Praktikum 2 Intelligente Sensortsysteme

Tim Tiedemann, Thomas Lehmann, Tobias de Gasperis

Einfache intelligente Sensoren und Datenvorverarbeitung

Im Praktikum 2 geht es zum einen um die Eigenschaften eines ersten einfachen intelligenten Sensors, sowie eine erste Datenvorverarbeitung mittels Mikrocontroller.

Lesen Sie sich die Aufgaben gut durch. Sollten Sie eine Aufgabe nicht lösen können, so beschreiben Sie zumindest, wie weit Sie gekommen sind und auf welche Weise Sie vorgegangen sind.

Die Aufgaben sind direkt hier als Protokoll zu bearbeiten. Das abgegebene Notebook soll ausführbar sein. Daneben ist der PDF-Export des Notebook mit abzugeben.

Autoren des Protokolls: Haron Nazari, Anton Tchekov

Hintergrund

Bewegungssensoren sind inzwischen trotz ihrer Komplexität Massenware und finden sich in verschiedenen Anwendungen. Als Beispiele sollen in diesem Praktikum ein Joystick und eine Alarmanlage auf Basis von Bewegungssensoren als Sensorsystem entwickelt werden. Dazu soll zunächst das Verhalten dieser Sensoren und die Kommunikation mit komplexeren Sensoren untersucht werden.

Oftmals ergibt sich die Schwierigkeit, dass Daten nicht direkt verarbeitet werden können, sondern zwischengespeichert und später übertragen werden. Die dabei entstehenden Probleme und Lösungsansätze sollen ebenfalls untersucht werden.

Im Rahmen der folgenden Praktika sollen Bewegungsmuster oder Phasen der Bewegung automatisch erkannt und klassifiziert werden (Stichwort Gestenerkennung). Die hierfür benötigten Rohdaten für die nachfolgenden Analysen sollen hier gesammelt werden.

Vorbereitungsaufgaben

Beschleunigungs- und Gyroskop-Sensoren

Beschaffen Sie sich die Datenblätter zu den Sensoren ST LSM6DS0, LIS2DW12 und LIS2MDL. Welche Quellen für Datenblätter kennen Sie und welche haben Sie warum gewählt?

Datenblätter

Wir haben die Datenblätter von ST (dem Hersteller) gewählt, da wir denken, dass diese Quelle am zuverlässigsten ist.

Um was für Sensoren handelt es sich jeweils? Beschreiben Sie kurz die Funktionsweise und wichtigsten technischen Parameter, die für die Praktikumsaufgabe relevant sein könnten!

Funktionsweise

Die Sensoren sind mikroelektromechanische Systeme (MEMS) also sehr kleine bewegliche Strukturen, die direkt auf einem Chip aufgebaut sind.

Der Accelerometer funktioniert über einen Kondensator, dessen Kapazität sich verändert, wenn eine Beschleunigung stattfindet. Eine Kondensatorplatte ist stationär, die andere beweglich auf einer Biegefeder. An der beweglichen Kondensatorplatte ist eine Probemasse angebracht. Bei einer Beschleunigung verändert sich die Kapazität des Kondensators, da die Probemasse durch ihre Trägheit an der gleichen Stelle verbleiben "will", während der Sensor selbst verschoben wird. Die Kapazität des Kondensators wird vom Chip gemessen und über die I2C Schnittstelle an den Master weitergegeben.

Das Gyroskop funktioniert über die Anwendung der Korioliskraft, also der Massenträgheit von Objekten relativ zu einem rotierenden Bezugssystem.

Die Umsetzung erfolgt über eine periodische Primärbewegung, die piezoelektrisch erzeugt wird, welche eine mittlere Elektrode zum Schwingen bringt.

Wenn eine Drehbewegung auf den Sensor wirkt, entsteht eine Sekundärbewegung die senkrecht zur Primärbewegung auf den seitlichen Außenelektroden, die wieder über die Kapazität, gemessen werden kann.

Bei beiden Sensorentypen gibt es jeweils drei Teile für jede Raumachse (X, Y, Z).

LIS2MDL: 3-axis magnetometer

LIS2DW12: 3-axis accelerometer

LSM6DS0: 3D accelerometer and 3D gyroscope

Über welche Kommunikationsschnittstelle(n) kann/können Messwerte der Sensoren (s.o.) ausgelesen werden. Wird/werden diese vom Mikrocontroller auf dem Nucleo-Board unterstützt? Können dafür bestimmte Funktionseinheiten innerhalb des Mikrocontrollers genutzt werden?

Schnittstellen

Alle drei Sensoren (**LIS2MDL**, **LSM6DS0**, **LIS2DW12**) können sowohl über I²C als auch SPI angesteuert werden. Das ITS-Board unterstützt beide Schnittstellen in Hardware. Der Code spricht alle Sensoren über I²C an.

Einarbeitung in den Demo-Code

Analysieren Sie das über Git/EMIL/MS-Teams zur Verfügung gestellte Mikrocontroller-Projekt. Was wird da gemacht? Wo findet ein Zugriff auf die Sensoren statt? Wo die initiale Konfiguration der Mikrocontroller-internen Komponenten? An welcher Stelle werden vermutlich Zugriffe auf das "whoami"-Register durchgeführt?

Insbesonere: Welches Kommunikationsschnittstelle verwenden die High-Level-Schnittstellen zu den Sensoren und an welcher Stelle wird diese konfiguriert? Somit auch an welchen Pins des Controllers sind die Sensoren angeschlossen?

Optional: Verfolgen Sie im Praktikum per Debugger die Ausführung der Zugriffsfunktionen durch die Schichten der Bibliothek. Wo erfolgt wirklich der Zugriff auf den Kommunikationskanal?

Analyseergebnis

Die Konfiguration der Mikrocontroller-internen Komponenten erfolgt in der Funktion `initITSboard`. Die Sensoren werden über die Funktionen `init` Initialisiert und mit `enable` Aktiviert. Zum Zugriff auf die I2C Schnittstelle zu den Sensoren wird die STM HAL verwendet. Der Zugriff auf das whoami-Register erfolgt über die `read_id`-Funktion.

Vorbereitung auf das Labor

Lesen sie die Aufgaben, welche im Labor durchgeführt werden sollen durch.

Sammeln Sie alle Fragen und Probleme auf die Sie dabei (oder bei den Vorbereitungsaufgaben) stoßen.

Im Labor

Beschleunigungs- und Gyroskop-Sensoren – die üblichen IMU-Spielereien

Analyse des Sensorverhaltens

Modifizieren Sie das gegebene Microkontroller-Programm derart, dass die Daten in einem CSV- Format ausgegeben werden (Trennung der Dimensionen per Komma oder Tabulator, ggf. konfigurierbar). Es reichen hier die Daten vom LSM6DS0, LIS2DW12 und LIS2MDL (also insgesamt 12 Werte je Zeile). Fügen Sie einen Codeauszug (wenige Zeilen) hier hinzu.

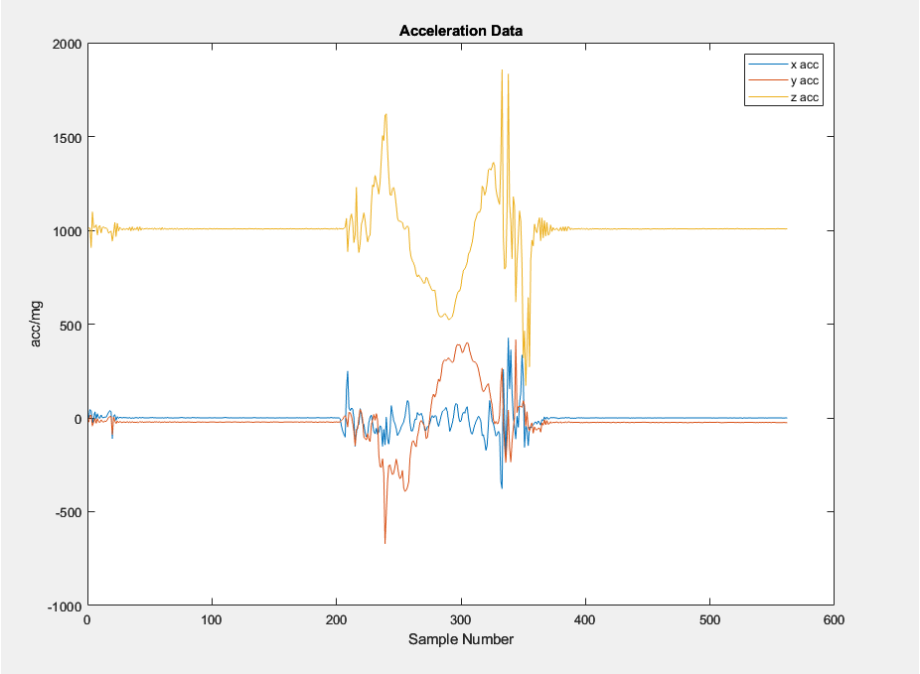
Code-Sample

```
for(;;)
{
        acc_gyro.get_a_axes(&axes[0]);
        acc_gyro.get_g_axes(&axes[3]);
```

```
        accelerometer.get_a_axes(&axes[6]);
        magnetometer.get_m_axes(&axes[9]);
        printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",
                axes[0], axes[1], axes[2], axes[3], axes[4], axes[5],
                axes[6], axes[7], axes[8], axes[9], axes[10], axes[11]);
        delay(1500);
}
```

Werten Sie die ausgegebenen Beschleunigungs- und Gyroskopwerte aus: In welchem Bereich liegen diese, wenn das Board ruhig auf dem Tisch liegt? Ist der Bereich bei allen drei Achsen X/Y/Z jeweils gleich? Wie ändert sich die Ausgabe, wenn Sie das Board geneigt halten? Fügen Sie diesem Protokoll geeignete Plots von aufgezeichneten Daten hinzu, mit drei Achsen x/y/z von einem (oder mehreren) der Sensoren, aufgenommen während einer Bewegung wie z.B. “Ruhelage auf dem Tisch – Bewegung – Ruhelage auf dem Tisch”. Datendateien sind ebenfals abzugeben, so dass das Notebook ausführbar bleibt.

Datenanalyse




```
%% Workspace säubern
clc;
clear;
close all;

%% a. Importieren der Dateien
```

```
DataIn = readtable("daten.csv");
samples = DataIn.Samples;

x_acc = DataIn.x_acc;
y_acc = DataIn.y_acc;
z_acc = DataIn.z_acc;

plot(samples, x_acc, samples, y_acc, samples, z_acc);

title("Acceleration Data");
xlabel("Sample Number");
ylabel("acc/mg");

legend("x acc", "y acc", "z acc");
```

Überlegen Sie, wie Sie Beschleunigungswerte generieren könnten, die vom Betrag möglichst klein sind (gleichzeitig auf allen drei Achsen und über mehrere Samples hinweg). Wie können Sie dies erreichen? Welche Werte erreichen Sie?

Hinweis: Sie könenn für Experimente auch ein Smartphone und die App "phyphox" der RWTH Aachen verwenden. Was erwarten Sie, was Sie währenddessen auf den drei Gyroskopachsen messen? Was messen Sie?

Lösung

Man könnte gleichzeitig auf allen drei Achsen den selben Wert bekommen indem man das Board so neigt, dass die Schwerkrft auf allen drei Achsen wirkt.

Die Erwartung ist, dass die Gyroskopachsen Werte nahe 0 ausgeben, da das Board nicht gedreht wird.

In der Realität zeigt der Gyro auch Werte an, obwohl das Board nicht gedreht wird.

Anwendungsentwicklung

Es sollen die beiden Anwendungen Joystick und Alarmanlage entwickelt werden. Erstellen Sie vorab zwei Kopien des Projektes, in denen Sie dann die Anwendungen entwickeln.

Joystick

Verändern Sie die Ausgabe Ihres Microcontroller-Programms derart, dass das Board als User- Interface genutzt werden kann. Geben Sie beispielsweise einen "X"-Wert und einen "Y"-Wert eines simulierten Joysticks aus. Beide Ausgabewerte sollen in waagerechter Ausrichtung des Boards 0 sein und mit Neigung um eine Achse in eine Richtung positiv und in die andere Richtung derselben Achse negativ werden. Bei ungefähr 45° Neigung soll der Wert 45 ausgegeben werden.

Fügen Sie einen C-Codeauszug (wenige Zeilen) hier hinzu. Geben Sie die vollständige Source-Datei mit ab (ohne Projektdateien!).

Code-Snippet

```
#define PI 3.14159f
#define RAD_TO_DEG (180.0f / PI)

void l_joystick(void)
{
        printf("> Joystick\r\n\r\n");
        int32_t axes[3];

        float x1, y1, z1;
        float jX, jY;

        for(;;)
        {
                acc_gyro.get_a_axes(axes);
                x1 = axes[0] / 1000.0f;
                y1 = axes[1] / 1000.0f;
                z1 = axes[2] / 1000.0f;

                jX = RAD_TO_DEG * (atan2(-y1, -z1) + PI);
                jY = RAD_TO_DEG * (atan2(-x1, -z1) + PI);

                if(jX > 270.0f) { jX -= 360.0f; }
                if(jY > 270.0f) { jY -= 360.0f; }

                printf("X: %5.2f | Y: %5.2f\r\n", jX, jY);
        }
}
```

Alarmanlage

Konfigurieren Sie einen Trigger (Bewegungsschalter? Alarmanlage?), der bei Ruhelage des Boards 0 ist (oder keine Ausgabe generiert). Bei Bewegung aus der Ruhelage (entweder um/entlang allen Achsen oder nur um/entlang einer) soll eine 1 ausgegeben werden. Es soll weder "1"-Ausgaben in Ruhelage geben ("false positive") noch "0"-Ausgaben bei Bewegung ("false negatives"). Fügen Sie einen C-Codeauszug (wenige Zeilen) hier hinzu. Geben Sie die vollständige Source-Datei mit ab (ohne Projektdateien!).

Code-Snippet

```
#define BASELINE_SAMPLE 1024
#define BASELINE_SENSI   20
```

```
void l_alarm(void)
{
        int32_t min_acc[3] = { INT_MAX, INT_MAX, INT_MAX };
        int32_t max_acc[3] = { INT_MIN, INT_MIN, INT_MIN };
        int32_t changed_acc[3];
        int32_t i;

        printf("> Alarmanlage\r\n\r\n");

        for(i = 0; i < BASELINE_SAMPLE; ++i)
        {
                acc_gyro.get_a_axes(changed_acc);

                if(changed_acc[0] < min_acc[0]) { min_acc[0] = changed_acc[0]; }
                if(changed_acc[1] < min_acc[1]) { min_acc[1] = changed_acc[1]; }
                if(changed_acc[2] < min_acc[2]) { min_acc[2] = changed_acc[2]; }

                if(changed_acc[0] > max_acc[0]) { max_acc[0] = changed_acc[0]; }
                if(changed_acc[1] > max_acc[1]) { max_acc[1] = changed_acc[1]; }
                if(changed_acc[2] > max_acc[2]) { max_acc[2] = changed_acc[2]; }

                delay(1);
        }

        max_acc[0] += BASELINE_SENSI;
        max_acc[1] += BASELINE_SENSI;
        max_acc[2] += BASELINE_SENSI;

        min_acc[0] -= BASELINE_SENSI;
        min_acc[1] -= BASELINE_SENSI;
        min_acc[2] -= BASELINE_SENSI;

        for(;;)
        {
                acc_gyro.get_a_axes(changed_acc);
                if(changed_acc[0] < min_acc[0] ||
                        changed_acc[1] < min_acc[1] ||
                        changed_acc[2] < min_acc[2] ||
                        changed_acc[0] > max_acc[0] ||
                        changed_acc[1] > max_acc[1] ||
                        changed_acc[2] > max_acc[2])
                {
                        GPIOD->BSRR = (1 << 0);
                }
        }
}
```

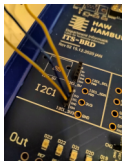
Kommunikationsanalyse

Beschaffen Sie sich den Schaltplan des Nucleo-Boards IKS01A3 (siehe EMIL-/MS-Teams- Raum). Fragen Sie ggf. die Laborassistenz.

An welchen Kontakten in ihrem Gesamtsystem können die Signale des I2C-Buses gemessen werden?

Anschlüsse

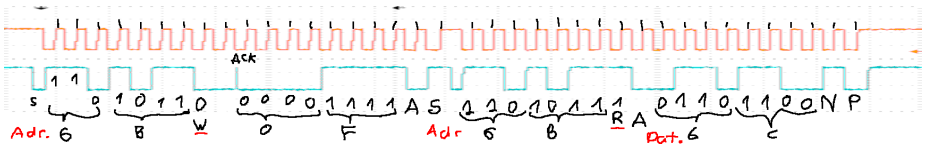
Die Signale des I2C-Busses können an I2C1 gemessen werden.



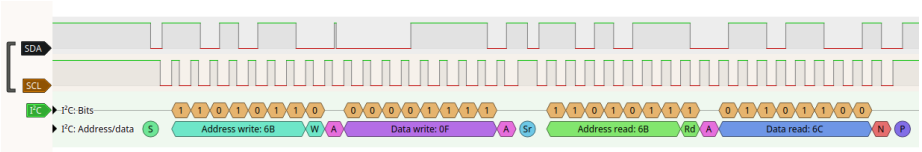
Untersuchen Sie den Zugriff auf das Who-Am-I-Register eines Sensors. Führen Sie eine Messung des Zugriffs mit dem Oszilloskop durch und betrachten Sie einen Bustransfer. Wie wird der Sensor hier angesprochen? Wie antwortet der Sensor? Fügen Sie einen “Screen Shot” der Übertragung hier mit ein. Analysieren Sie die Signale nach dem Protokollstandard und ergänzen Sie Ihre Ergebnisse (Bits etc.) in dem Signalverlauf.

Hinweis: Es müssen ggf. mehrere Messungen durchgeführt werden und zu einem Screen Shot zusammengesetzt werden.

Protokollanalyse



Kontrolle mit Logic Analyzer:



Welche Datenrate im Sinne von Messungen/s (Datenzeile/s) kann Ihr Gesamtsystem über die serielle Schnittstelle (RS232) in der Konfiguration aus der Teilaufgabe "Analyse des Sensorverhaltens" zum Host maximal übertragen? Kein Delay im Programm. Schätzen Sie den Wert begründet ab.

Datenrate

Schätzung der Datenrate:

Serielle Schnittstelle Geschwindigkeit: 115200 Baud -> 11520 Zeichen pro Sekunde (8 Datenbits + 1 Startbit und 1 Stoppbit)

Bei durchschnittlich 50 Zeichen pro Datenzeile sind dauert die Übertragung 4.3 ms.

Das Auslesen aller 12 Werte vom Sensor dauert etwa 6.2 ms.

Also insgesamt 10.5 ms, das sind also ungefähr 95 Datenzeilen pro Sekunde.

Feature-Generation / -Selection

Data Buffer

Im Folgenden sollen Messwerte zunächst lokal auf dem Mikrocontroller gesammelt und dann getrennt weiterverarbeitet werden. Ändern Sie Ihr Mikrocontroller-Programm (am besten als neue C-Datei oder neues Projekt bzw. sichern Sie die C-Datei aus der letzten Aufgabe). Die Messwerte sollen in einem Puffer gesammelt werden, ohne diese gleich an den Host-PC zu übertragen. Die Puffergröße soll dabei (zur Compile-Zeit) konfigurierbar sein. Sie brauchen mindestens 1024 Samples von

1. den drei Beschleunigungsachsen des LIS2DW12,
2. den drei Beschleunigungsachsen des LSM6DS0,
3. den drei Gyroskopachsen des LSM6DS0,
4. den drei Magnetometerachsen des LIS2MDL

(also insgesamt mindestens 1024 * 12 Integerwerte). Geben Sie danach den Inhalt des Pufferspeichers (alle mindestens 1024 12-dimensionalen Samples) in einem CSV-kompatiblen Format über die serielle Schnittstelle aus. Implementieren Sie beide Speicherzugriffe (Messwerte speichern und Daten ausgeben) innerhalb derselben Funktion (main oder eine andere).

Entfernen Sie den Aufruf “wait(1.5);”, falls noch nicht geschehen. Wiederholen Sie den Ablauf Datensammlung–Datenausgabe in einer Endlosschleife. Stellen Sie die Baudrate von mindestens 115.200 Baud ein. Fügen Sie einen Codeauszug hier hinzu-

Code Snippet

```
#define NUM_SAMPLES  1024

void l_buffer(void)
{
        static int32_t samples[NUM_SAMPLES * 12];
        int32_t *cur;
        int32_t i;

        printf("> Buffer\n\n");
        for(;;)
```

```
        {
                cur = samples;
                for(i = 0; i < NUM_SAMPLES; ++i)
                {
                        acc_gyro.get_a_axes(cur);
                        cur += 3;
                        acc_gyro.get_g_axes(cur);
                        cur += 3;
                        accelerometer.get_a_axes(cur);
                        cur += 3;
                        magnetometer.get_m_axes(cur);
                        cur += 3;
                }

                cur = samples;
                for(i = 0; i < NUM_SAMPLES; ++i)
                {
                        printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",
                                cur[0], cur[1], cur[2], cur[3], cur[4], cur[5],
                                cur[6], cur[7], cur[8], cur[9], cur[10], cur[11]);
                        cur += 12;
                }
        }
}
```

Deklarieren Sie den Pufferspeicher einmal lokal (in der main- oder ggf. der eigenen Funktion) und ändern Sie die Größe auf 1.000.000 Samples. Kompilieren Sie das Programm. Ist dies möglich, was passiert? Wie groß ist der benötigte Speicher und wieviel Speicher steht in dem Mikrocontroller zur Verfügung?

Ergebnis

Das Programm kompiliert, kann aber nicht funktionieren, da auf dem ITS-Board 192 KiB RAM vorhanden sind, was für einen so großen Puffer nicht ausreicht. Bei 1.000.000 Samples mal 12 Werte je 4 byte (32-bit integer) sind das 48 MB die benötigt werden.

Deklarieren Sie den Pufferspeicher global und wiederholen Sie den Versuch, einen 1.000.000 Samples großen Zwischenspeicher zu verwenden. Was passiert nun?

Ergebnis

Da es sich um eine globale Variable handelt, wird der Platzbedarf zur Compilezeit erkannt und es kommt zu einem Linker error:

```
.\ITSboard\ITSboard.axf: Error: L6406E: No space in execution
regions with .ANY selector matching main.o(.bss.l_buffer.samples).
```

Sampling Duration

Fügen Sie Ihrem Programm ein LED-Objekt hinzu. Alternativ zur mbed-Dokumentation (s.o.) können Sie sich auch das Nucleo-"Hello World"-Beispiel "Blink LED" anschauen. Fragen Sie ggf. Ihre Laborassistenz. Schalten Sie die "LED1" ein, bevor Sie Messdaten in den Pufferspeicher einlesen. Schalten Sie sie wieder aus, bevor Sie die Daten über die serielle Schnittstelle ausgeben. Messen Sie die Zeitdauer für das Einlesen der Daten mittels Oszilloskop.

Zeitdauer

Zeitdauer für 1024 Samples: 6.328 Sekunden

Timestaps

Nicht jeder Schleifenduchlauf ist gleich lang und oftmals benötigt mann eine genaue Information darüber wann die Daten (Sampling) erfasst wurden.

Fügen Sie Ihrem Programm einen Timer hinzu. Suchen Sie Informationen hierzu und fragen Sie ggf. Ihre Laborassistenz.

Erfassen Sie den Timerstand (in μs) vor Beginn des Messwertauslesens. Speichern Sie die einzelnen Zeitstempel mit im Puffer der Sensordaten. Geben Sie entsprechend zu Beginn jeder CSV-Zeile einmal den jeweiligen Zeitstempel mit aus.

In welchem Bereich liegt die Zeitdifferenz von einem zum folgenden Sample? Ist der Wert konstant? Wenn ja, in welchem Bereich variiert er? Geben Sie den Datensatz und den Code mit ab.

Zeitanalyse

```
#define NUM_SAMPLES  1024

void l_buffer(void)
{
        static int32_t samples[NUM_SAMPLES * 13];
        int32_t *cur;
        int32_t i;

        printf("> Buffer\n\n");
        for(;;)
        {
                GPIOD->BSRR |= (1 << 1);
                TIM2->CNT = 0;
                cur = samples;
                for(i = 0; i < NUM_SAMPLES; ++i)
                {
                        *cur++ = TIM2->CNT / TICKS_PER_US;
                        acc_gyro.get_a_axes(cur);
```

```
                cur += 3;
                acc_gyro.get_g_axes(cur);
                cur += 3;
                accelerometer.get_a_axes(cur);
                cur += 3;
                magnetometer.get_m_axes(cur);
                cur += 3;
        }
        GPIOD->BSRR |= (1 << (16 + 1));

        printf("----------------------\n");
        cur = samples;
        for(i = 0; i < NUM_SAMPLES; ++i)
        {
                printf("%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\n",
                        cur[0], cur[1], cur[2], cur[3], cur[4], cur[5],
                        cur[6], cur[7], cur[8], cur[9], cur[10], cur[11], cur[12]);
                cur += 13;
        }
        printf("----------------------\n");
    }
}
```

Die Zeitdifferenz ist eigentlich konstant, die Differenz ist immer 6180 / 6181 µs. (siehe timing.csv)

Beispieldatensätze für Klassifikationen

Für das folgende Praktikum benötigen wir mehrere Testdatensätze (siehe unten) (jeweils mindestens 1024 direkt nacheinander aufgenommene Samples lang). Mit Hilfe der LED kann dabei beobachtet werden, wie lange das Füllen des Puffers dauert, so dass die Bewegungen jeweils immer innerhalb dieser Zeit abgeschlossen werden. Nehmen Sie folgende Datensätze auf:

1. Nucleo-Board in Ruhelage auf dem Tisch
2. Nucleo-Board in Ruhelage, dann Drehung zum Aufrichten, Nucleo-Board in Ruhe stehend, Drehung zurück in Liegeposition und wieder in Ruhelage
3. Nucleo-Board in Ruhelage, gradlinige Bewegung in eine Richtung hin und wieder zurück
4. Nucleo-Board in Ruhelage, dann mehrmals gradlinige Bewegung in eine Richtung und wieder zurück
5. Nucleo-Board in Ruhelage, dann unterschiedliche gradlinige und rotatorische Bewegungen – aber alle "liegend" auf dem Tisch, das heißt in einer Ebene
6. Nucleo-Board in Ruhelage, dann diverse unterschiedliche Bewegungen (gradlinig und rotatorisch) über alle drei Raumachsen

Schauen Sie sich den ersten und den zweiten aufgenommenen Datensatz (nach der Liste oben) an. Verhalten sich alle Sensorwerte so, wie Sie es erwartet haben? Wenn nein, welche nicht? Haben Sie eine Vermutung warum nicht? Wenn ja, warum sehen die Messkurven gerade so aus?

Fügen Sie Ihrem Protokoll zwei interessante Plots hinzu. Geben Sie an, an welchen Stellen/in welchen Bereichen welche Bewegung ausgeführt wurde.

Hinweis: Dokumentieren/skizzieren Sie bei den komplexeren Bewegungen (3./4./5.) Ihre Bewegungsmuster, da später die Daten den Bewegungsabschnitten zugeordnet werden sollen.

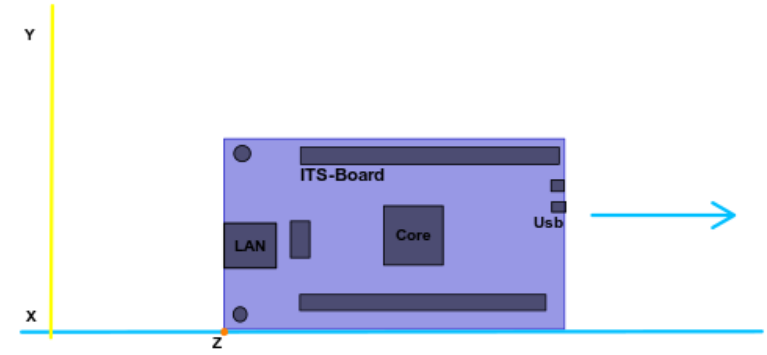
Es ist überraschend wie viel Sensibler das Gyroskop ist, teilweise hilft bei visuellen Plots die Daten des Gyroskops viel mehr als die Daten des Accelerometers.

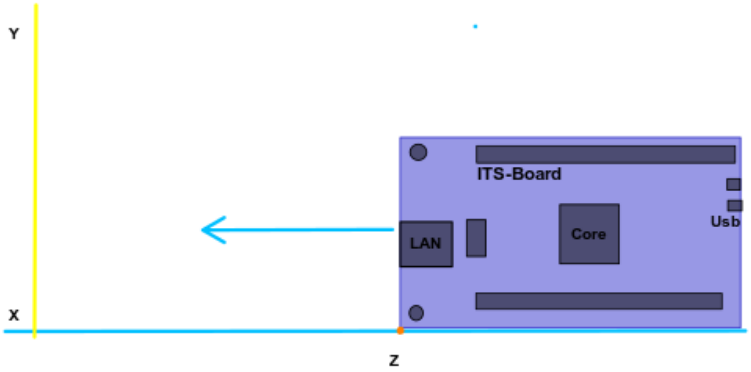
Der erste Accelerometer Plot ist wie man sich denken würde, Es sind relativ Stille linien, wobei die Achse(z) die zur Gravitation ausgerichtet ist, an einem höheren Wert als die anderen aufweist. Der Gyroskop Plot sieht jedoch unerwartet aus. Jede Achse scheint ihren eigenen Werte-Bereich zu haben an denen sie Pendeln. Wir können uns vorstellen, dass die Z und X Achse unterschiedlich von der Schwerkraft beeinflusst werden, wobei die Y-Achse sich auf einer anderen Ebene als die Schwerkraft befindet(wenn in ruhelage auf dem tisch) und somit von der Schwerkraft nicht beeinflusst wird.

Vier und Fuenf waren sehr interessante Plots, da sie unserer Meinung nach visuell viel Aussagen können.

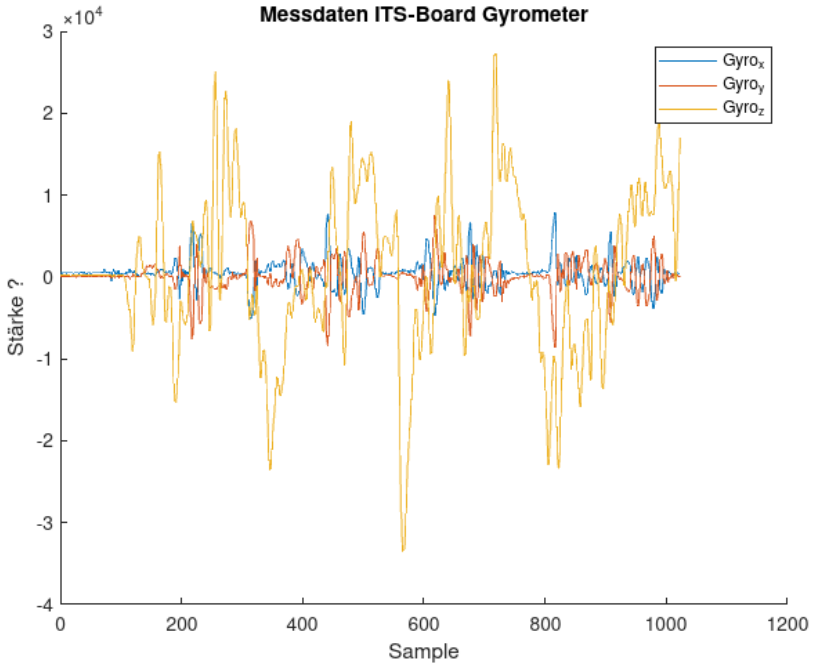
Vier :

Die Bewegung (Mehrmals) :

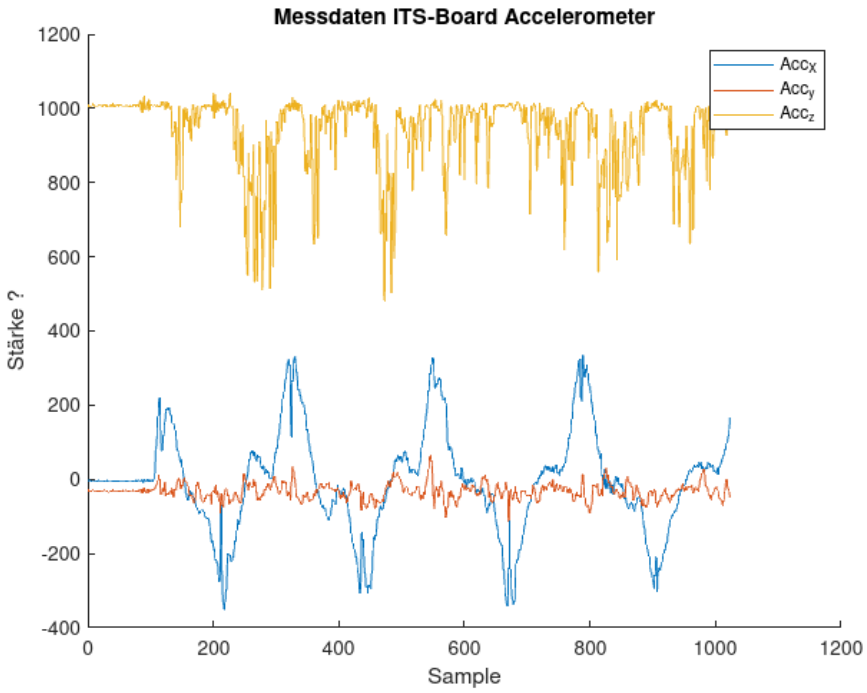




Die Gyroskop Daten :

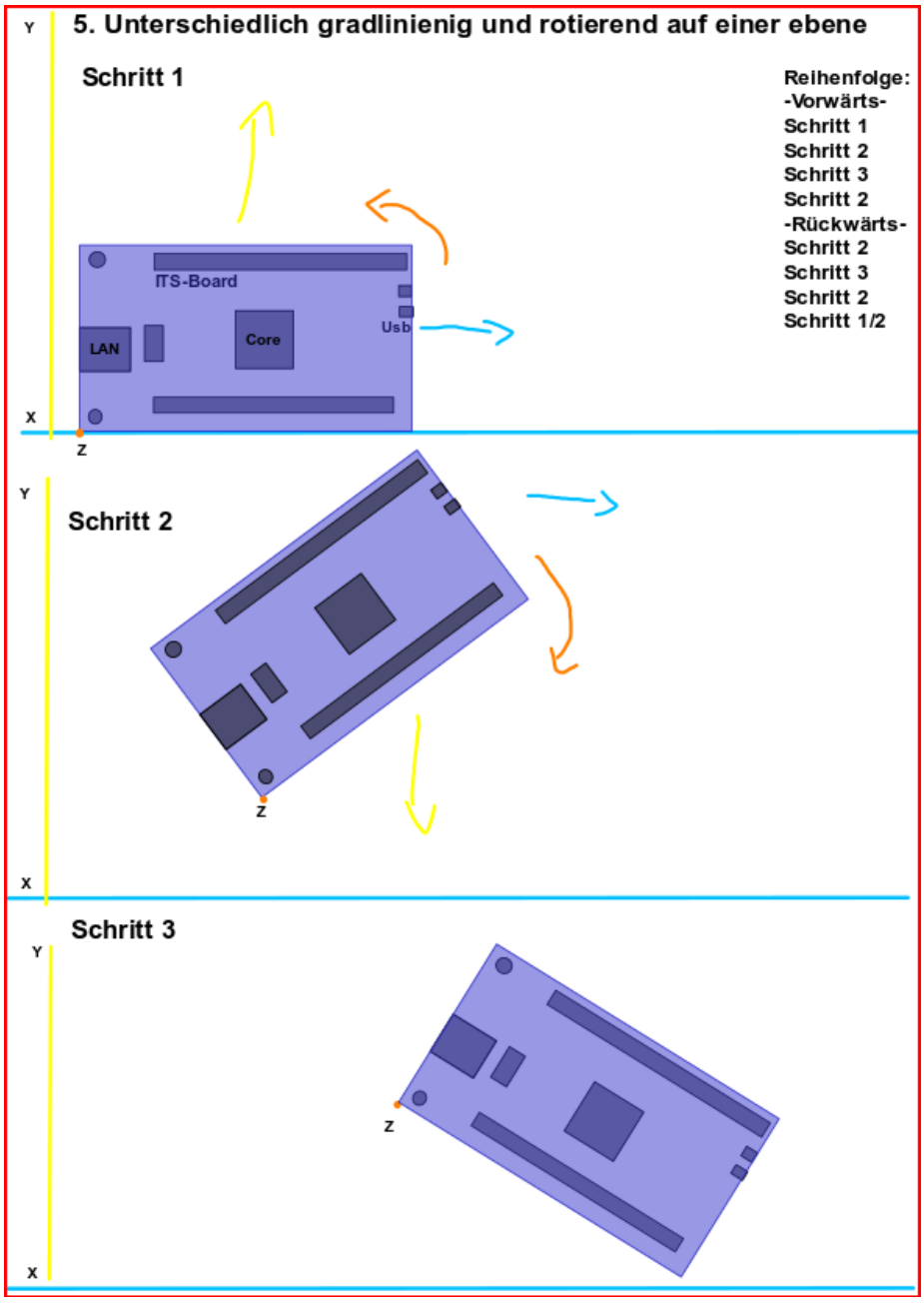


Die Accelerometer Daten :

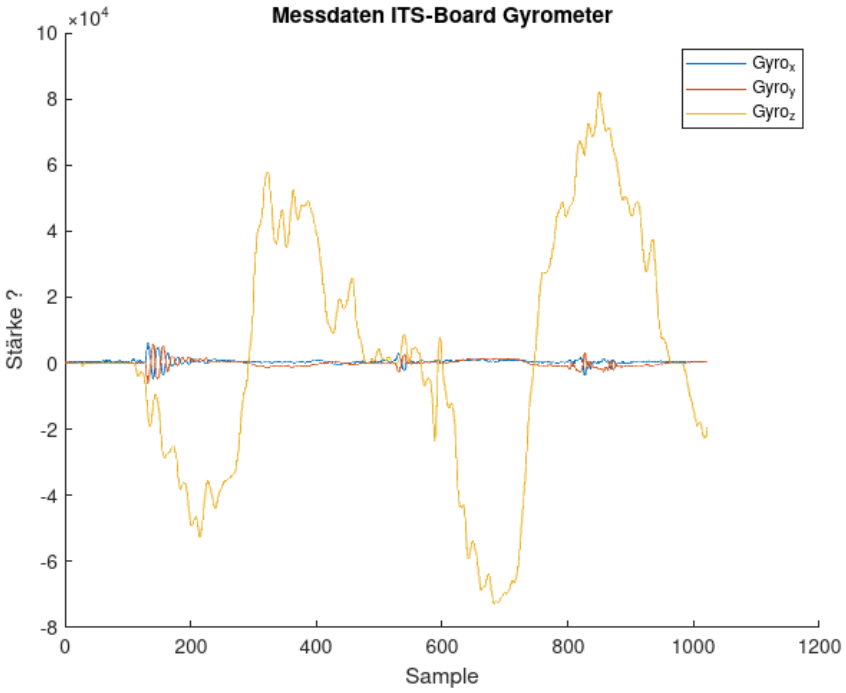


Fuenf :

Die Bewegung :



Die Gyroskop Daten :



Die Accelerometer Daten :