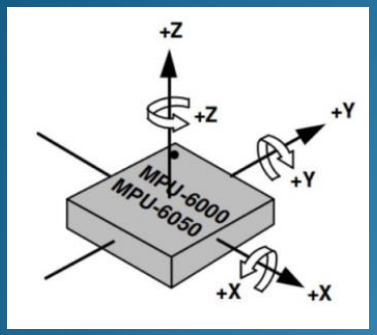
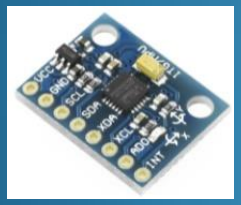
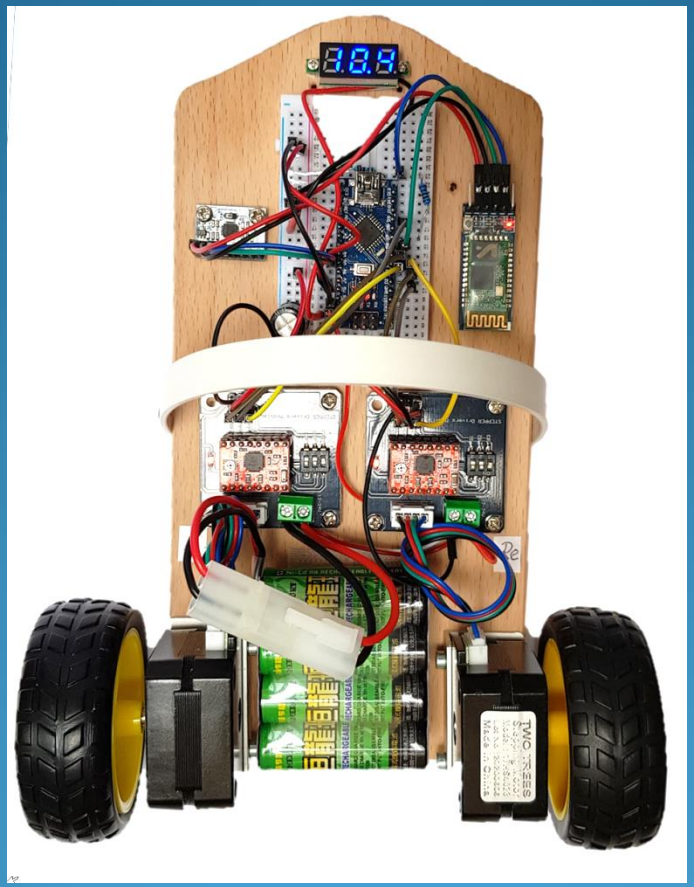
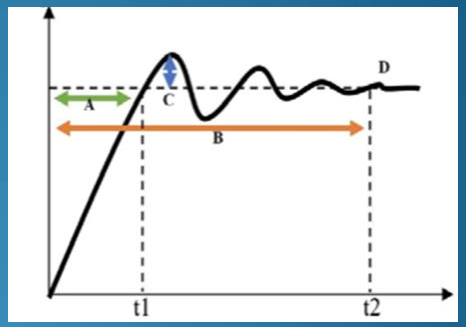
GTA Mechatronik Teil 4

Wir bauen und programmieren einen Mini-Segway und lernen dabei viel über Beschleunigungs- und Lagesensoren, Regler und Schrittantriebe









Inhaltsverzeichnis

Tutorial

Komponenten und Aufbau
PID-Regler
Lagemessung
Lagesensor-Chip MPU-6050
MPU6050_tockn.h
Schrittmotor
Schrittmotor-Treiber
Bluetooth
Bluetooth Modul HC-05

Sketche

Zuordnung Pins am Arduino Nano
Sketch 81_I2C Bus Scan
Sketch 82_MPU6050 Daten auslesen
Sketch 83_MPU6050_tockn
Sketch 84_Schrittmotoren Test
Sketch 85_Modul HC-05 konfigurieren
Sketch 86 Bluetooth Kommunikation
Sketch 87_Segway Balance
Sketch 88_Segway Fahren
Sketch 88_Regelung

Anhang

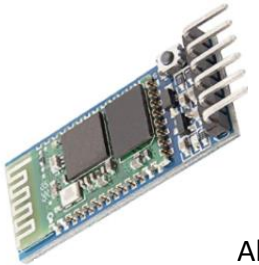
Studyflix
ASCII-Code
Portmanipulation
Bitoperationen
Laufzeit Loop Sketche
App „Arduino Bluetooth Controller“

Komponenten und Aufbau

Modul GY-521
mit MPU-6050
Gyrosensor

Anzeige
Akkuspannung

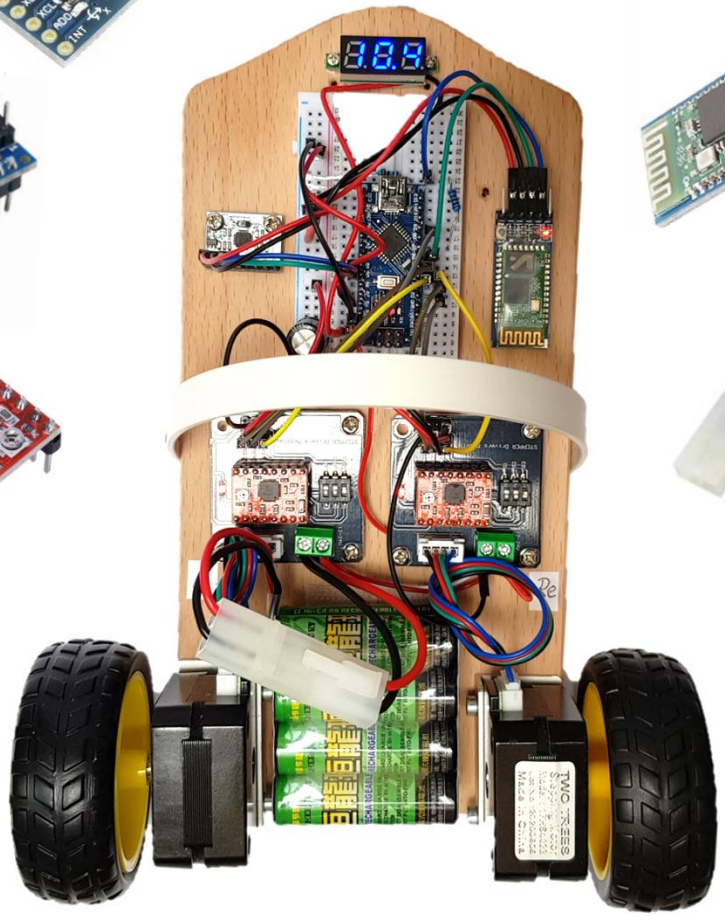
Arduino Nano



Bluetooth Modul
HC-05

Akkupack 9,6V

Schrittmotor-Treiber
A4988



Schrittmotoren

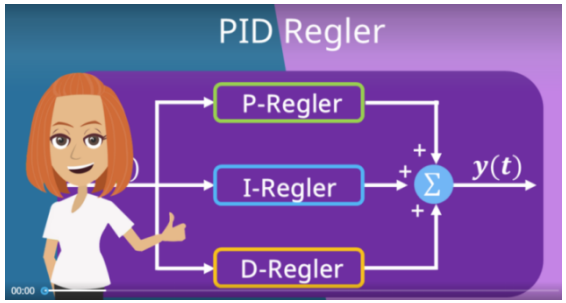
Radnaben Bohrung ausfeilen

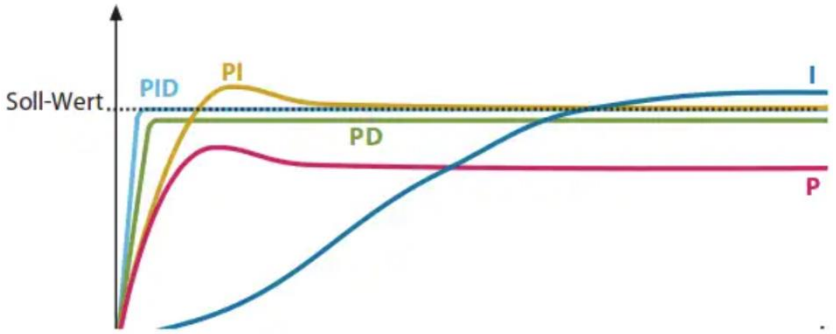
PID-Regler

- Der Segway wird mittels eines PID-Reglers in Balance gehalten.
- Er ist als Software im Sketch des Arduino hinterlegt, also ein digitaler Regler.
- Aller etwa 10ms (Variable `Loopzeit`) wird die Lage (Neigung) ausgewertet und der Regelalgorithmus ausgeführt.

- Siehe Sketch 88

- Zu PID-Regelung siehe:
- [Wie funktioniert ein Segway? | Elektro- und Informationstechnik an der Uni Magdeburg – YouTube]
- [https://studyflix.de/elektrotechnik/pid-regler-1450]



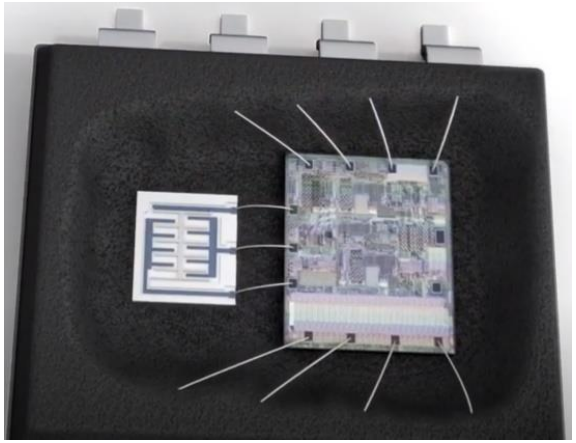


[Grundlagen der Regelungstechnik – YouTube]

[https://www.youtube.com/watch?v=fusr9eTceEo]

Lagemessung mit Accelerometer- und Gyro-Sensor

- Kernstück des Segway ist die Messung der Winkellage mit einem Lagesensor.
- Er misst die auftretenden Beschleunigungen in geradliniger Richtung (Accelerometer) als auch Änderungen der Winkellage (Gyrosensor).
- Aus beiden Werten zusammen kann die Lage (Neigung) des Segway berechnet werden.
- Die Änderung der Winkellage (der Neigung) wird über eine Kraft detektiert, welche aufgrund des Coriolis-Effekts auftritt. Zum Coriolis-Effekt siehe:
- https://studyflix.de/ingenieurwissenschaften/corioliskraft-berechnen-1915
- Die Beschleunigung als auch die Corioliskraft wird mit mikromechanischen Strukturen gemessen, bezeichnet auch als MEMS (Micro Electro Mechanical System)
- DE | Bosch Funktionsprinzip eines Beschleunigungssensors - YouTube
- DE | Bosch Funktionsprinzip eines Drehratensensors für ESP® - YouTube

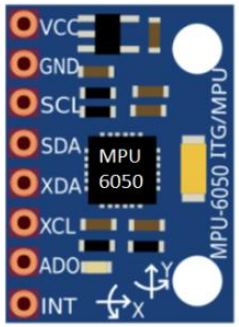


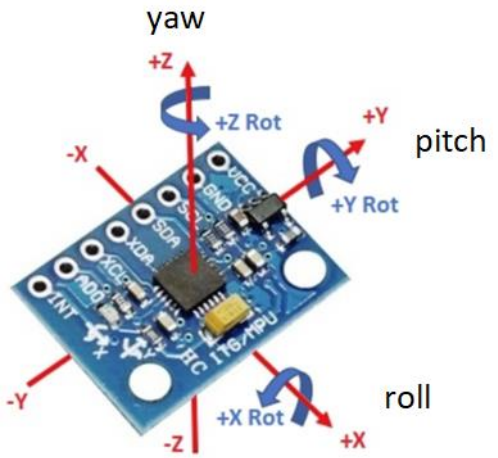
Modul GY-521 mit Lagesensor-Chip MPU-6050

- Der Sensor-Chip MPU-6050 kann die Winkellage (Neigung) in allen drei Achsen messen:
  Roll-, Nick- und Gierwinkel (Roll, Pitch and Yaw)
- https://de.wikipedia.org/wiki/Roll-Nick-Gier-Winkel

- Chip: MPU-6050
- Betriebsspannung: 3.3V bis 5V
- 16-Bit AD-Wandler
- Beschleunigungssensorbereich: ±2, ±4, ±8, ±16g
- Gyroscopebereich: ± 250, 500, 1000, 2000 °/s
- Befestigungslöcher Durchmesser: 3mm
- Adresse I2C: AD0 LOW =0x68 / AD0 HIGH =0x69

Board GY-521





- Der Chip misst Werte mit einer Genauigkeit von 16 Bit, verwendet intern jedoch 8-Bit-Register.
- Für exakte Messwerte muss man also zwei Register einlesen.
- Zusätzlich enthält der MPU6050 noch einen Temperatursensor, ebenfalls mit einer Genauigkeit von 16 Bit.
- Die Lagesensoren sind ab Werk kalibriert.
- Er kommuniziert mit dem Arduino über I2C Bus, der MPU als Slave, der Arduino als Master.
- Die Adresse des MPU am I2C Bus ist 0x68 (festgelegt in dessen Hardware).
- Das Board GY-521 verträgt auch VCC = 5V

Modul GY-521 mit Lagesensor-Chip MPU-6050

- Produktspezifikation des MPU6050:
- https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

- Beschreibung der Register des MPU6050:
- https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

InvenSense	MPU-6000/MPU-6050 Register Map and Descriptions	Document Number: RM-MPU-6000A-00 Revision: 4.2 Release Date: 08/19/2013

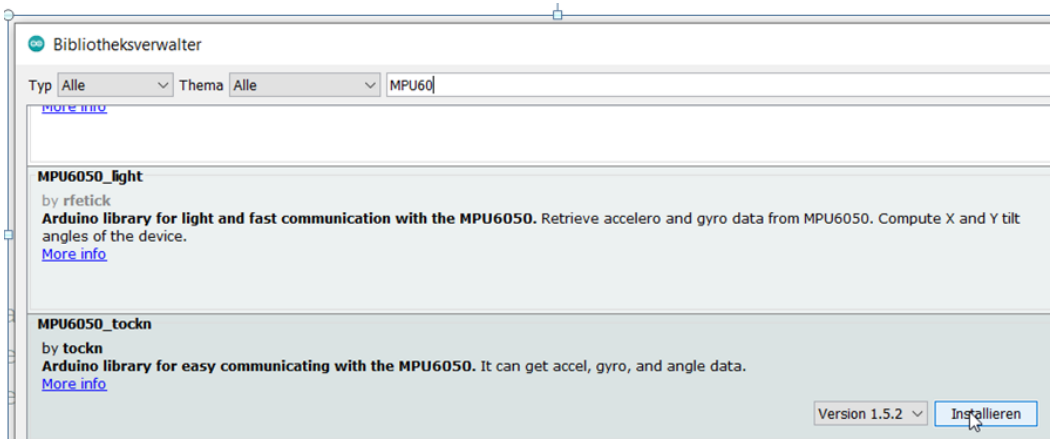
- Hier die wichtigen Register:

Register	Funktion
Beschleunigung	
0x3Bh	X-Achse High-Byte
0x3Ch	X-Achse Low-Byte
0x3Dh	Y-Achse High-Byte
0x3Eh	Y-Achse Low-Byte
0x3Fh	Z-Achse High-Byte
0x40h	Z-Achse Low-Byte
Temperatur	
0x41h	High-Byte
0x42h	Low-Byte
Lage	
0x43h	X-Achse High-Byte
0x44h	X-Achse Low-Byte
0x45h	Y-Achse High-Byte
0x46h	Y-Achse Low-Byte
0x47h	Z-Achse High-Byte
0x48h	Z-Achse Low-Byte

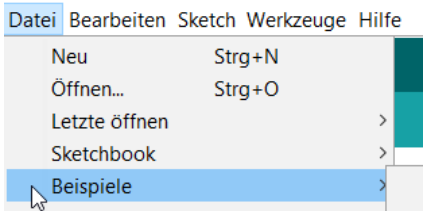
Bibliotheksprogramm MPU6050_tockn.h

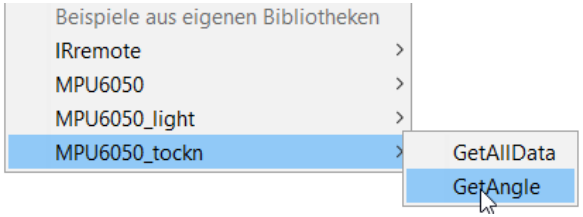
- Zur Ermittlung der Sensordaten im MPU-6050 und deren Übertragung per I2C-Bus zum Arduino gibt es ein Bibliotheksprogramm, das wir installieren: MPU6050_tockn

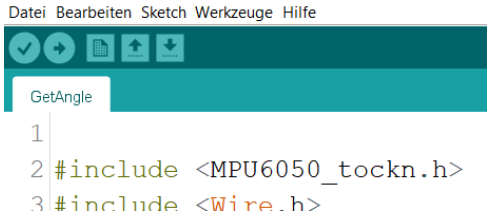
- MPU6050_tockn - Arduino Reference



- Einen Sketch zum Auslesen der Daten findet man:







- Siehe auch sketch_83_MPU6050_tockn_test

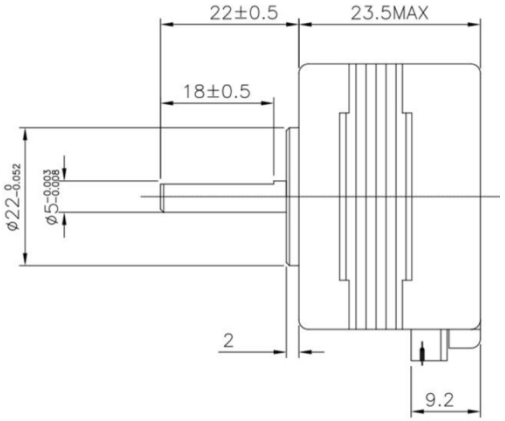
Schrittmotor 17HS4401S

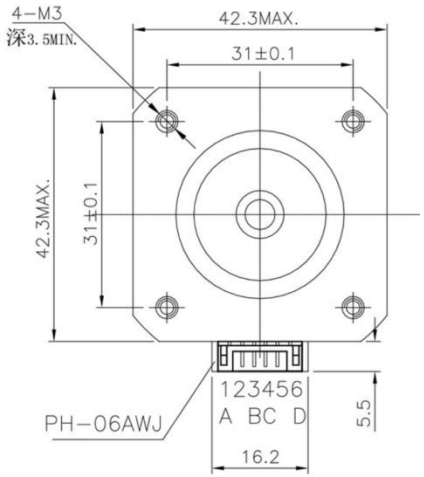
Twotrees NEMA 17 Schrittmotor für Titan-Extruder 3D-Druckermotor 17HS4023 mit Draht (Schwarz, 1Pcs)

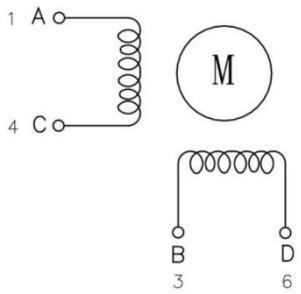
9,99 €
prime

Name:	Nema 17 Schrittmotor
Modell:	17HS4023
Schrittwinkel:	1,8 ° ± 0,09 °
Nennspannung:	DC 4,1 V
Nennstrom:	DC 1,0 A / Phase
Haltemoment:	≥ 13OmN. M
Maximale Leerlaufstartfrequenz:	≥ 1400 PPS
Gewicht:	132 g

Widerstand pro Phase: 4,3 Ohm





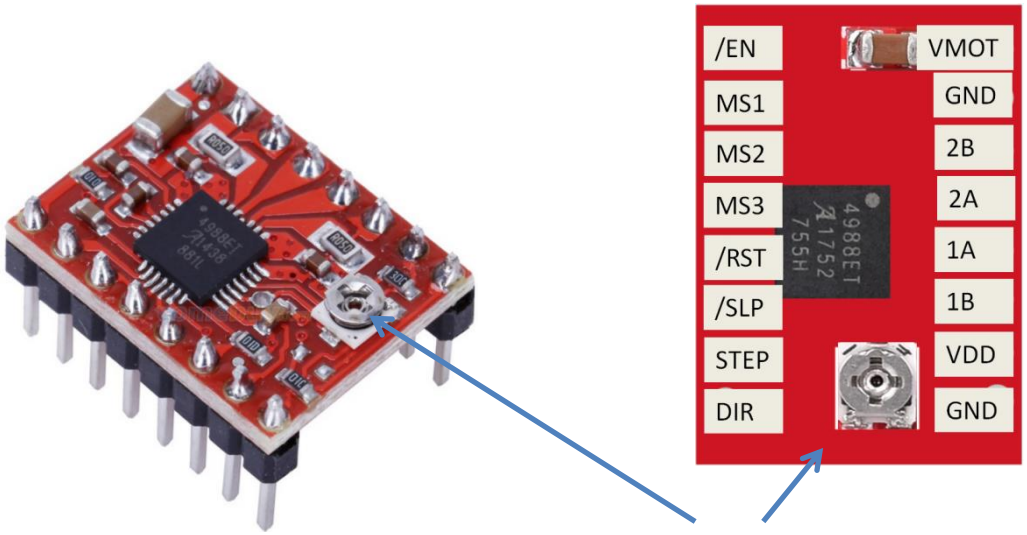


Schrittmotor-Treiber Modul mit A4988

Das Schrittmotor-Treiber-Modul (Stepper Motor Driver Carrier) Pololu A4988 ist ein Breakout Board für den Schrittmotor-Treiberschaltkreis A4988 von Allegro und ermöglicht die Steuerung eines bipolaren Schrittmotors.
Die zulässige Spannung beträgt 8,0 – 35V , d.h. die Akkuspannung darf 8,0 V nicht unterschreiten.
Der maximal zulässige Dauer-Ausgangsstrom pro Phase (pro Wicklung) ist 1,0 A .
Der Phasenstrom muß mit dem Potentiometer eingestellt werden.
Das geht nur bei eingeschaltetem Motor, denn der Strom wird über einen Meßwiderstand (Shunt) gemessen.

Dabei gilt laut Datenblatt: Phasenstrom = 2x VREF , z.B. 1 A = 0,5 V

Sicht auf Oberseite:



Einstellregler für Phasenstrom
Direkt am Regler kann die Spannung VREF gegen GND gemessen werden

Die tatsächlichen Werte können aber abweichen.

Den Phasenstrom so einstellen, daß die Funktion erfüllt wird , er aber so gering als möglich ist (Akku-Laufzeit).

Schrittmotor-Treiber Modul mit A4988

- Wir stellen 1/8 Schritt-Betrieb ein:

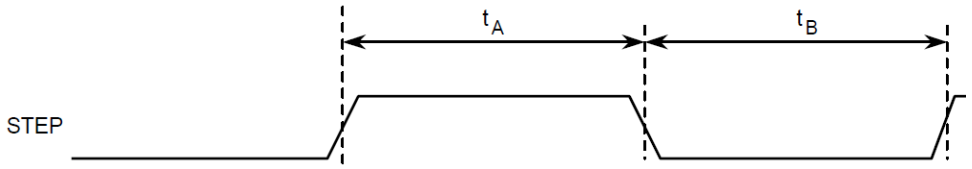
MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

- Eingänge:

  /RST Änderung auf LOW: Reset. Muß auf HIGH liegen (an +5V) oder offen bleiben, sonst keine Funktion.
  /ENABLE LOW: Treiber eingeschaltet HIGH oder frei: Treiber ausgeschaltet
  /SLP (nicht auf Breakout-Board kontaktiert) LOW: Treiber ausgeschaltet (Sleep-Mode)

- Siehe auch Sketch 87 und Sketch 88

- Erforderliche Mindestdauer eines Schrittimpuls-Signals am Eingang STEP (siehe Dokumentation des A4988):

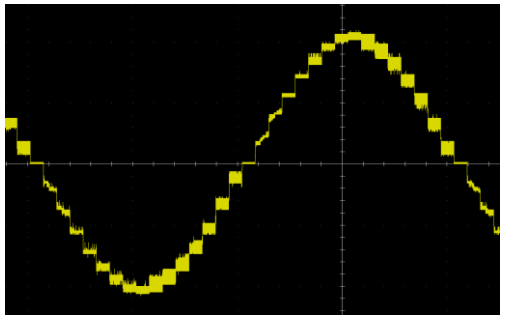


STEP minimum, HIGH pulse width	t_A	1	µs
STEP minimum, LOW pulse width	t_B	1	µs

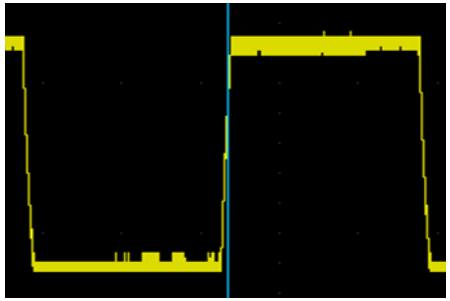
Schrittmotor-Treiber Modul mit A4988

- Messung des Stromverlaufs durch eine Motorwicklung mit einem Oszilloskop Hantek 6022BE
- (siehe Mechatronik Teil 1):
-
- Einfügen eines Meßwiderstandes („Shunt“), dargestellt in rot:
- Der Wert sollte deutlich kleiner sein als der Phasenwiderstand des Motors
- (1 Ohm, besser noch geringer), sonst wird der Motorstrom verfälscht.
- Bei Anschluß der Meßleitung zum Oszi muß das USB-Kabel zum Nano abgezogen sein, wegen Potentialunterschieden der GND-Leitungen.

- Stromverlauf bei 1/8 Schritt Betrieb:

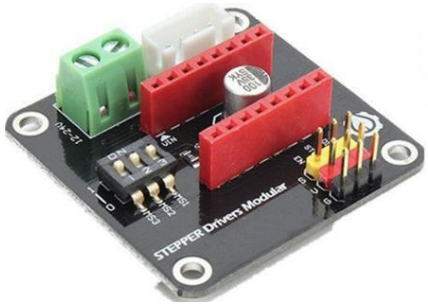


- Bei einer Einstellung zu Vollschritt (Full step)
- würde der Stromverlauf so aussehen:



Breakout Board für Modul mit A4988

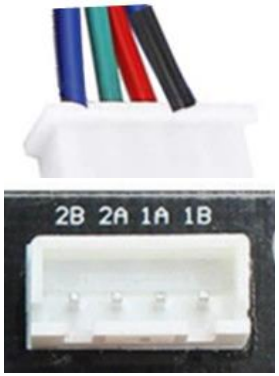
Wir verwenden ein zusätzliches Breakout Board für die Treiber-Module mit A4988.
Damit lassen sich sich die Anschlüsse zum Arduino Nano und vor allem zu den Schrittmotoren besser verbinden.



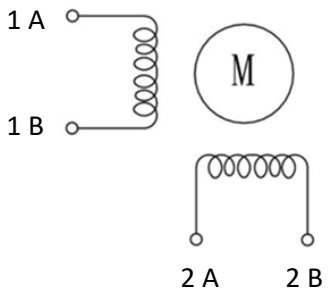
Wenn das Breakout Board verwendet wird, erfolgt die Einstellung für MS1...MS3 durch die Schiebeschalter:

Stecker am Board

Am originalen Stecker müssen Kabel blau und schwarz getauscht werden



Stecker am Motor



Bluetooth

- Bluetooth ist ein Standard für die Datenübertragung per Funk, im Frequenzbereich zwischen 2,40 GHz und 2,48 GHz.

- Nur über kurze Strecken
- Bei geringem Energieverbrauch
- Unkompliziert, funktioniert mit jedem Betriebssystem

- Für den Verbindungsvorgang gilt das erste Gerät als sogenannter „Meister“ („Master“), da von ihm das Netz aufgebaut wird. Im Folgenden muss man ein weiteres Gerät („Slave“) hinzufügen, das nach dem Signal des Meisters sucht. Danach baut sich die Verbindung zwischen den Geräten auf, meist mit vorheriger PIN-Abfrage.
- Diesen Vorgang nennt man auch „Pairing“. Bis zu acht Geräte können sich zeitgleich in einem Netzwerk verbinden.

- Bluetooth » einfach erklärt (conrad.de)

Bluetooth Modul HC-05

Mit den Modulen HC-05 und HC-06 kann der Arduino eine Bluetooth-Funkverbindung aufbauen.
Der Anschluß des HC-05 bzw. HC-06 zum Arduino erfolgt über eine serielle UART-Verbindung (RxD, TxD).
HC-05 kann als Master oder Slave, HC-06 dagegen nur als Slave fungieren.

Das Modul benötigt eine Konfigurierung (Programmierung von Einstellungen), dazu muß es in den sogenannten AT-Modus gebracht werden:

- den Reset-Taster am HC-05 gedrückt halten, während das USB-Kabel angeschlossen wird

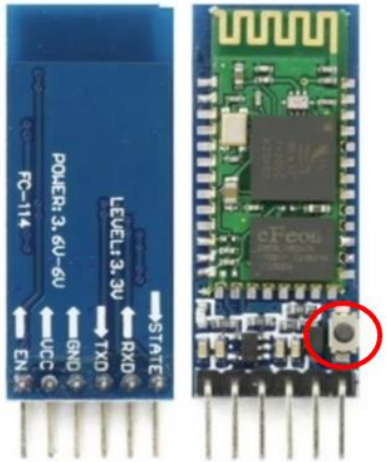
oder

- den Anschluss "EN" (auch als "Key" bezeichnet) an 3,3V legen (bevor das USB-Kabel angeschlossen wird)

Im AT-Mode blinkt die rote LED am Modul dann langsam aller zwei Sekunden.

Der Anschluss des HC-05 an die Pins RxD / TxD des Arduino stört aber das Hochladen des Sketches über die USB-Schnittstelle, es erscheint eine Fehlermeldung. Das Modul müßte während des Hochladens jedesmal abgetrennt werden.
Besser ist es, eine zusätzliche UART-Schnittstelle zu schaffen,
was hier an den Pins D6 und D7 geschieht.
Dazu existiert ein Bibliotheksprogramm "SoftwareSerial".

Bei der Konfigurierung muß die Baudrate zum HC-05 auf 38400 eingestellt sein.
Der Eingang RxD des HC-05 verträgt laut Datenblatt nur 3,3V ,
das Signal TxD vom Arduino sollte darum über einen Spannungsteiler geführt werden.



Reset

Sketche

Inhaltsverzeichnis

Zuordnung Pins am Arduino Nano

Nano	Funktion	Eing/Ausg		
A4	SDA	Eing+Ausg		I2C-Bus „Serial Data“ zum MPU-6050
A5	SCL	Eing+Ausg		I2C-Bus „Serial Clock“ zum MPU-6050
D2	STEP MotRechts	Ausgang		Schrittfrequenz Motor rechts
D3	DIR MotRechts	Ausgang		Drehrichtung Motor rechts
D9	STEP MotLinks	Ausgang		Schrittfrequenz Motor links
D5	DIR MotLinks	Ausgang		Drehrichtung Motor links
D6	Receive Data vom HC-05	Eingang		UART-Bus , von TxD des HC-05
D7	Transmit Data zum HC-05	Ausgang		UART-Bus , an RxD des HC-05
D4	/EN	Ausgang		/Enable für beide Motoren (aktiviert bei LOW)

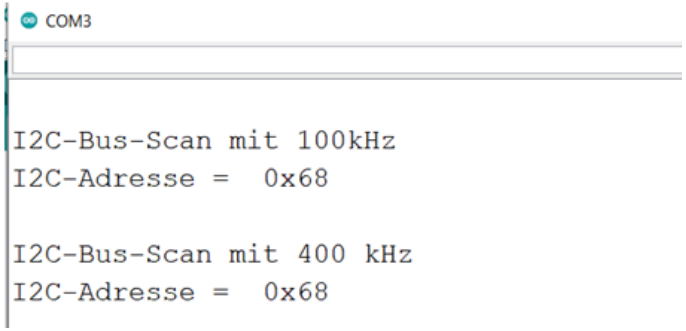
Beachte: Je nach Ausfuhrung des Nano bei „Werkzeuge“ evtl "ATmega328P (Old Bootloader)" einstellen.

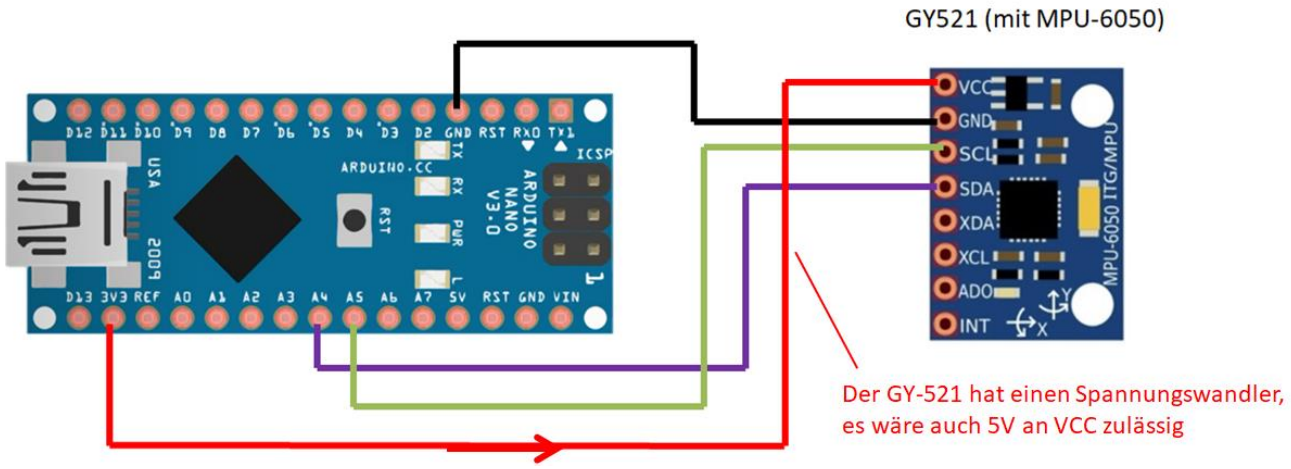
Sketch 81 I2C Bus Scan

Der I2C Bus wird gescannt, nacheinander mit zwei Bus-Takten: 100kHz und 400kHz.

Dabei werden alle möglichen Adressen abgefragt: 0 ... 128 (dezimal)

Da nur der MPU6050 angeschlossen ist , erscheint lediglich seine Geräte-Adresse (hexadezimal 68) im Serial Monitor:



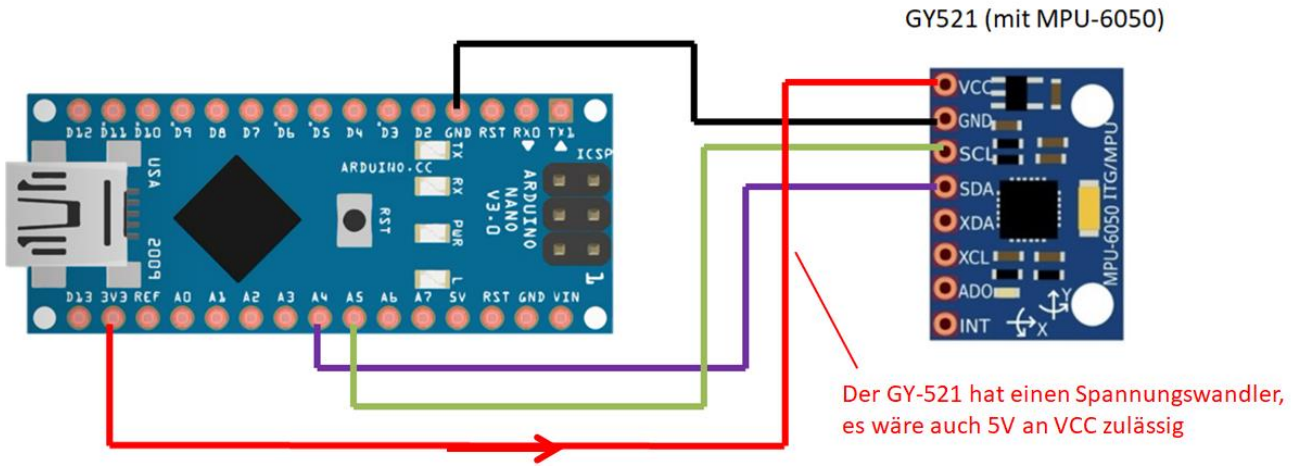


Sketch 82 MPU6050 Daten auslesen

Es wird der MPU-6050 ausgelesen und die Daten werden im Serial Monitor angezeigt.

Neben den Accelerometer-Werten und den Gyrosensor-Werten wird auch die Temperatur ausgegeben.

```
COM3
AcX = 476 | AcY = 56 | AcZ = -17744 | Tmp = 29.42 | GyX = -139 | GyY = 15 | GyZ = 100
AcX = 448 | AcY = -12 | AcZ = -17908 | Tmp = 29.42 | GyX = -141 | GyY = 23 | GyZ = 100
AcX = 356 | AcY = 28 | AcZ = -17900 | Tmp = 29.47 | GyX = -146 | GyY = 24 | GyZ = 112
AcX = 476 | AcY = -28 | AcZ = -17892 | Tmp = 29.61 | GyX = -164 | GyY = -13 | GyZ = 94
AcX = 356 | AcY = 48 | AcZ = -17892 | Tmp = 29.47 | GyX = -137 | GyY = 20 | GyZ = 103
AcX = 428 | AcY = 40 | AcZ = -18032 | Tmp = 29.66 | GyX = -172 | GyY = 5 | GyZ = 95
```



Sketch 83 MPU6050_tockn

Das Bibliotheksprogramm „MPU6050-tockn“ (siehe Seite 8) wird im Sketch 87 und 88 verwendet.

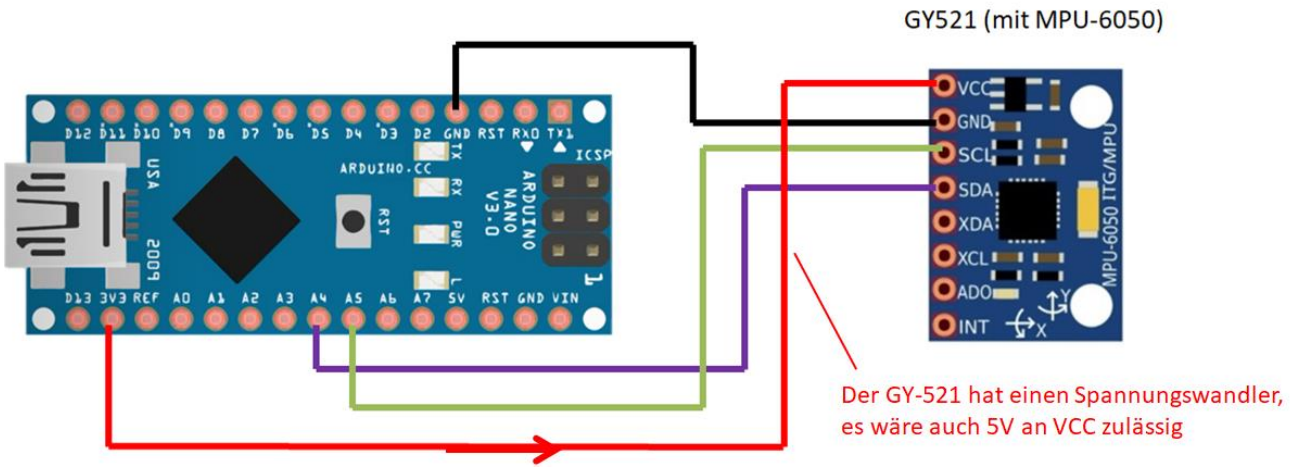
Hier wird damit nur die Funktion des MPU6050 geprüft und der Winkel ermittelt, bei dem der Segway in Balance ist.

Dieser Winkel wird in die Variable BALANCEWINKEL Im Sketch 87 und 88 eingetragen.

Für den konkreten Aufbau beträgt er etwa -106°, sollte aber für jeden Segway geprüft werden.

Dieser Sketch kann auch aufgerufen werden mit: Datei/Beispiele/MPU6050_tockn/GetAngle

```
========================================
Calculating gyro offsets
DO NOT MOVE MPU6050...
Done!
X : -2.10
Y : 0.35
Z : -0.34
Program will start after 3 seconds
========================================Winkel X: -1.08
Winkel X: -1.07    Winkel Y: -108.16    Winkel Z : -0.89
Winkel X: -1.04    Winkel Y: -108.15    Winkel Z : -0.89
```



Sketch 84 Schrittmotoren Test

Mit diesem Sketch wird die Funktion der Schrittmotortreiber A4988 und der Schrittmotoren geprüft.
Obwohl für diesen Sketch nicht alle Module benötigt werden, wird die komplette Hardware des Segway verwendet (siehe Sketch 86).

Am A4988 ist Betrieb mit 1/8 Schritt eingestellt:

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Halft Step
LOW	HIGH	LOW	Quarter Step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth Step

Wenn das Breakout Board verwendet wird, erfolgt die Einstellung für MS1...MS3 durch die Schiebeschalter:

Im Sketch:

```
int MotLinksGeschwind = 200; // bei 200: 500000/200 = 2500 Mikrosek
// zwischen zwei 1/8 Schritten a 0,225°, das sind 0,25 U/Sek.
```

Das Potentiometer am Board A4988 wird so eingestellt, daß der Phasenstrom (Strom in jeder Motorwicklung) einerseits für die Funktion ausreicht, andererseits so gering als möglich ist (Akku-Laufzeit).

Sketch 85 Bluetooth Modul HC-05 konfigurieren

USB-Kabel an Arduino Nano anschliessen – rote LED auf HC-05 blinkt schnell.
Sketch 85 hochladen, rote LED auf HC-05 blinkt weiterhin schnell.

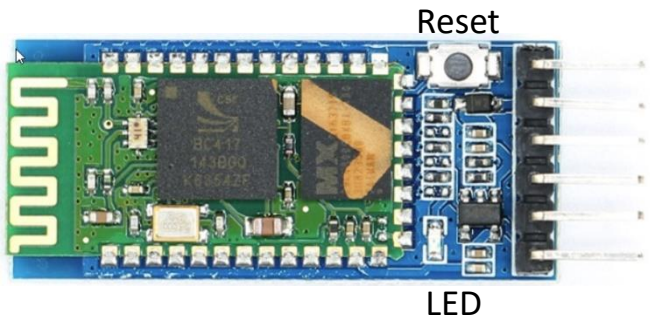
Danach in den sogenannten „**AT-Modus**“ wechseln (siehe auch im Tutorial):

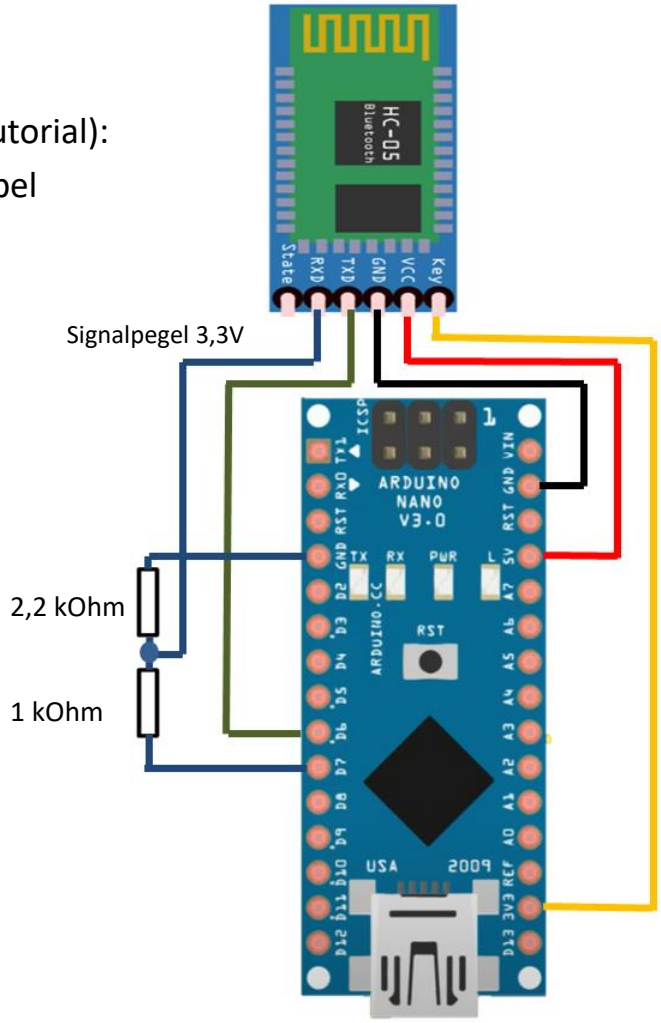
- den Reset-Taster am HC-05 gedrückt halten, während das USB-Kabel angeschlossen wird,

oder

- den Anschluss "EN" (auch als "Key" bezeichnet) an 3,3V legen bevor das USB-Kabel angeschlossen wird

Im AT-Mode blinkt die rote LED dann langsam aller zwei Sekunden.



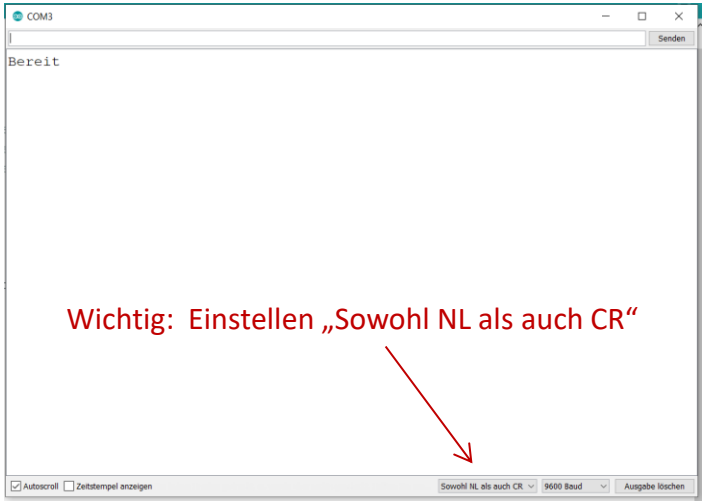


Eine Übersicht zu AT-Befehlen:

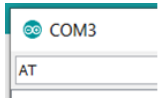
[Technik & Software erklärt - Bluetooth Modul HC-05 (google.com)]

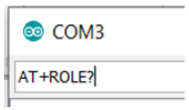
Sketch 85 Bluetooth Modul HC-05 konfigurieren

Ausgabe im Serial Monitor nach Start des Sketches:

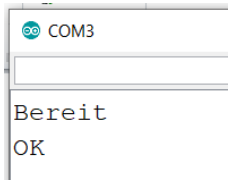


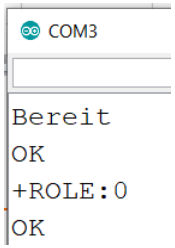
Eingabe und Senden:





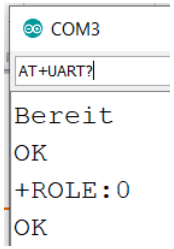
Daraufhin Anzeige:

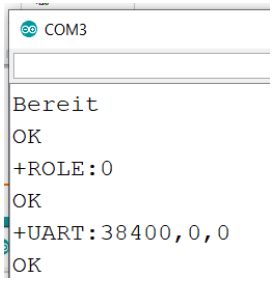




ROLE muß auf „0“ stehen, das ist werksseitig auch so eingestellt.
HC-05 ist Slave, das Smartphone ist Master.
Sonst umstellen mit Eingabe: AT+ROLE=0

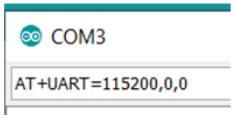
Abfrage, ob Baudrate 38400 eingestellt ist:

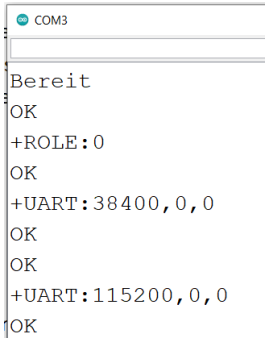




Sketch 85 Bluetooth Modul HC-05 konfigurieren

Bei der Steuerung des Segway ist eine schnelle Reaktionszeit wichtig
Deshalb erfolgt eine Erhöhung der Bluetooth-Übertragungsgeschwindigkeit, Baudrate von 38400 auf 115200.
Dazu weiter im AT-Modus:

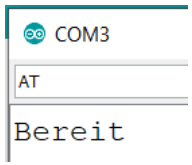


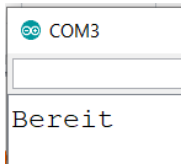


Nun wieder in den Normal-Modus gehen (schnelles Blinken).

Sketch wieder in Arduino hochladen
Jetzt die Kommunikation mit dem Smartphone starten

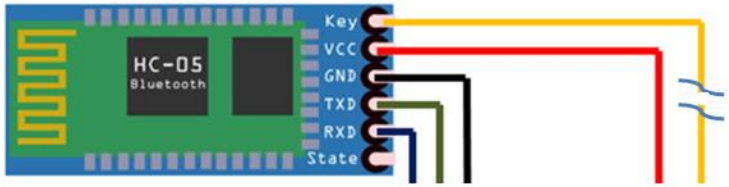
Für Änderungen im AT-Mode muß die Baudrate im Sketch wieder auf 38400 gestellt werden.





Sketch 85 Bluetooth Modul HC-05 konfigurieren

Nach erfolgter Konfigurierung im AT-Modus wird die Verbindung zum Key (EN) wieder getrennt
Bzw. Spannungsversorgung kurz ab- und wieder zugeschaltet (USB-Kabel kurz abziehen).



Jetzt ist der HC-05 wieder im Normal-Modus. Die LED am Modul blinkt jetzt schnell.

Im Normal-Modus sollte die Übertragungsgeschwindigkeit höher sein, für eine möglichst kurze Loop-Durchlaufzeit der Sketche. Im Sketch:

```
20  HC_Serial.begin(115200);
```

Nun kann die Verbindung zu einem Smartphone hergestellt werden.
Wenn die Verbindung steht, blinkt die LED zweimal kurz mit anschließender langer Pause.
Siehe Sketch 86.

The led light indicates the Bluetooth connection status.
Three modes:

- Fast flashing: no Bluetooth connection.
- Slow flashing: entering the AT mode.
- Double flash: that Bluetooth is connected and the port is open.

Sketch 86 Bluetooth Kommunikation HC05 - Smartphone

Der Segway wird per Bluetooth-Verbindung von einem Android-Smartphone gesteuert.

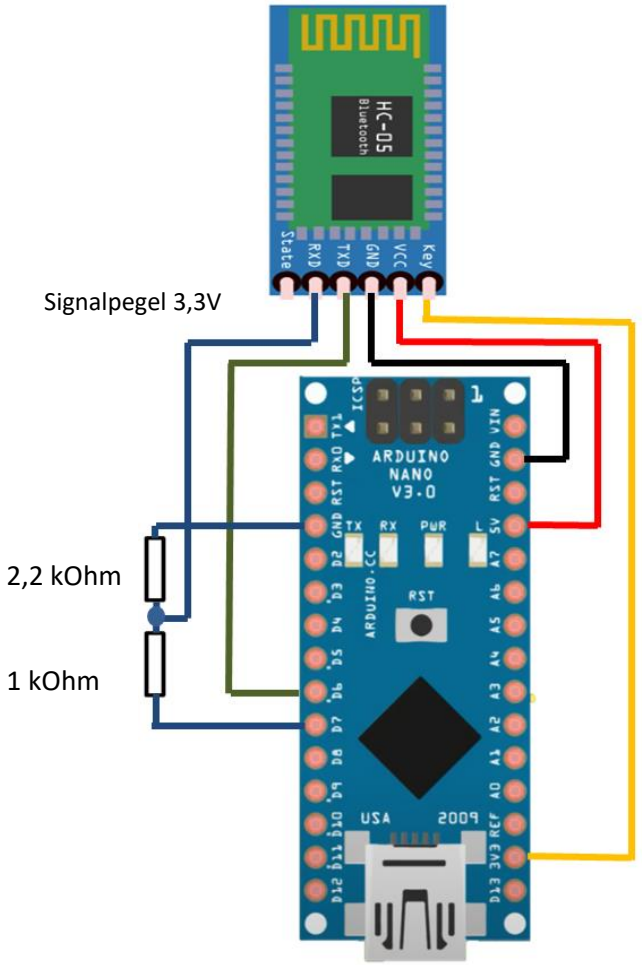
Dazu wird folgende App installiert:

Bluetooth Terminal HC-05

mightyIT Kommunikation

Enthält Werbung · Bietet In-App-Käufe an

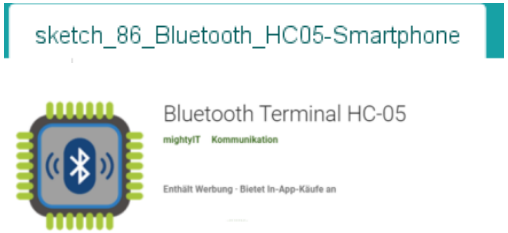
Hardware wie im Sketch 85:



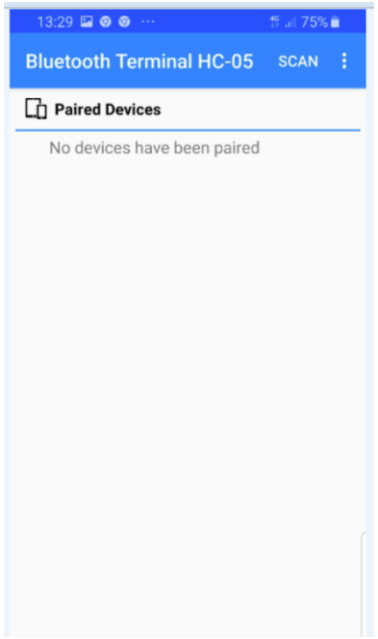
Sketch 86 Bluetooth Kommunikation HC05 - Smartphone

Start des Sketches 86.
Die LED auf dem HC-05 blinkt schnell.

Start der App im Smartphone.

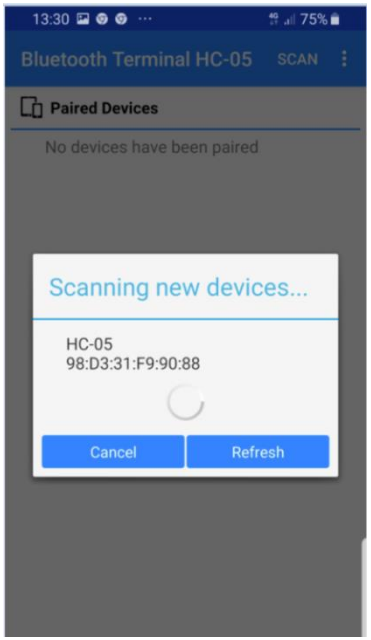


Nach Start erscheint:

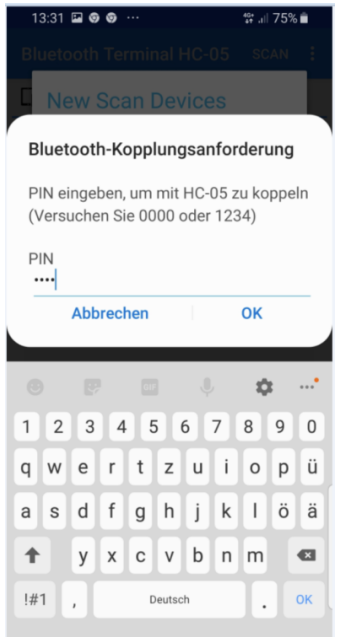


Drücke „Scan“.
Wenn Scan abgeschlossen,

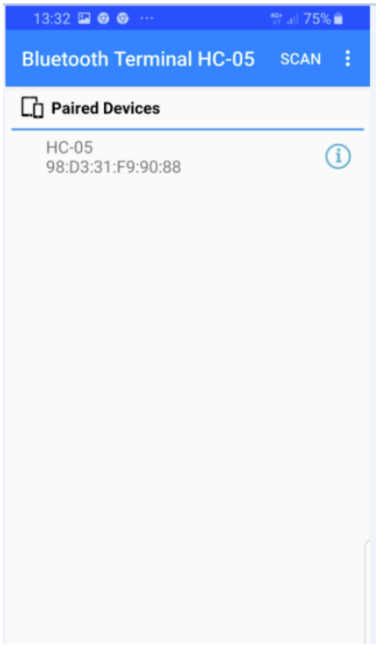
drücke „HC-05“



Eingabe der Pin „1234“:



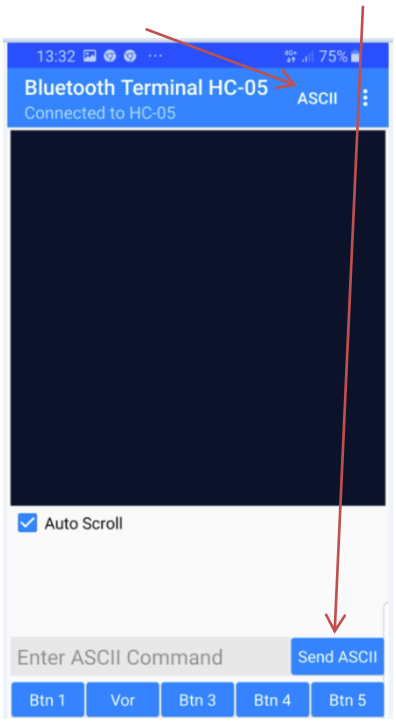
Jetzt steht die Verbindung.
Die LED auf dem HC-05 blinkt
2x kurz mit langer Pause.



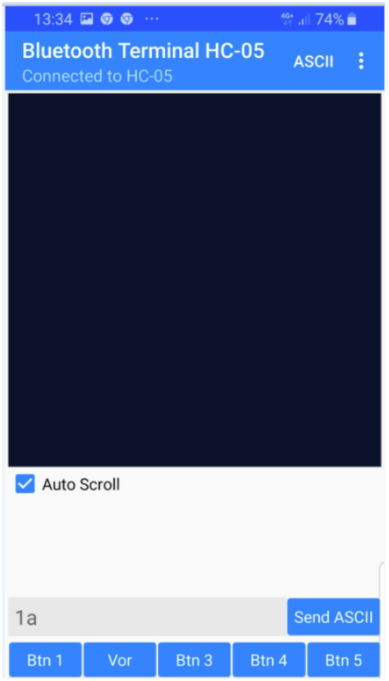
Sketch 86 Bluetooth Kommunikation HC05 - Smartphone

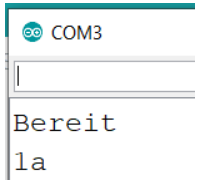
Beachte: Die Baudrate im Sketch muß mit der im AT-Modus eingestellten Baudrate übereinstimmen ! HC_Serial.begin(.....)

Die Kommunikation soll für ASCII-Zeichen eingestellt werden:

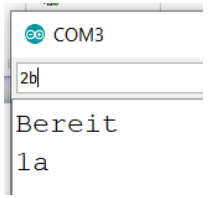


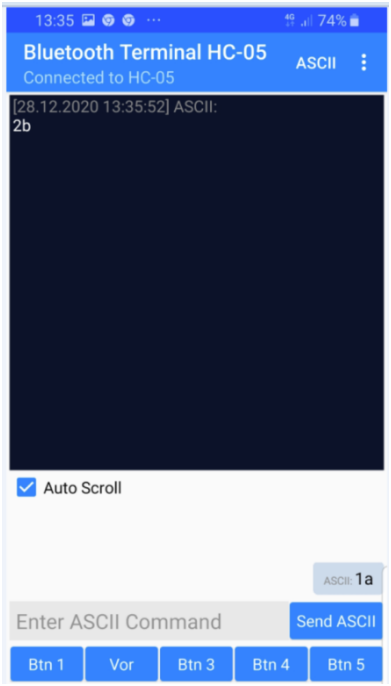
Am Smartphone eingeben und senden von „1a“:





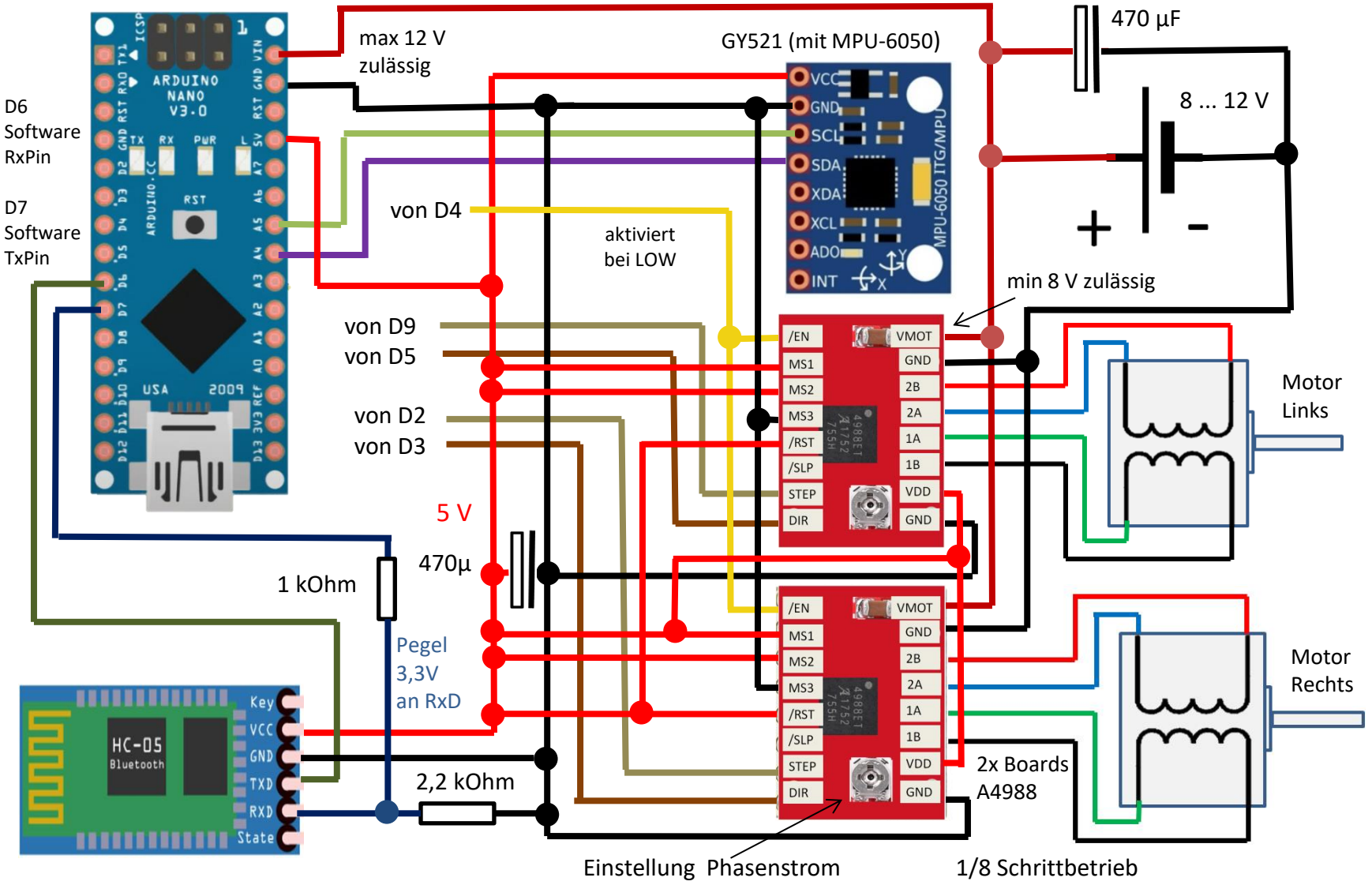
Am Serial Monitor eingeben und senden von „2b“:





Sketch 87 + 88 Hardware

D7 und D8 nicht verwendbar, werden von tockn.h auf HIGH gezogen.
MS1 ...MS3 siehe auch Breakout Board Schiebeschalter



Sketch 87 Segway Balance Test

Es wird getestet, ob sich der Segway in Balance hält.
Akku ausreichend geladen ? Mindestens 8V notwendig für die A4988, bei Betrieb der Motoren.
Noch keine Fahrbefehle über Bluetooth vom Smartphone (siehe Sketch 88).
Der Segway wird etwa in Balance-Lage gehalten.
Nach Hochladen kommt Ausschrift am SerialMonitor:
(wenn Segway in dieser Lage gehalten wird)

```
=====================================
Calculating gyro offsets
DO NOT MOVE MPU6050...
Done!
X : -2.10
Y : 0.35
Z : -0.34
Program will start after 3 seconds
=====================================Winkel X: -1.08
Winkel X: -1.07    Winkel Y: -108.16    Winkel Z : -0.89
Winkel X: -1.04    Winkel Y: -108.15    Winkel Z : -0.89
```

Der Balancewinkel ist abhängig von der Anordnung des MPU-6050, wir befestigen das Modul so, dass der Winkel Y relevant ist.
Er kann mit dem sketch_83_MPU6050_tockn_test ermittelt werden.
In die Variable BALANCEWINKEL muss möglichst genau dieser Wert, sonst bewegt sich der Segway in eine Richtung auch ohne Fahrbefehl.

Takt des I2C-Bus erhöhen von standardmaessig 100kHz auf 400kHz, mit Wire.setClock(400000).

Das Schrittmotor-Treiber-Modul mit A4988 ist auf 1/8 Schritt-Betrieb eingestellt:
Eingänge MS1 und MS2 an +5V sowie MS3 an GND. Eingang /RST an +5V.
Diese Verbindungen sollten geprueft werden, sonst unerklärliche Fehlfunktion.

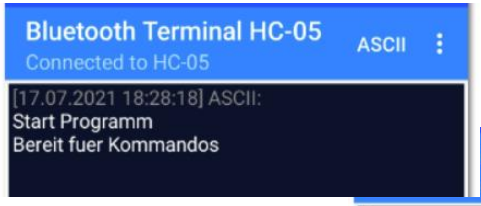
Am Potentiometer Einstellung Phasenstrom so, dass Funktion erfüllt wird.

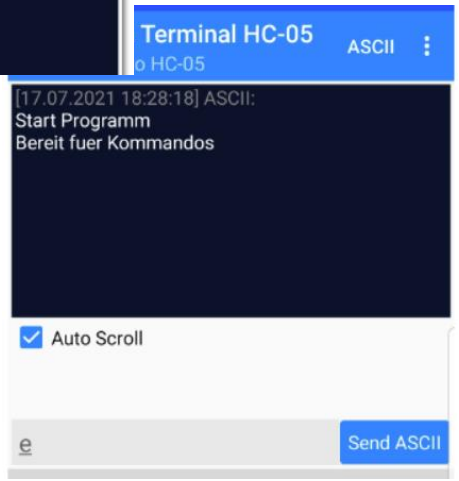
Sketch 88 Segway Fahren

Die vorhergehenden Sketche dienen zur Vorbereitung bzw. zur Prüfung der Funktion der einzelnen Komponenten.
Mit dem Sketch 88 wird nun der Segway in Balance gehalten als auch gefahren.

- Hochladen des Sketches vom PC
- Abziehen des USB Kabels (jetzt Spannungsversorgung des Nano über Batterie)
- Starten der App „Bluetooth Terminal HC-05“ im Smartphone
- Reset am Nano
- „Bereit“ kommt erst nach einigen Sekunden (Kalk Gyro Offsets, siehe Sketch 87 oben)
- Halten des Segway etwa im Balancewinkel

Das Smartphone verbindet sich über Bluetooth mit dem HC-05, es erscheint folgende Meldung:





Die Schrittmotoren sind vorerst deaktiviert
(Eingang /EN des A4988 auf HIGH gesetzt).

Nach Eingabe „e“ werden die Motoren aktiviert, der Segway kann losgelassen werden und hält die Balance.

Weitere Kommandos siehe nächstes Seite bzw. Beschreibung im Sketch 86.

Sketch 88 Kommandos vom Smartphone

Alle Kommandos Kleinbuchstaben.

e Motoren einschalten

Eine Eingabe von ' e ' aktiviert die Motoren (Beendigung von Sleep)

a Motoren ausschalten

Eine Eingabe von ' a ' deaktiviert die Motoren (in Sleep)

v Fahren vorwärts

Jede erneute Eingabe von ' v ' erhöht die Neigung nach vorn weiter um den Wert, der in der Variablen NEIGUNG steht

h Fahren nach hinten

Jede erneute Eingabe von ' h ' erhöht die Neigung nach hinten weiter um den Wert, der in der Variablen NEIGUNG steht

s Stop

Eine Eingabe von ' s ' stoppt die Weiterfahrt

l Links drehen

Eine Eingabe von ' l ' startet eine Richtungsänderung (Linksdrehung)

r Rechts drehen

Eine Eingabe von ' r ' startet eine Richtungsänderung (Rechtsdrehung)

o Drehen beenden

Eine Eingabe von ' o ' beendet jede Richtungsänderung

Sketch 88 Segway Fahren - Hinweise

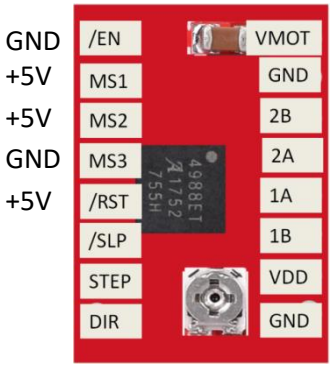
Akku ausreichend geladen (siehe Voltmeter Anzeige am Segway) ?

Koppeln HC-05 mit Smartphone: Siehe App „Bluetooth Terminal HC-05“.

Der GY-521 mit dem MPU-6050 ist im Segway so angeordnet wird, dass der Balancewinkel (der Segway ist ausbalanciert) in AchseY etwa -107° ist.
Bei Änderungen am Aufbau muß der Balancewinkel wieder neu ausgetestet werden.

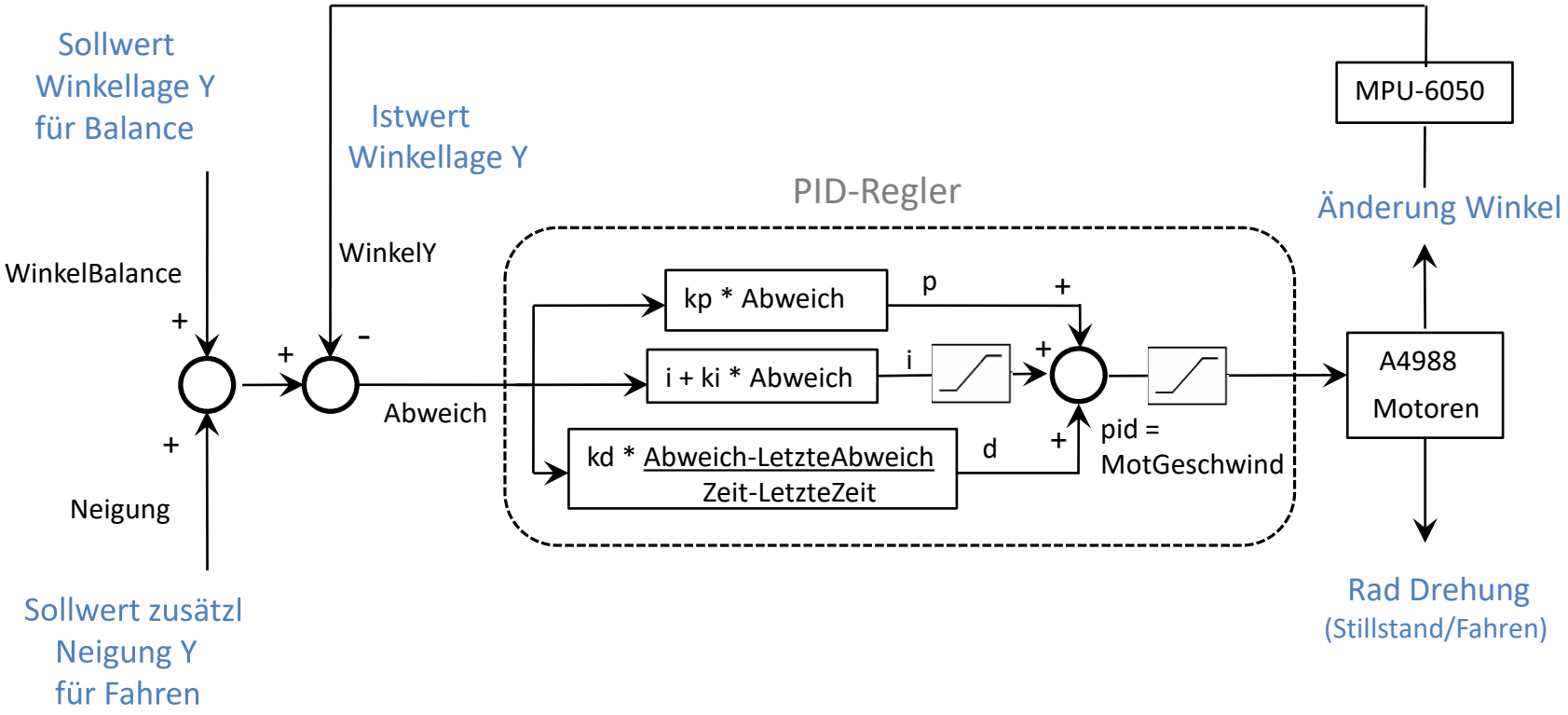
In die Variable BALANCEWINKEL muss möglichst genau der Balancewinkel sonst bewegt sich der Segway in eine Richtung auch ohne Fahrbefehl.

Das Schrittmotor-Treiber-Modul ist auf 1/8 Schritt-Betrieb eingestellt:
Eingänge MS1 und MS2 an +5V sowie MS3 an GND.
Eingang /RST an +5V.
Wenn /EN auf LOW geht → Segway fährt
Die Verbindungen sollten geprüft werden, zur Vermeidung von Fehlfunktionen.



Am Potentiometer den Phasenstrom einstellen, daß die Funktion erfüllt wird , der Strom aber so gering als möglich ist (Akku-Laufzeit).

Sketch 88 Regelung - Struktur



Sollwertführung siehe - siehe nächste Seite.

Begrenzung („anti-windup“)

Sketch 88 Regelung - Sollwertführung

Die Sollwerte für „WinkelBalance“ und „Neigung“ werden nicht abrupt geändert, sondern schrittweise, bei jedem Durchlauf einer Programmschleife (d.h. aller 10ms).

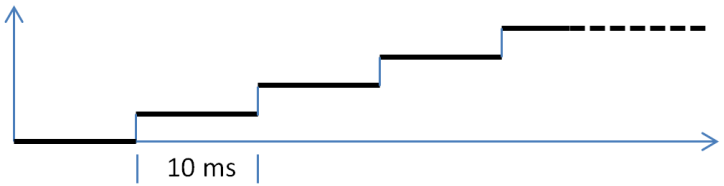
Im Sketch siehe Programmteil nach

```
/* Fuehrungsfunktionen fuer Regler-Sollwertvorgaben: */
```

Der Sollwert für „WinkelBalance“ wird zum Start vorerst dem tatsächlichen „WinkelY“ gleichgesetzt und dann schrittweise dem Wert angepasst, bei dem der Segway in Balance steht („BALANCEWINKEL“).

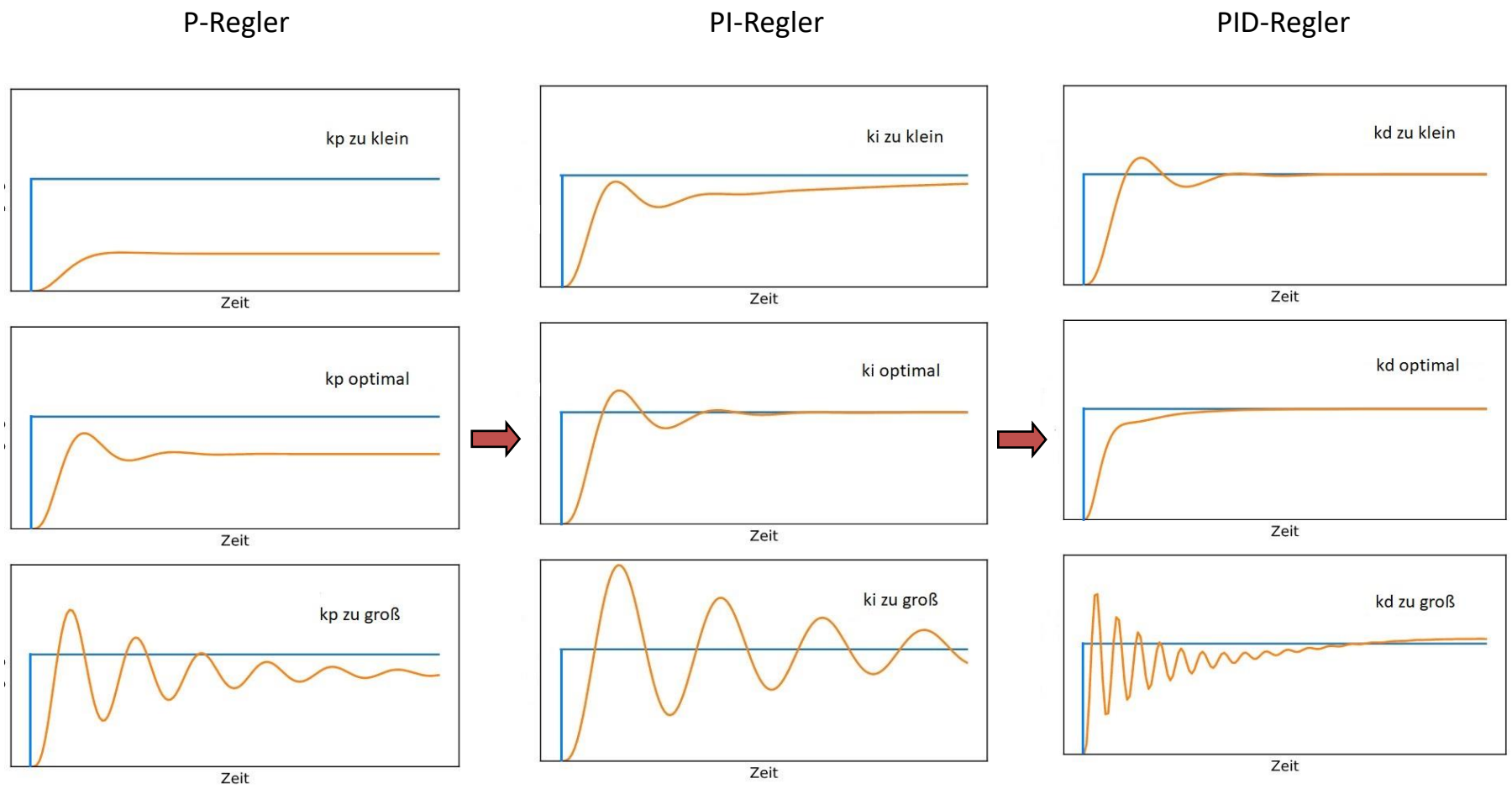
Damit wird verhindert, dass die Regelabweichung sofort sehr groß wird und die Räder durchdrehen, falls der Segway am Start zu schief gehalten wird.

Für ein sanfteres Beschleunigen und eine geringere Anregung des Reglers wird der Sollwert für „Neigung“ schrittweise erhöht, bis er den Endwert „BeschleunNeigung“ erreicht.



Danach wird er schrittweise reduziert auf den geringeren Wert „Fahrneigung“.
Das gleiche gilt beim Reduzieren auf Null nach Kommando „s“ für Stop.

Sketch 87 + 88 Einstellung des PID-Reglers



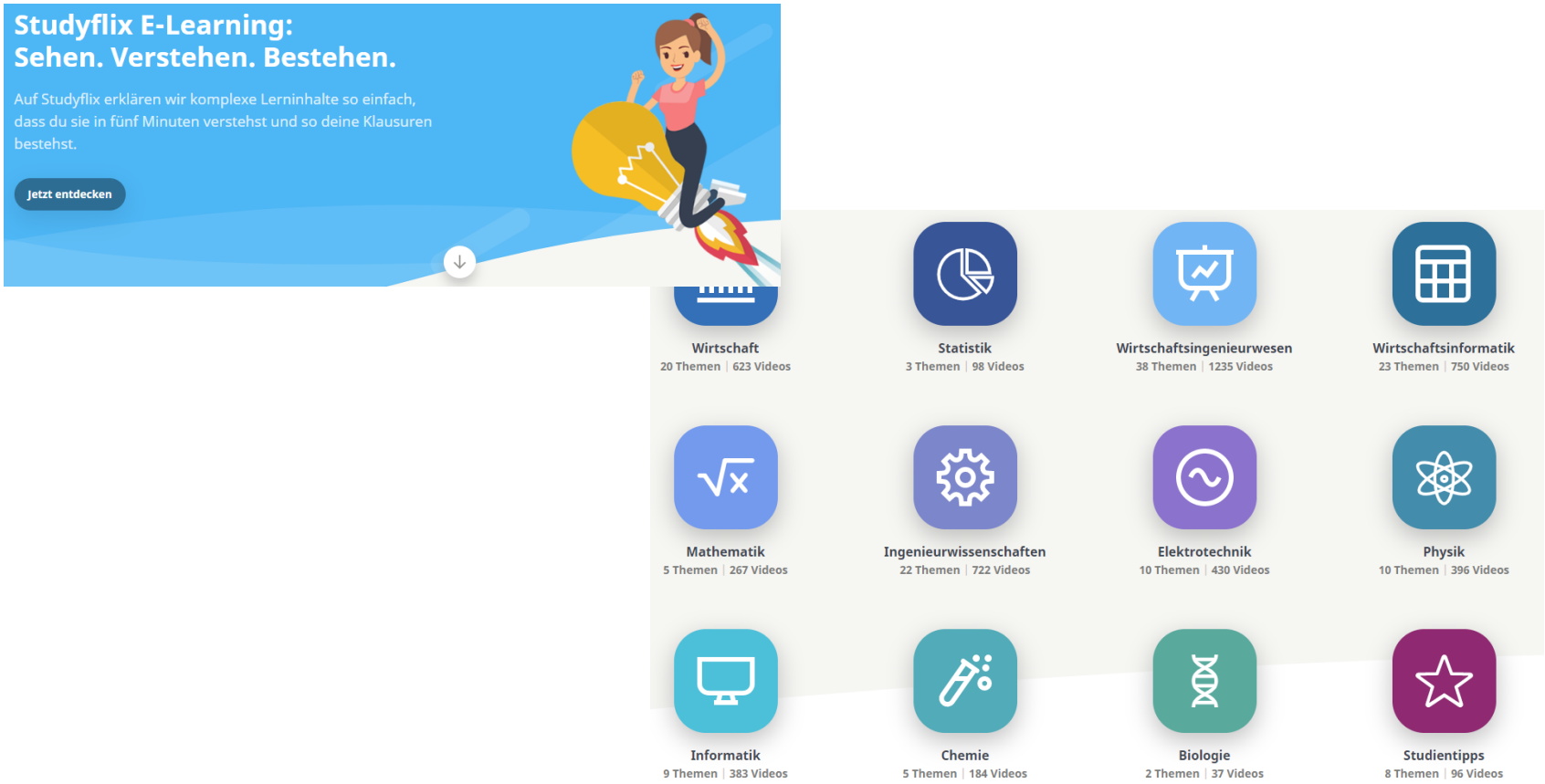
Zuerst nur den P-Anteil einstellen (ki und kd zu null) , danach den I-Anteil erhöhen.
ZumSchluß kann noch ein D-Anteil zugefügt werden.

Anhang

Inhaltsverzeichnis

Studyflix

https://www.youtube.com/channel/UC-vOkM5UgF3uqh2lpBcIu9g



https://studyflix.de/elektrotechnik/thema/regelungstechnik-25

ASCII-Code, ANSI-Code und UNICODE

ASCII (American Standard Code for Information Interchange) ist ein 7-Bit-Code und somit sind 128 Zeichen definiert.

Die Kodierung besteht aus 33 nicht druckbaren und 95 druckbaren **Zeichen** und umfasst sowohl Buchstaben, Interpunktionszeichen und Ziffern als auch Steuerzeichen.

ASCII Code / ASCII Tabelle - Verständliche Erklärung auf Deutsch – YouTube

ASCII-Zeichentabelle, hexadezimale Nummerierung

Code	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	…C	…D	…E	…F
0…	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1…	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2…	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5…	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6…	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7…	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

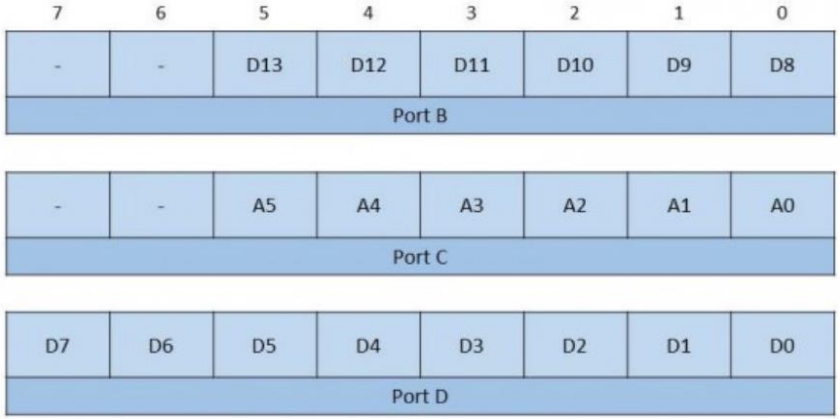
ANSI ist eine Erweiterung auf 8 Bit , für die Codierung von 256 Zeichen (z.B auch Ü, ä, ß).

UNICODE ist eine Erweiterung für alle relevanten Weltsprachen, aber auch Emojis u.a.

ASCII und UNICODE einfach erklärt! – YouTube

Portmanipulation

Der Atmega328P-Mikrocontroller des Arduino Uno oder Nano besitzt 3 Ports: Port B, C und D.
Hier die Zuordnung der Binär-Pins D0 bis D13 und der Anlog-Pins A0 bis A5 zu den Ports:



Selektives Setzen von Bits

Um nun ein einzelnes oder mehrere Bits *selektiv* zu setzen, verwendet Ihr das logische ODER. Folgendermaßen könnt ihr zum Beispiel PB5 auf HIGH zu setzen, ohne die restlichen Pins des PORTB zu beeinflussen:

`PORTB = PORTB | 0b100000` bzw. `PORTB |= 0b100000` oder, bevorzugt:

```
PORTB |= (1<<PB5)
```

PB5 ist über ein #define schlicht als 5 definiert. Man könnte also genauso gut `PORTB |= (1<<5)`

Mehr Infos: https://arduino-projekte.webnode.at/registerprogrammierung/ein-ausgangsports/

Bitoperationen

Die logischen Bitoperatoren sind:

& logisches UND

| logisches ODER „Pipe“ , mit Tastatureingabe „ AltGr “ + „ < “

^ exklusives ODER (XOR)

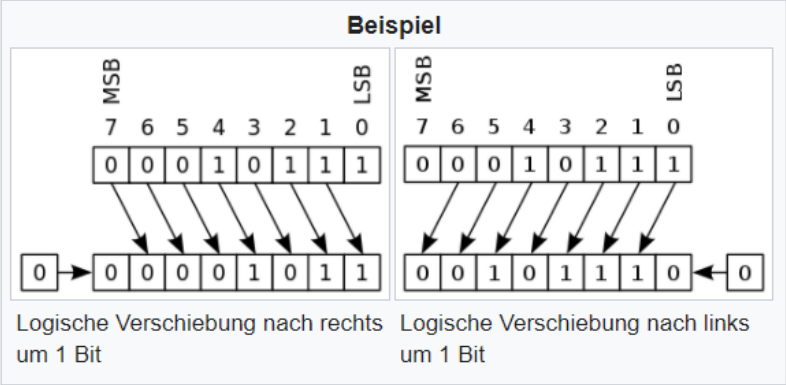
~ Negation (NICHT)

Ein wesentlicher Unterschied zu den gewohnten Operatoren wie z.B. Plus oder Minus ist, dass die Bitoperatoren bitweise angewendet werden.

Die Shiftoperatoren sind:

>> Rechts-Shift

<< Links-Shift

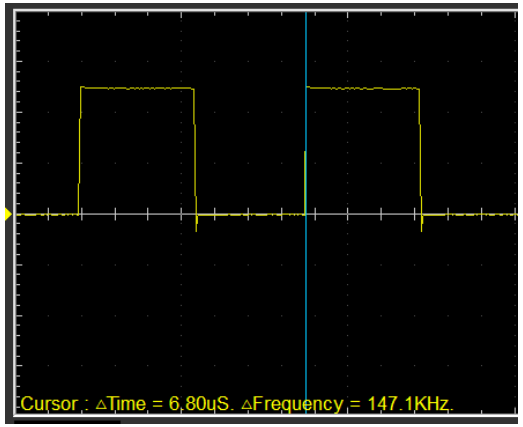


Laufzeit Loop Sketche - Arduino Nano

Minimal-Sketch: Der Loop wird innerhalb 6,8 Mikrosekunden durchlaufen

TestZeit_1 §

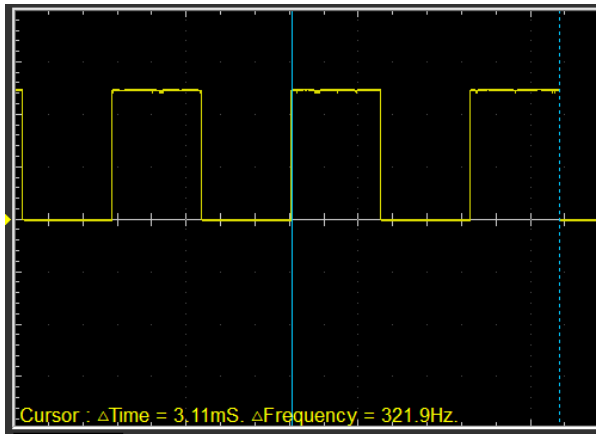
```
void setup(){
pinMode(2, OUTPUT);
}
void loop(){
digitalWrite (2,LOW);
digitalWrite (2,HIGH);
}
```



Mit zusätzlicher Ausgabe auf dem seriellen Monitor verlängert sich die Laufzeit erheblich auf 171 Mikrosekunden wenn die Baudrate auf 115200 eingestellt ist (bei 9600 noch erheblich länger).

TestZeit_3

```
void setup()
{
pinMode(2, OUTPUT);
Serial.begin(115200);
}
void loop()
{
digitalWrite (2,LOW);
Serial.write("A");
digitalWrite (2,HIGH);
Serial.write("B");
}
```

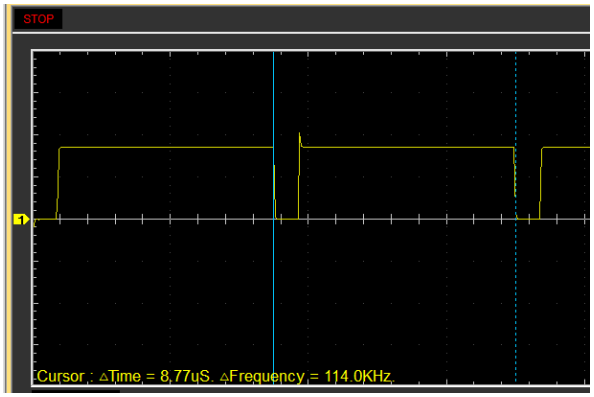


Laufzeit Loop Sketche - ESP8266

Minimal-Sketch auf ESP8266: Trotz höherer Taktfrequenz ist die Laufzeit nicht geringer: 8,8 Mikrosekunden
Die Rückkehr-Zeit zum Loop-Anfang (High-Ausgabe) ist wesentlich länger.

TestZeit_1 §

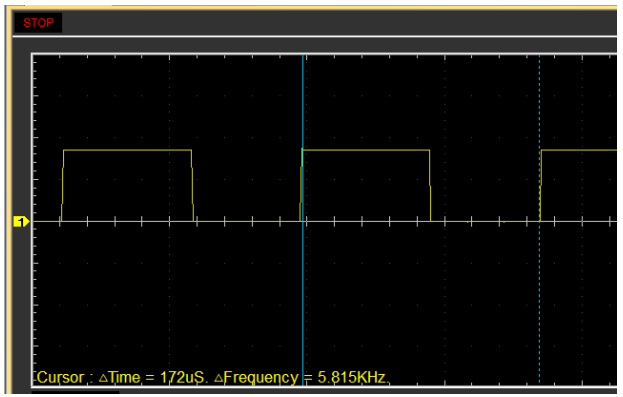
```
void setup()
{
pinMode(D2, OUTPUT);
Serial.begin(115200);
}
void loop()
{
digitalWrite (D2,LOW);
digitalWrite (D2,HIGH);
}
```



Bei Ausgabe am seriellen Monitor kein Unterschied mehr zu Arduino Nano:

TestZeit_3 §

```
void setup()
{
pinMode(D2, OUTPUT);
Serial.begin(115200);
}
void loop()
{
digitalWrite (D2,LOW);
Serial.write("A");
digitalWrite (D2,HIGH);
Serial.write("B");
}
```



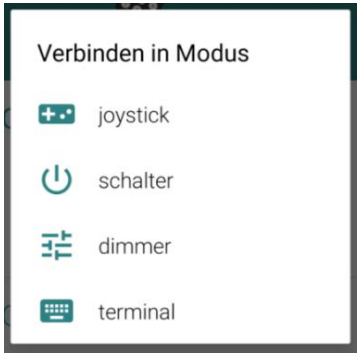
App „Arduino Bluetooth Controller“

Diese App  kann in den Sketchen 86 und 88 alternativ zur App

verwendet werden.
Die App steht für Android-Smartphones zur Verfügung.

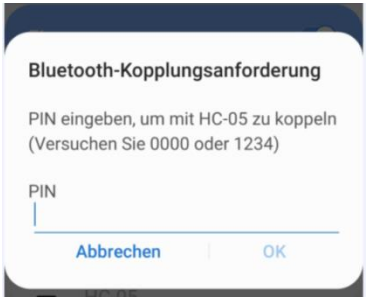
Nach Start der App erscheint:

und nach Klick auf „HC-05“:



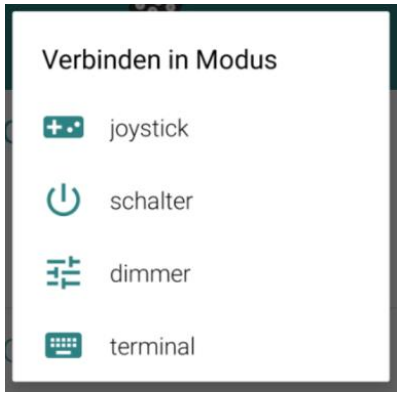
Die Auswahl des Modus funktioniert erst, wenn die Bluetooth-Verbindung zum HC-05 steht.

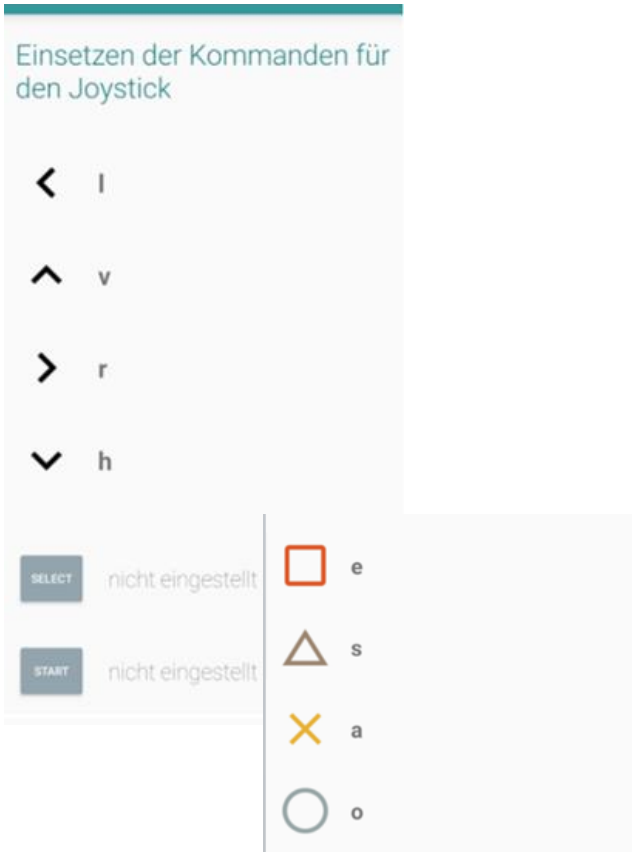
Dann auch Eingabe der PIN: 1234

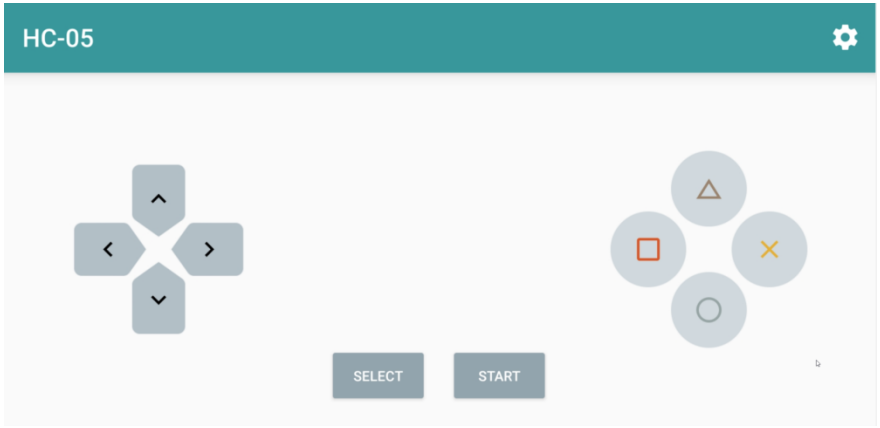


App „Arduino Bluetooth Controller“

Die Steuerung ist im Modus „terminal“ oder im Modus „joystick“ möglich. Für „joystick“ rechts oben anklicken  und die im Sketch verwendeten Zeichen den Tasten zuordnen:







Ende

Inhaltsverzeichnis