Signale Sigi2.0

3. Dezember 2019

Die Komponenten des Sigi2.0 tauschen untereinander die Signale aus, wie sie in der Graphik auf der nächsten Seite abgebildet sind. Dabei ist zu beachten, dass der Motor ein Drehmoment zurück gibt und die Sensoren geben die Werte der jeweilig gemessenen Grösse zurück. Diese sind einfach Zahlenwerte, die dann mit der entsprechenden Sensitivität multipliziert den Wert in der entsprechenden physikalischen Grösse zurückgeben. Dabei hängt die Sensitivität von der gewählten Full-Scale ab. In den folgenden Abschnitten wird noch genauer auf die einzelnen Hardware-Komponenten eingegangen. Für genauere Spezifikationen (nichtlinearität, Offset, Temperaturabhängigkeit etc.) wird auf das jeweilige Datenblatt verwiesen.

Einheiten

Im hier werden folgende Einheiten verwendet, die weder SI-Einheiten noch davon abgeleitete mit besonderem Namen sind:

Einheit	Beschreibung	Besonderes
LSB	Least Significant Bit	Kleinster Schritt entspricht einer Änderung von 1 der Zahl
g	Erdanziehung	1 g $\approx 9.80665 \frac{m}{s^2}$
dps	Grad pro Sekunde	
gauss	Magnetische Flussdichte	1 gauss $= 10^{-4}T = 10^{-4}\frac{kg}{A \cdot s^2}$ Magnetische Flussdichte des Erdmagnetfelds (Mitteleuropa) $\approx 48\mu T$ ($20\mu T$ horizontal, $44\mu T$ vertikal)

Inertialmodul

Accelerometer

In der folgenden Tabelle werden typische Werte[1] für die Sensitivität des Accelerometers angegeben. Diese Werte gelten für eine Temperatur von $T = 25°C$, die Temperaturabhängigkeit kann dem Datenblatt entnommen werden, und für eine Spannung von $1.8V$

Full-Scale [g]	Sensitivität [mg/LSB]
±2	0.061
±4	0.122
±8	0.244
±16	0.488

[1] ohne Garantie vom Hersteller

Dies kann man auch wie folgt als Formel schreiben, wenn man als FS jeweils die halbe Full-Scale ($FS = 2$ für eine Full-Scale von ± 2) nimmt:

$$Sens\left[\frac{mg}{LSB}\right] = \frac{FS}{2} * 0.061$$

Gyroskop

In der folgenden Tabelle werden typische Werte[2] für die Sensitivität des Gyroskops angegeben. Diese Werte gelten für eine Temperatur von $T = 25°C$, die Temperaturabhängigkeit kann dem Datenblatt entnommen werden, und für eine Spannung von $1.8V$

Full-Scale [dps]	Sensitivität [mdps/LSB]
±125	4.375
±245	8.75
±500	17.5
±1000	35
±2000	70

Im Sensor LSM6DS33 ist auch ein Filter eingebaut, der sich je nach Konfiguration unterschiedlich verhält. Es hat jeweils nach beiden Sensoren zuerst einen Analogen Anti-aliasing LP-Filter, bevor das Signal Digitalisiert wird und danach noch mal zuerst einen LP-Filter und dann noch je nach Sensor eine kompliziertere Filterstruktur. Diese hängt von der Konfiguration ab und kann dem Datenblatt entnommen werden.

Magnetometer

In der folgenden Tabelle werden typische Werte[3] für die Sensitivität des Magnetometers angegeben. Diese Werte gelten für eine Temperatur von $T = 25°C$, die Temperaturabhängigkeit kann dem Datenblatt entnommen werden, und für eine Spannung von $2.5V$

Full-Scale [gauss]	Sensitivität [LSB/gauss]
±4	6842
±8	3421
±12	2281
±16	1711

Motoren

Die Ansteuerung der Motoren beginnt mit einem `int16` Signal vom Raspberry Pi an den ATmega 32U4. Die möglichen Werte dieses Signals hängen davon ab, ob der „turbo mode“ aktiviert ist. In diesem Fall kann das Signal einen Zahlenwert im Bereich [-400, 400] annehmen. Hier entspricht ein positives, bzw. negatives Vorzeichen einer Vorwärtsrotation, bzw. einer Rückwärtsrotation. Zudem ist im „turbo mode“ die betragsmässig grösstmögliche Drehgeschwindigkeit abrufbar. Allerdings ist häufige Verwendung des „turbo mode“ schädlich für die Motoren. Aus diesem Grund arbeitet der Sigi im „normal mode“, bei dem sich das Signal im Bereich [-300, 300] befinden kann. Zahlenwerte ausserhalb dieses Bereichs werden auf -300, bzw. 300 gerundet.

Der ATmega 32U4 wendet Pulsdauermodulation (PWM) an, um das Geschwindigkeitssignal des Raspberry Pi in eine Schrittdauer der Motoren umzuwandeln. Jedes dieser Signale besteht aus einem „prescalar“ und „phase-correct“. Da der ATmega 32U4 mit den Motoren mit 16MHz

[2] ohne Garantie vom Hersteller
[3] ohne Garantie vom Hersteller

kommuniziert, jeder Puls aus maximal 400 Schritten besteht und über ein Vorzeichen verfügt, liegt die maximale ansteuerbare Frequenz der Motoren bei 20KHz. Dies kann der .cpp-Datei des Microcontrollers entnommen werden.

Bevor das Signal die Motoren erreicht, wird es von dem Motortreiber DRV8838 von Texas Instruments verarbeitet. Dieser ist in der Lage, eine Spannung an die Motoren anzulegen, wenn das pulsdauer-modulierte Signal es vorgibt.

Letztendlich produzieren die Motoren eine gewisse Drehgeschwindigkeit, welche von der anliegenden Last abhängig ist. Dies kann aus dem Datenblatt der Motoren entnommen werden. Unsere Versuche zeigen, dass die anliegende Last am Sigi klein genug ist, um vernachlässigt zu werden. Somit ist die erzielte Drehzahl als Funktion des ursprünglichen Signals („input“):

$RPM_{Rad} = \frac{RPM_{Max,Motor}}{i_{Getriebe}} \cdot \frac{input}{400}$

Beispiel: Berechnung der maximalen Drehzahl im „normal mode“ unter Verwendung eines HPCB 6V Motors mit interner Übersetzung von 50:1 und mittlerer Getriebestufe. Dem Datenblatt ist einer Freilaufdrehzahl von 650rpm und einer Getriebeübersetzung von 2.14:1 zu entnehmen. Experimentell haben wir einen Wert von zirka 225rpm am Sigi gemessen. Um die Maximaldrehzahl im „normal mode“ zu erreichen, muss das Eingangssignal ±300 betragen. Es folgt:

$RPM_{Rad} = \frac{650}{2.14} \cdot \frac{300}{400} = 227.8rpm \approx 225rpm$

Encoder

Die Encoder geben jeweils auf 2 Pins des Atmega 32U4 ein digitales Hi/Low Signal. Eines davon gibt, ob sich die Rotation des kleinen Magnets verändert hat und das andere gibt die Richtung zurück. Dabei werden 12 Ticks pro Umdrehung des kleinen Magnetes detektiert. Dies wird dann auf dem Mikrocontroller mittels Interrupts aufsummiert und dann wird die totale Anzahl Ticks an den Raspberry Pi weitergeleitet.
Die Umdrehungen des Magnets sind dabei direkt an die Umdrehung des Motors gekoppelt. Daher hängt die Anzahl Ticks pro Umdrehung des Rads sowohl vom eingebauten Getriebe im Motor als auch vom zusätzlichen Getriebe, das beim Zusammenbau des Pololu Balboa 32U4 ausgewählt wird, ab.

Buttons

Die Buttons sorgen für eine Unterschiedliche Spannung auf den zugehörigen Pins des Atmega 32U4. Diese hat 2 Zustände (Hi/Low), wobei Hi die Standardspannung ist und wenn der Knopf gedrückt wird, wird die Spannung gesenkt. Dies wird vom Programm einfach in einen Booleschen Wert umgewandelt, wobei 1 für einen gedrückten Knopf steht und 0 für den Nichtgedrückten Knopf.

LEDs

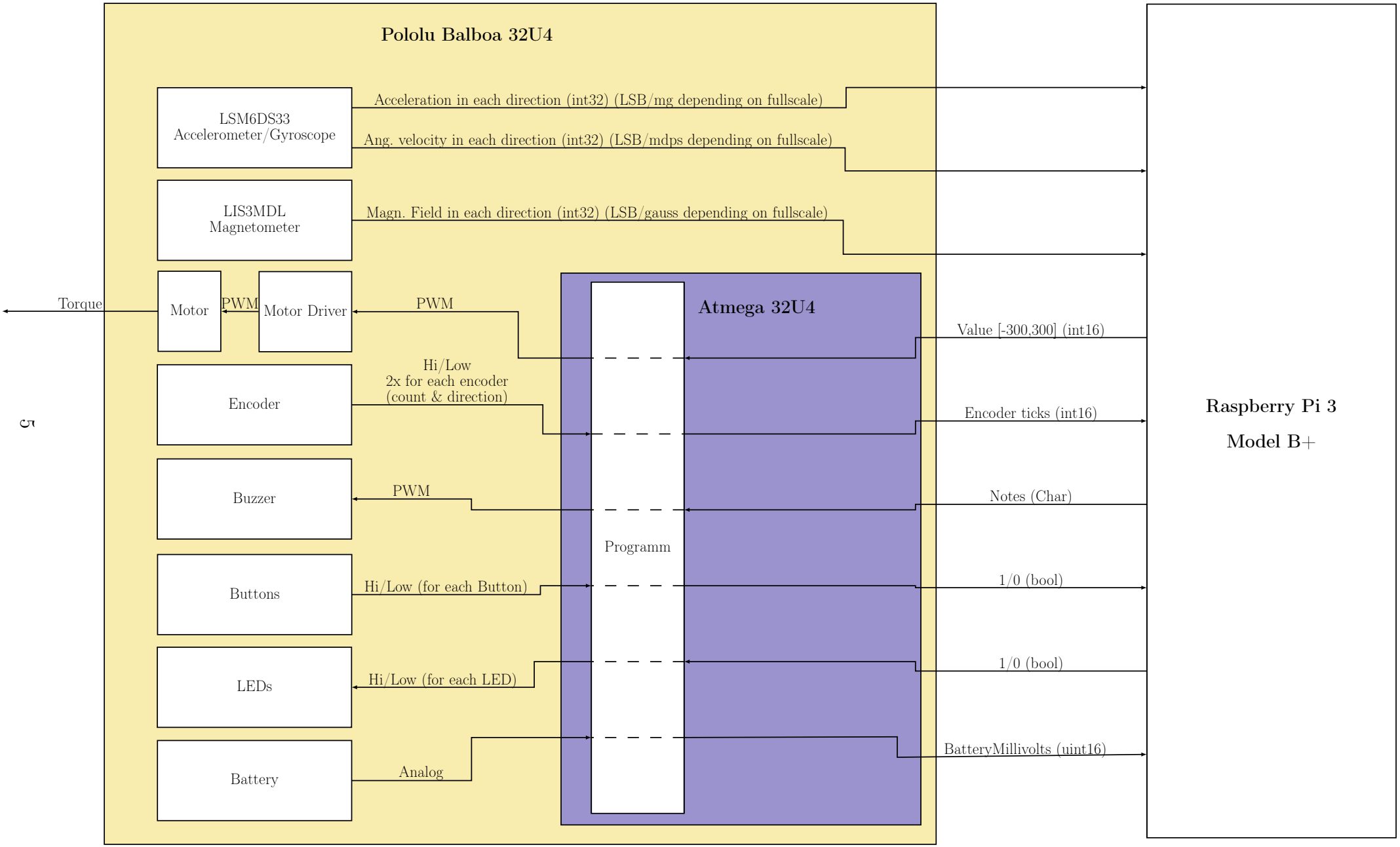
Die LEDs werden vom Raspberry Pi durch das Senden eines Booleschen Wertes an den Atmega 32U4 gesetzt. Das Programm, setzt dann jeweils den Richtigen Pin, was die LED dann an-/ausschaltet.
Gewisse LEDs werden vom Arduino auch benutzt um anzuzeigen ob Daten über den USB-Port übertragen werden. Daher kann es schwer sein die LEDs zu steuern, wenn der USB-Port angeschlossen ist.

Batterie

Die Batterie ist direkt an den ADC (Analog-Digital-Konverter) des Atmega 32U4 angeschlossen. Dieser wandelt dann die Spannung in einen Digitalen Wert um, der dann in $[mV]$ umgerechnet wird. Die Spannung in mV wird dann an den Raspberry Pi weitergegeben.
Es ist zu beachten, dass falls diese zurückgegebene Spannung unter $5500mV$ fällt, die eigentliche Batteriespannung signifikant tiefer sein kann.

Buzzer

Der Rasperry Pi kann dem Atmega 32U4 auch töne senden, die dieser dann abspielt. Der Atmega 32U4 ermittelt mittels einer Tabelle zuerst die Frequenz, die es abzuspielen gilt. Diese wird dann in die nötige Frequenz der Pulsdauermodulation, mittels der die Information an den Buzzer gesendet wird, umgerechnet und ebenfalls der der Lautstärke entsprechende Tastgrad ermittelt. Dann wird das so ermittelte Signal an den Buzzer gesendet, der dann die Note abspielt.


Pololu Balboa 32U4
LSM6DS33
Accelerometer/Gyroscope
Acceleration in each direction (int32) (LSB/mg depending on fullscale)
Ang. velocity in each direction (int32) (LSB/mdps depending on fullscale)
LIS3MDL
Magnetometer
Magn. Field in each direction (int32) (LSB/gauss depending on fullscale)
Torque
Motor
PWM
Motor Driver
PWM
Atmega 32U4
Value [-300,300] (int16)
Hi/Low
2x for each encoder
(count & direction)
Encoder
Encoder ticks (int16)
Raspberry Pi 3
Model B+
Buzzer
PWM
Notes (Char)
Programm
Buttons
Hi/Low (for each Button)
1/0 (bool)
LEDs
Hi/Low (for each LED)
1/0 (bool)
Battery
Analog
BatteryMillivolts (uint16)