

Dokumentation der Projektarbeit

Florian Weber - 44907

25. Mai 2018

Inhaltsverzeichnis

1 Einleitung	3
1.1 Zielsetzung	3
2 Funktion	4
2.1 Sensoren	5
2.2 Bedienschnittstelle	5
2.3 Arduino	6
2.4 RaspberryPi	6
2.5 Tiefsetzsteller	6
2.6 Software	7
2.6.1 RaspberryPi	7
2.6.2 Arduino	8
3 Bedienungsanleitung	9
3.1 Manuelles Fahren	9
3.2 Automatisiertes Fahren	9
4 Zusammenfassung	10
5 Quellenverzeichnis	11
6 Anhang	12
6.1 Pinbelegung	12
6.1.1 Arduino	12
6.1.2 RaspberryPi	12

Abbildungsverzeichnis

1 Signalfluss	4
2 Überblick über die Carrera-Bahn	4
3 Bedienschnittstelle	5
4 Möglichkeiten zum Programmablauf	8

1 Einleitung

1.1 Zielsetzung

2 Funktion

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 1, sowie auf die Abbildung 2 Bezug genommen.

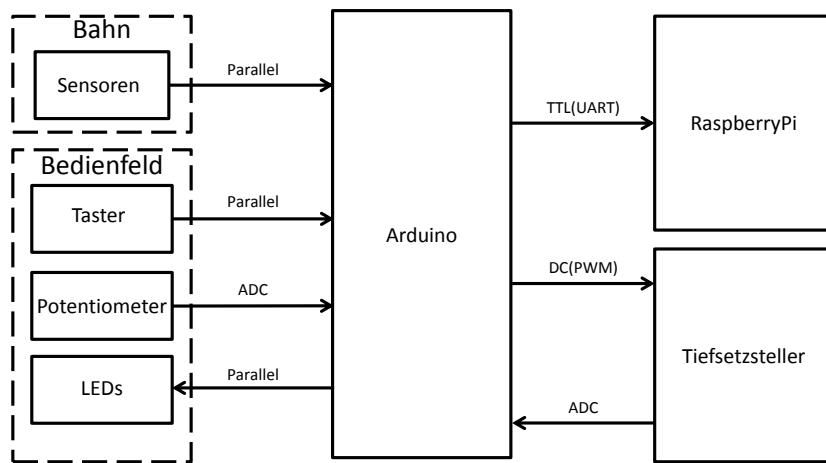


Abbildung 1: Signalfluss



Abbildung 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

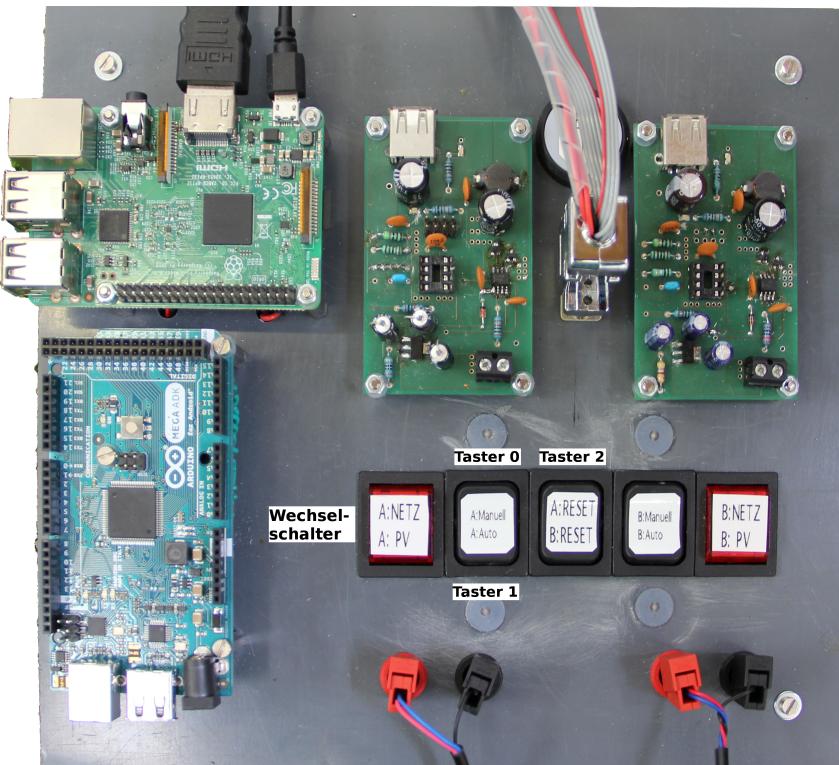


Abbildung 3: Bedienschnittstelle

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.

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 eine Regelgröße dar.

2.6 Software

2.6.1 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 zum Beispiel die zu geringe Framerate der Visualisierung. Als serielle Schnittstelle wurde "ttyAMA0" benutzt.

2.6.2 Arduino

Die Software des Arduino wurde, wie oben bereits erwähnt, nicht in der Arduino Entwicklungsumgebung geschrieben. Stattdessen habe ich auf die IDE AtmelStudio6 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.

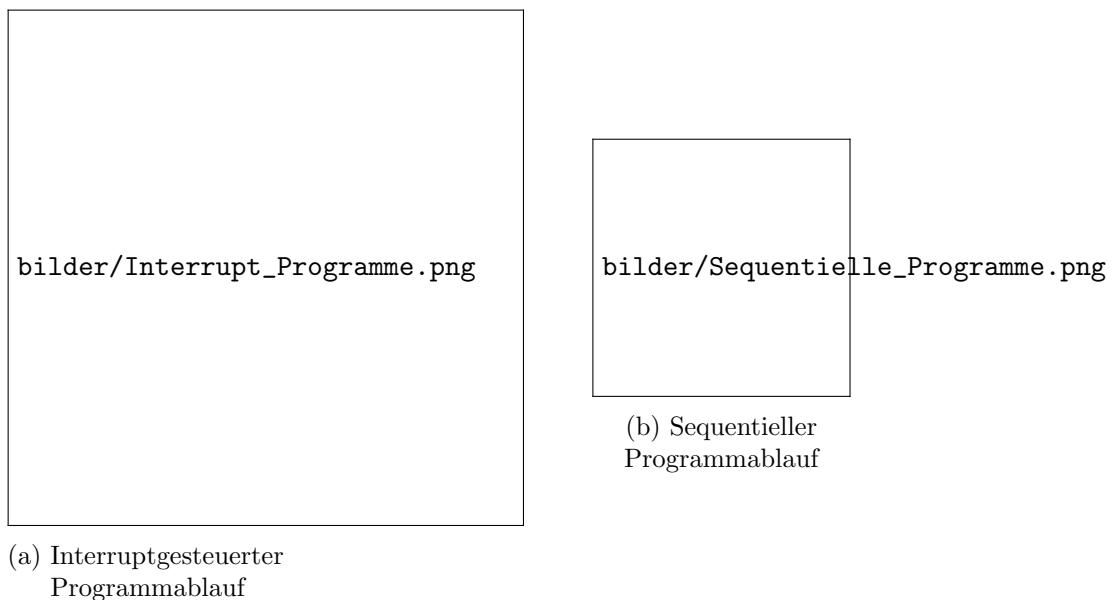


Abbildung 4: Möglichkeiten zum Programmablauf

Quelle:Mikrocontroller.net

Der Programmablauf ist größtenteils Interruptgesteuert. Dies hat den Vorteil, dass das Timing nicht mehr von der Länge der MainLoop abhängt und Dinge wie zum Beispiel der Regelalgorithmus immer in der selben Frequenz ausgeführt werden.

3 Bedienungsanleitung

Die CarreraBahn besitzt 2 auswählbare Modi. Den Modus "Manuelles Fahren", sowie den Modus "Automatisiertes Fahren".

3.1 Manuelles Fahren

3.2 Automatisiertes Fahren

4 Zusammenfassung

5 Quellenverzeichnis

6 Anhang

6.1 Pinbelegung

6.1.1 Arduino

6.1.2 RaspberryPi