

Dokumentation der Projektarbeit

Florian Weber - 44907

7. Juni 2018

Inhaltsverzeichnis

1 Vorwort	3
1.1 Probleme der vorhandenen Carrera Bahn und Zielsetzung zur Projektarbeit	3
2 Funktion - Hardware	4
2.1 Sensoren	5
2.2 Bedienschnittstelle	5
2.3 Arduino	6
2.4 RaspberryPi	6
2.5 Tiefsetzsteller	6
3 Software	8
3.1 RaspberryPi	8
3.2 Arduino	9
3.2.1 Interrupts	9
3.2.1.1 Timer	10
3.2.1.2 Pin Change Interrupt	10
3.2.2 Regler	11
3.2.3 Analog Digital Wandler	13
4 Bedienungsanleitung	14
4.1 Manuelles Fahren	15
4.2 Automatisiertes Fahren	15
4.2.1 Auto-Wait	15
4.2.2 Auto-Run	15
5 Zusammenfassung	16
6 Anhang	17
6.1 Pinbelegung	17

Abbildungsverzeichnis

2.1	Signalfluss	4
2.2	Überblick über die Carrera-Bahn	5
2.3	Bedienschnittstelle	6
3.1	Möglichkeiten zum Programmablauf	9
3.2	Kaskadenreglung Bahngeschwindigkeit	11
4.1	Zustandsautomat Betriebsmodi	14

Kapitel 1

Vorwort

Zu Beginn des Projekts war die Carrera Bahn auf dem Stand, dass die Fahrzeuge konventionell sowie automatisiert gesteuert werden konnten. Außerdem war die Rundenzeitmessung und Visualisierung bereits realisiert. Bei der Energieversorgung konnte zwischen Solarstrom und Netzversorgung gewählt werden. Die Solarstromversorgung wurde pro Bahn durch 2 Solarpanels und einem Tiefsetzsteller mit konstantem Dutycycle hergestellt, die Netzstromversorgung mit einem 15V Schaltnetzteil.

Die Manuelle Steuerung durch den Nutzer wurde realisiert, indem man die klassischen Carrera Drücker, als Variablen Vorwiderstand eingesetzt hat. Im automatisierten Modus musste ein Potentiometer bedient, um die langsame Geschwindigkeit des Autos einzustellen. In den schnellen Abschnitten der Strecke wurde dieses Poti durch ein Relais gebrückt und die volle Versorgungsspannung lag an dem Auto an.

1.1 Probleme der vorhandenen Carrera Bahn und Zielsetzung zur Projektarbeit

Der im Vorwort grob erwähnte Aufbau der vorhandenen Anlage wies einige Probleme auf:

- 1. Solarstromversorgung und Netzversorgung sind nicht chancengleich ausgeführt.(Die Netzversorgung stellt eine stabilisierte Spannungsquelle dar, die Solarstromversorgung allerdings nicht)
- 2. Drift der durch den Poti erzeugten Spannung im automatisierten Modus (Temperaturdrift)
- 3. Framerate der Visualisierung ist zu gering, sodass diese immer die selbe Zahlensequenz nach dem Komma anzeigt.
- Manchmal wird durch einen Aliasingeffekt das Überschreiten eines Sensors in der Bahn nicht erkannt und die Geschwindigkeit wird nicht umgeschalten.

Die oben genannten Probleme sollten durch ein neues Steuer-/Regelkonzept gelöst werden.

Kapitel 2

Funktion - Hardware

Im Folgenden wird der grundlegende Aufbau der Carrera Bahn erklärt. Diese Beschreibung ist immer nur für Bahn-A, da die Bahn-B analog dazu funktioniert. Dazu wird immer wieder auf die Abbildung 2.1, sowie auf die Abbildung 2.2 Bezug genommen.

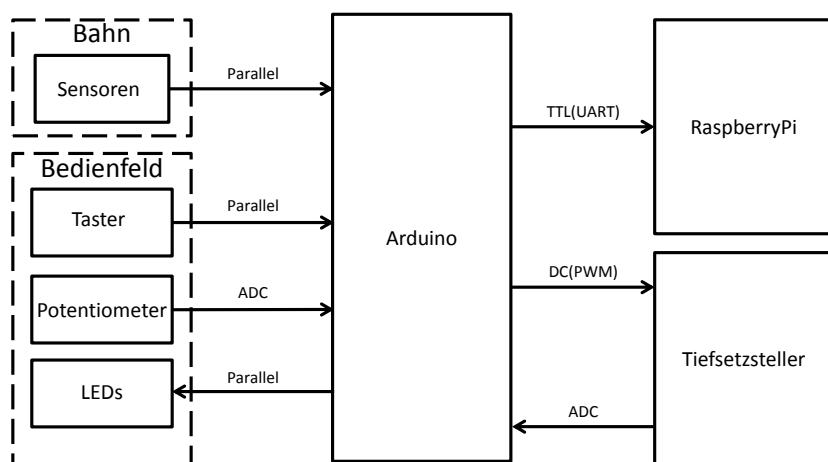


Abbildung 2.1: Signalfloss

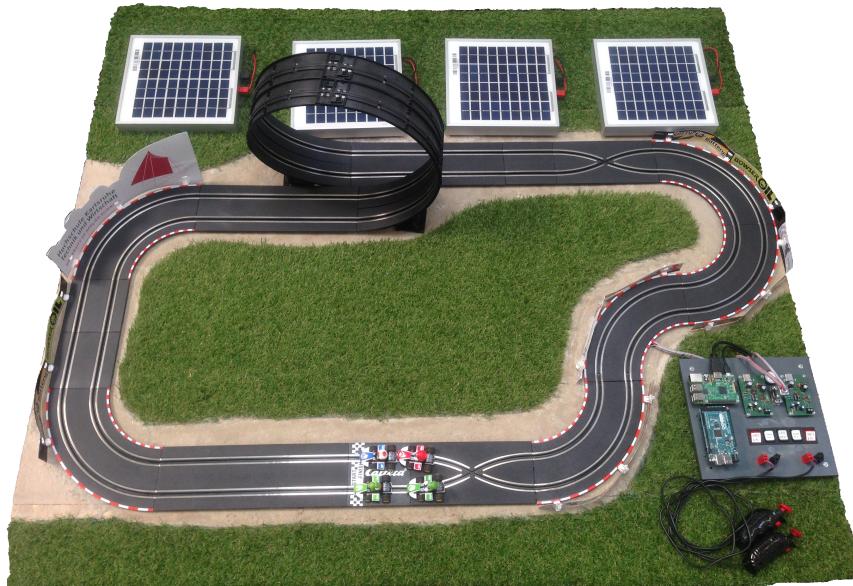


Abbildung 2.2: Überblick über die Carrera-Bahn

2.1 Sensoren

Pro Schiene gibt es 3 Sensoren die als Gabellichtschranke ausgeführt sind. Sensor 0 befindet sich am Start, Sensor 1 vor dem Looping und Sensor 2 Nach dem Looping.

Sensor 0 wird im Wesentlichen nur zur Zeitmessung genutzt, dazu später mehr. Sensor 1 wird genutzt um ein Signal zu generieren, so dass die Steuerung das Auto im Automatik-Modus für den Looping beschleunigen kann. Das Signal von Sensor 2 wird schließlich genutzt um nach dem Looping wieder die langsame Geschwindigkeit zu triggern.

Im manuellen Modus dienen die Sensoren nur zur Zeitmessung für die Visualisierung. Da die Flanke nur sehr kurz ist, lässt sich diese nicht per Polling ohne Aliasing Effekte digitalisieren. Stattdessen wurden die Hardware Pin Change Interrupts des Atmega2560 genutzt.

2.2 Bedienschnittstelle

Das HID besteht aus 3 Tastern und einem Wechselschalter mit Mittelstellung je Bahn. Der Wechselschalter ist zum Auswählen, ob die Bahn mit Energie aus den Solarpanels betrieben wird, oder mit Netzstrom. Ist Dieser in Mittelstellung, wird die Bahn nicht versorgt und das Auto steht - unabhängig des gewählten Modus. Über Taster 0 beziehungsweise Taster 1 lässt sich der Modus der Bahn auswählen, welcher durch die LEDs bei den Tastern signalisiert wird.

Hier steht Automatik und Manuell zur Auswahl. Mit Taster 2 lässt sich der Regler der Bahngeschwindigkeit genauer - Spannung außerhalb des Loopings, auf seinen Startwert zurücksetzen.

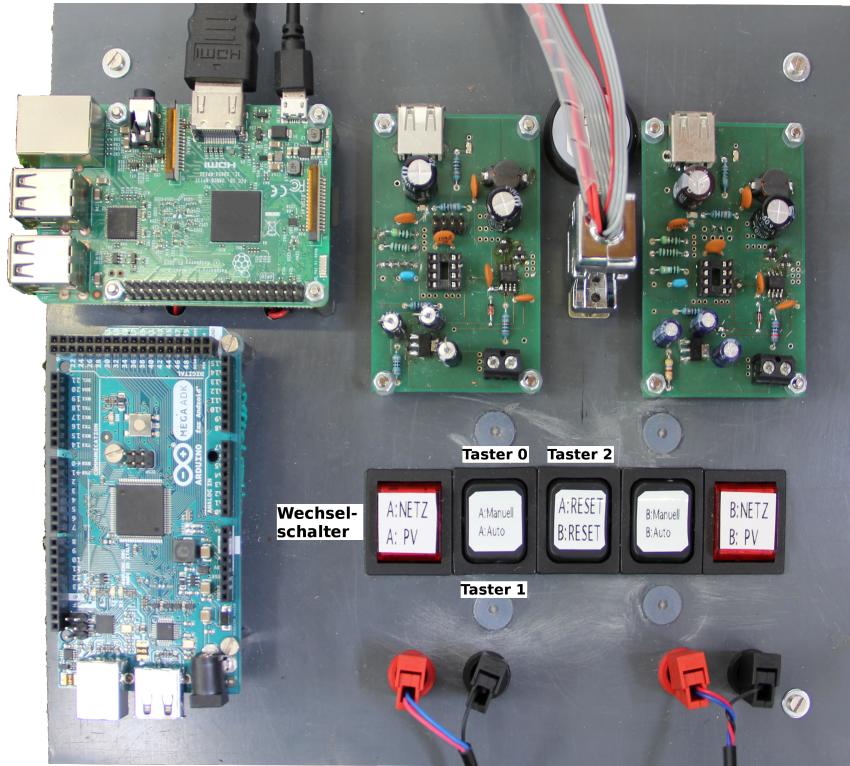


Abbildung 2.3: Bedienschnittstelle

2.3 Arduino

Beim Arduino handelt es sich um einen einfachen Arduino Mega 2560 ADK. Dieser wurde gewählt da es sich um eine preiswerte Platine handelt, die bereits die wichtigsten Beschaltungen des Mikrocontroller enthält wie zum Beispiel die Abblockcondensatoren an der Versorgungsspannung aber auch einen USB-Seriell Wandler den man nutzen kann um sich Daten parallel zum Prozess an ein Terminal auszugeben. Letzteres lässt sich sehr gut für das Debugging nutzen.

2.4 RaspberryPi

Die Visualisierung ist durch ein RaspberryPi Model 3 B realisiert. Dieser empfängt per Uart die codierten Signale der Sensoren, sowie des Reset Tasters der Visualisierung. Die Visualisierung war zu Beginn der Projektarbeit bereits vorhanden und in Form eines Python Skripts implementiert.

2.5 Tiefsetzsteller

Die Tiefsetzsteller fungieren als Stellglieder der Spannungsregelungen der Bahnen. Jeder bekommt sein PWM-Signal(Steuergröße) direkt vom Arduino. Des Weiteren ist über ein Shunt eine Strommessung realisiert. Der Wert liegt zwar

im Arduino digital vor, wird allerdings nicht weiter verarbeitet und ist lediglich für weiterführende Projekte gedacht. Da die Ausgangsspannung des Tiefsetzstellers größer sein kann wie die Referenzspannung des ADC, wird diese über einen Spannungsteiler angepasst und auf einen Kanal des ADC geführt. Diese Spannung stellt die Regelgröße dar.

Kapitel 3

Software

3.1 RaspberryPi

Bei der Software die auf dem RaspberryPi ausgeführt wird, handelt es sich um ein Python Skript. Dieses war zu Beginn der Projektarbeit bereits vorhanden und hat die Sensoren der Bahn parallel eingelesen. Mit den Flanken der Sensoren wurde die Rundenzeit gemessen und die Bestzeit ermittelt.

Dieses Skript wurde größtenteils übernommen und lediglich in der Richtung abgeändert, dass die Trigger-Signale der Sensoren nun über die Serielle Schnittstelle dem Pi mitgeteilt wurden. Des Weiteren wurden diverse Bugs behoben wie zB die zu geringe Framerate der Visualisierung. Als serielle Schnittstelle wurde "ttyAMA0" benutzt.

3.2 Arduino

Die Software des Arduinos wurde, wie oben bereits erwähnt, nicht in der Arduino Entwicklungsumgebung geschrieben. Stattdessen habe ich auf die IDE AtmelStudio 7 zurückgegriffen und den Mikrocontroller per ISP beschrieben. Der Bootloader des Arduino musste dazu entfernt werden. Gründe dazu waren unter anderem die Wahl der Sprache C++, aber auf die Möglichkeit direkt auf die Hardware zuzugreifen(Timer, Hardwareinterrupts...). Letzteres ist notwendig um das Timing des Controllers exakt zu steuern.

3.2.1 Interrupts

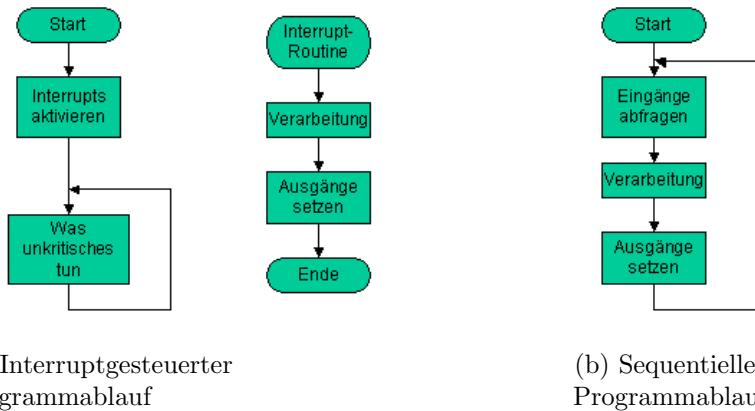


Abbildung 3.1: Möglichkeiten zum Programmablauf
Quelle:Mikrocontroller.net

Der Programmablauf ist größtenteils Interruptgesteuert. Dies hat den Vorteil, dass das Timing nichtmehr von der Länge der MainLoop abhängt und Dinge wie zum Beispiel der Regelagorythmus immer in der selben Frequenz ausgeführt werden. Interruptquellen des Programms sind:

- Timer (Abschnitt 3.2.1.1)
 1. Timer1, Compareregister B match
 2. Timer3, Compareregister A match
 3. Timer4, Compareregister A match
- Pin Change Interrupts (Abschnitt 3.2.1.2)
 4. PCINT0_vect
 5. PCINT1_vect
 6. PCINT2_vect
- Analog Digital Converter
 7. ADC_vect

Zu Anfang des Programms wird die ganze Peripherie initialisiert, und die Variablen mit ihren default Werten geladen. Anschließend werden die Interrupts global freigegeben. Ab hier ist die Steuerung Funktionsfähig.

3.2.1.1 Timer

Für Die Funktion der Steuerung werden viele Timer benötigt.

Da im Mikrocontroller allerdings nur begrenzt Timer zur Verfügung stehen, wurde deren Funktionalität in einem Software-Timer nachgebildet. Dieser erhält sein Takt von einem Hardwaretimer. Die Verwendung der Hard- sowie Softwaretimer kann der Tabelle 3.1 entnommen werden.

Timer	Funktion
Hardwaretimer	
Timer0	Generierung der 2 PWM Kanäle für die Tiefsetzsteller
Timer1	Auslösen der Analog Digital Wandlung
Timer2	Ohne Verwendung
Timer3	Trigger für Spannungsregelung
Timer4	Trigger für Softwaretimer
Softwaretimer	
0..1	Zeitmessung Abschnitt 3 (Nach dem Looping → Startlinie)
2..7	Entprellen der Bahnsensoren
8..14	Entprellen der HID Taster
15..16	Ohne Verwendung
17	Trigger für Zeitmessung Resettaster

Tabelle 3.1: Belegung der Hard- sowie Softwaretimer

3.2.1.2 Pin Change Interrupt

Wenn ein Auto nun auf, Beispielsweise Sensor 0(Bahn A, am Start), fährt wird der Pegel kurze Zeit *low* und nach verlassen des Sensors wieder *high*.

Da Sensor 0 an PCINT16 angeschlossen ist und 16 im Bereich [16,23] liegt, wird das letzte Pin Change Interrupt, PCINT2_vect, zweimal ausgelöst. In der ISR (Interrupt Service Routine) muss nun unterschieden werden, durch welchen Pin das Interrupt ausgelöst wurde. Außerdem gibt es die zwei Möglichkeiten:

- war es ein High → Low Übergang
- war es ein Low → High Übergang

Zum Entprellen des Eingangs wird direkt, nachdem das Event gehandelt wurde, der Eingang deaktiviert. Danach wird ein Software-Timer gestartet der den Eingang nach dessen Ablauf wieder aktiviert. Mit diesem einfachen Prinzip wird

sichergestellt, dass wenn das Auto beim Überfahren des Sensors das dement-sprechende Event nur einmal triggert. Analog zu diesem Sensor 0, ist dies für jeden Sensor sowie Taster realisiert. Die Belegung der benutzten Pin Change Interrupts kann Tabelle 3.2 entnommen werden.

Zugehörigkeit	Signal	Bezeichnung
PCINT0_vect	PCINT4	Button: B-Automatik
	PCINT5	Button: B-Manuell
	PCINT6	Button: A-Automatik
PCINT1_vect	PCINT9	Button: A-Reset
	PCINT10	Button: B-Reset
PCINT2_vect	PCINT16	Sensor: 0
	PCINT17	Sensor: 1
	PCINT18	Sensor: 2
	PCINT19	Sensor: 3
	PCINT20	Sensor: 4
	PCINT21	Sensor: 5
	PCINT22	Button: Reset/Shutdown Pi
	PCINT23	Button: A-Manuell

Tabelle 3.2: Belegung der Pin Change Interrupts

3.2.2 Regler

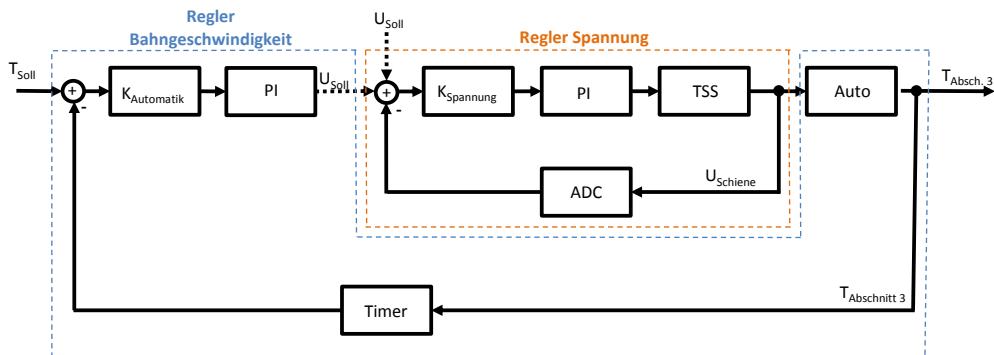


Abbildung 3.2: Kaskadenregelung Bahngeschwindigkeit

Die Regelung für die Bahngeschwindigkeit ist als Kaskadenregelung (Abbildung 3.2) ausgeführt. Dem Führungsregler wird eine Sollzeit für Streckenabschnitt 3 vorgegeben, dieser gibt nun die Sollspannung für den Folgeregler vor. Der Fol-

geregler hat als Stellgröße den Duty-Cycle des zugehörigen Tiefsetzstellers und regelt die Spannung auf der Schiene auf den gegebenen Wert. Der Regelalgorithmus des Spannungsreglers wird in festen Zeitabständen, vorgegeben durch OCR3 (Overflow Compare Register Timer3), zyklisch aufgerufen. Der Regelalgorithmus des Reglers der Bahngeschwindigkeit wird immer dann aufgerufen, wenn ein neuer Zeitwert für den Streckenabschnitt 3 vorliegt. Dieses Vorgehen ist sehr störanfällig, allerdings ist es die einzige Möglichkeit, einen solche Regelung mit den vorhandenen Sensoren zu realisieren.

Der Vorfaktor des Führungsreglers ist eher verhältnismäßig klein gewählt, um ein Überschwingen möglichst zu verhindern. Die gerade beschriebene Regelstrategie ist als solche nur im Automatik-Modus aktiv.

Im manuellen Modus wird die Sollspannung direkt durch den Handregler vorgegeben.

3.2.3 Analog Digital Wandler

Der Mikrocontroller des Arduinos (Atmel Atmega 2560) hat bereits einen 10 bit ADC (Analog Digital Converter) auf dem Chip integriert, dass auf einen externen Wandler verzichtet werden kann. Da der Wandler immer nur einen Messung durchführen kann, hat der Hersteller ein (MUX) Multiplexer integriert um zwischen den einzelnen ADC Pins des Mikrocontroller durchzuschalten. Die Belegung der ADC Pins des Mikrocontrollers ist in Tabelle 6.3 enthalten.

Wenn die angestoßene Messung des ADC abgeschlossen ist, wird ein Interrupt ausgelöst.

Die Steuerung des Multiplexers erfolgt über ein Zustandsautomat der immer dann getriggert wird, wenn wenn die letzte Messung des ADC abgeschlossen ist.

Kapitel 4

Bedienungsanleitung

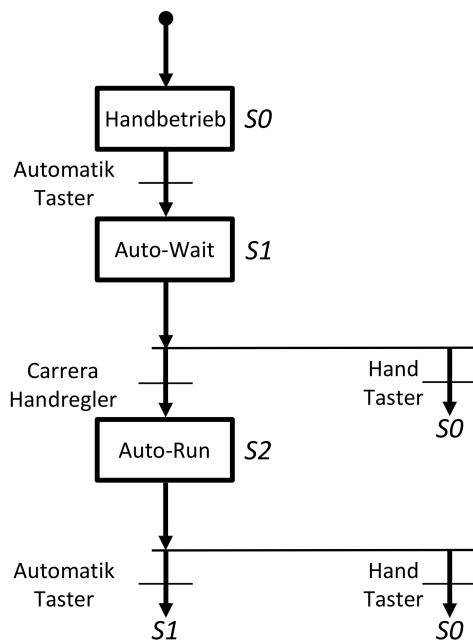


Abbildung 4.1: Zustandsautomat Betriebsmodi

Die Carrera-Bahn beherrscht 2 Modi.

- Manuelles Fahren
- Automatisiertes Fahren

Die Modi können unabhängig voneinander auf beiden Bahnen gewählt werden, sodass man zum Beispiel manuell gegen ein Automatisiertes Fahrzeug antreten kann. Die Wahl eines dementsprechenden Modus wird durch eine Led über, beziehungsweise unter, des jeweiligen Tasters des gewählten Modus signalisiert. Außerdem kann die Energieversorgung der Bahn mit dem Wechselschalter ausgewählt werden. Unabhängig des gewählten Modus, wird in der Visualisierung die Zeit der 2 Abschnitte, sowie die Rundenzeit angezeigt. Nach 30 Runden ist

das Rennen vorbei und die Visualisierung wird statisch. Der Wechsel zwischen den Modi findet nach Abbildung 4.1 statt und ist in den jeweiligen Abschnitten genauer erklärt.

4.1 Manuelles Fahren

Im Modus Manuelles Fahren kann das Fahrzeug konventionell über die orginalen Carrera Handregler gesteuert werden.

4.2 Automatisiertes Fahren

Der Automatisierte Modus ist aufgeschlüsselt in 2 Zustände:

- Auto-wait
- Auto-run

4.2.1 Auto-Wait

Immer wenn in den Modus des Automatisierten Fahrens gewechselt wird, startet die Steuerung in Auto-wait und das Auto steht. Durch kurzes durchdrücken des zugehörigen Handreglers wechselt das Auto in den Auto-run Zustand.

4.2.2 Auto-Run

Im Auto-run Zustand fährt das Auto selbständig. Die Strecke ist in 3 Abschnitte unterteilt.

- 1. Startlinie → Vor dem Looping
- 2. Vor dem Looping → Nach dem Looping
- 3. Nach dem Looping → Startlinie

Die Grenze dafür ist durch die Position der Sensoren festgelegt und somit nicht variabel. In der Visualisierung ist die Strecke von der Startlinie bis vor den Looping, sowie nach dem Looping bis zur Startlinie in Streckenabschnitt 1 zusammengefasst. Die der Weg in dem Looping bildet dort Streckenabschnitt 2.

Kapitel 5

Zusammenfassung

Kapitel 6

Anhang

6.1 Pinbelegung

Sub-D Pin	Funktion
1	VCC (5V)
2	GND
3	-
4	-
5	-
6	Sensor 0
7	Sensor 1
8	Sensor 2
9	Sensor 3
10	Sensor 4
11	Sensor 5
12	Schiene A - Plus
13	Schiene A - Minus
14	Schiene B - Plus
15	Schiene B - Minus

Tabelle 6.1: Pinbelegung der Sub-D Buchse auf dem HID

Raspberry Pin	Funktion
1	VCC MCU (3V3)
2	VCC Input (5V)
3→5	-
6	GND Input
7	-
8	-
9	GND Uart
10	RXD Uart
11→40	-

Tabelle 6.2: Pinbelegung GPIO RaspberryPi

Arduino Pin	Atmel Pin	Funktion
0		Ohne Funktion
1		Ohne Funktion
2		Ohne Funktion
3		Ohne Funktion
4	OC0B	PWM Tiefsetzsteller Schiene B
5		Ohne Funktion
6		Ohne Funktion
7		Ohne Funktion
8		Ohne Funktion
9		Ohne Funktion
10	PCINT4	Taster HID „Schiene B - Automatik“
11	PCINT5	Taster HID „Schiene B - Manuell“
12	PCINT6	Taster HID „Schiene A - Automatik“
13	OC0A	PWM Tiefsetzsteller Schiene A
14	PCINT10	Taster HID „Schiene B - Reset Regler Bahngeschwindigkeit“
15	PCINT9	Taster HID „Schiene A - Reset Regler Bahngeschwindigkeit“
16		Ohne Funktion
17		Ohne Funktion
18		Ohne Funktion
19→45		Ohne Funktion
46	PL3	HID LED „Schiene B - Manuell“
47	PL2	HID LED „Schiene B - Automatik“
48	PL1	HID LED „Schiene A - Manuell“
49	PL0	HID LED „Schiene A - Automatik“
A1→A0	ADC1→ADC0	Shunt Tiefsetzsteller Schiene A
A3→A2	ADC3→ADC2	Shunt Tiefsetzsteller Schiene B
A4	ADC4	Spannung Schiene A
A5	ADC5	Spannung Schiene B
A6	ADC6	Carrera Handregler Schiene A
A7	ADC7	Carrera Handregler Schiene B
A8	PCINT16	Sensor 0 (Schiene A - Startlinie)
A9	PCINT17	Sensor 1 (Schiene A - vor dem Looping)
A10	PCINT18	Sensor 2 (Schiene A - nach dem Looping)
A11	PCINT19	Sensor 3 (Schiene B - Startlinie)
A12	PCINT20	Sensor 4 (Schiene B - vor dem Looping)
A13	PCINT21	Sensor 5 (Schiene B - nach dem Looping)
A14	PCINT22	Taster HID „Reset Rundenzeit Visualisierung“
A15	PCINT23	Taster HID „Schiene A - Manuell“

Tabelle 6.3: Pinbelegung des Arduino