zählers nicht konfiguriert werden.

Werten Sie also zunächst das Datenblatt des Motor-Shields und des Arduino-Boards aus, um zu ermitteln, welche Timer und welche PWM mit den beiden Motortreiberkanälen verbunden sind. Konfigurieren Sie die Kanäle als Fast-PWM.