Carrera Documentation

Release 0.0.1

BG 13 2011/2012

14. 01. 2012

Inhaltsverzeichnis

Einleitung

1.1 Idee

Alles fing damit an, dass wir zu Beginn des 13. Schuljahres gefragt wurden, wie wir den weiteren Technik-Unterricht angehen wollten. Es gab zum einen die Möglichkeit des herkömmlichen Unterrichts und alternativ dazu die Arbeit an einem eigenständigen und kreativen Projekt. Letztendlich entschieden wir uns dafür, das Projekt des vorherigen Abiturjahrgangs zu übernehmen, welcher eine Carrera-Strecke mit computergestützter Zeitmessung und Rundenzählung ausstatten wollte.

Unter der Betreuung unseres Lehrers Herr Dipl.-Ing. Althen stellten wir uns dieser Herausforderung. Außerdem erhielten wir die Möglichkeit, das Ergebnis der Arbeit beim „Tag der offenen Tür" als Aushängeschild unserer Schule präsentieren zu können.

So starteten wir das Projekt und dachten zunächst darüber nach, wie sich die Rundenmessung umsetzen lässt. Im Verlauf dieser Arbeit kam die Idee, eine *„KI"* (Seite 12) zu entwickeln, welche die Strecke möglichst optimal fahren kann und so einen starken Gegner darstellt.

1.2 Über uns

Das BG 13 Carrera-Racingteam besteht aus 11 Schülern des 13. Jahrgangs des Beruflichen Gymnasiums Eschwege.

Im Rahmen des Technikunterrichts mit dem Thema Regelungstechnik stellten wir uns der Aufgabe, eine Carrera-Strecke zu entwickeln, welche Fahrtzeiten und Rundenanzahlen der Fahrer mit Hilfe eines Computers ermittelt und ausgibt.

Die Mitglieder der Gruppe sind:

- Beck, Örkel
- Fidora, Mario
- Gisa, Maximilian
- Hartmann, Jakob
- Lambach, Manuel
- Ludwig, Martin
- Rohmund, Paul
- Stüber, Jonas
- Wegener, Sören

- Wendorf, Sebastian
- Zindel, Steffen

Jeder von uns hat dabei viel gelernt und in infolgedessen ein persönliches Fazit erstellt.

1.2.1 Stephen Beck

Mein Einstieg in die Gruppenarbeit gestaltete sich recht schwierig, weil ich aus gesundheitlichen Gründen in der Anfangsphase nicht mitwirken konnte und daher die Aufgabenverteilung und die Fortschritte des Projekts an mir vorbeigingen.

Nachdem ich mich nun anschließend mit den Neuerungen einigermaßen vertraut gemacht habe, übernahm ich überwiegend kleinere Aufgaben, wie z.B. das Vorbereiten von Kabeln (was sich in Form von zurechtschneiden und abisolieren äußerte), das Überarbeiten der Runden- und Zeitanzeige in Kooperation mit Sören und vieler anderer unterstützenden Tätigkeiten. Im weiteren Verlauf der Arbeit beschäftigte ich mich außerdem mit der Erstellung eines Logos sowie eines Banners, das für die Außenseite der Steilkurve gedacht war. Das Logo wollten wir außerdem an kleinen „Türmen" befestigt und entlang der Strecke verteilen, jedoch fand sich kein geeignetes Material für die Fertigung dieser Türme, sodass diese Idee verworfen wurde.

Des Weiteren half ich beim Befestigen der Platten, auf welchen die Strecke steht, beim Verlegen der Verkabelung unterhalb der Tische und suchte/deutete Datenblätter verschiedener Transistoren und Verstärkern, um den jeweils geeigntsten auswählen zu können.

Ich habe in der Dokumentation an den Kapiteln „Inhaltsverzeichnis", „Idee" und „Über uns" gearbeitet und darüber hinaus einige Texte sprachlich und grammatisch überarbeitet.

1.2.2 Mario Fidora

Damit für jeden Außenstehenden klar ist, worum es sich bei unserem Projekt handelt und wer daran beteiligt ist, habe ich zusammen mit Stephen Beck und Manuel Lambach ein interessantes Logo entworfen.

Da die alte Strecke durch zu viele Kurven nicht dem gewünschten Fahrspaß zur Verfügung stellte haben wir entschlossen, eine neue Strecke zu entwickeln. Jene Strecke konstruirten wir mittels des Programmes „Carrera-Streckenplaner". Nachdem die neue Strecke Entworfen wurde, musste dafür gesorgt werden, dass die Streckenspannung bei jedem Streckenabschnitt aufrecht erhalten wird. Zudem musste die neue Strecke dem Fundament angepasst werden. Die Strecke sollte für jeden Fahrer, egal wo er startet, die gleichen Vorraussetzungen haben.

Da es sich um 2 nebeneinander verlaufenden Strecken mit jeweils 2 Bahnen handelte musste dafür gesorgt werden, dass beide Strecken den selben Abstand zueinander haben. Da die Autos verschiedene Fahreigenschaften im Bezug auf Geschwindigkeit und Kurvenverhalten besaßen wurde versucht möglichst gleich schnelle Wagen zu finden.

Eine weitere Aufgabe bestand darin die Datenblätter bestimmter Transistoren herauszusuchen und ihre Vor- und Nachteile gegeneinander abzuwägen.

Ich habe in der Dokumentation an den Kapiteln Einleitung, Hardware und Carrera-Strecke gearbeitet.

1.2.3 Maximilian Gisa

Da mich das ganzes Projekt sehr interessiert hat, war ich von Anfang an mit viel Enthusiasmus bei der Sache. Ich half mit, die ersten *Lichtschranken* (Seite 15) mit dem UE9 zu verbinden und zu verbessern. Danach wandte ich mich mit Steffen und Paul den Relais zu. Da wir viele Leute waren und es sehr ineffektiv war, wenn jeder mal hier und mal da etwas tat, versuchte ich die Arbeit strukturiert zu verteilen und gleichzeitig möglichst viel mitzuarbeiten. Das war jedoch nicht im Bereich Software nötig, da Sören fast eigenständig arbeitete.

Nachdem wir damit begannen, das selbstständige Fahren in Angriff zu nehmen, verbrachte ich sehr viel Zeit damit, eine geeignete Verstärkerschaltung zu entwickeln, was mir mit Hilfe von Herrn Althen auch gelang.

Außerdem half ich gemeinsam mit Steffen bei der Kommunikation zwischen den einzelnen Teams, um eine gute Abstimmung der Hardware auf die Software und umgekehrt zu erreichen.

Ich bemühte mich, alle aufkommenden Fragen zufriedenstellend zu beantworten und konnte mich ebenso gut auf meine Team-Kollegen verlassen.

Auch habe ich oft an der *Verkabelung* (Seite 18) gearbeitet und hier und da ein wenig *gelötet* (Seite 19).

Ich beschäftigte mich somit hauptsächlich mit den Gebieten:

- *Lichtschranken* (Seite 15)
- *Ampel* (Seite 15)
- *Streckenfreischaltung* (Seite 18)
- *Verkabelung* (Seite 18)
- *UE9* (Seite 21)
- *Arduino* (Seite 22)

Fazit: Leztendlich haben wir alles zum laufen gebracht, einige Kleinigkeiten, wie etwa die richtige Ansteuerung der Ampel, ausgenommen. Ich kann von mir behaupten, dass ich viel erreicht und gelernt habe, und dass sich die freiwilligen Zusatzstunden wirklich gelohnt haben. Da ich von allen Bereichen einen mehr oder weniger intensiven Eindruck erhalten habe, kann ich zweifellos sagen, dass wir eine sehr eindrucksvolle Arbeit geleistet haben.

Vorschlag: Den folgenden Satz weglassen, da du bereits oben ne ausführliche Liste inklusive Seitenverweisen hast.

Ich habe in der Dokumentation an den Kapiteln Hardware (ausführlich), Einleitung, Carrera-Stecke gearbeitet.

1.2.4 Jakob Hartmann

Da dieses Projekt nun einmal wieder etwas anderes und neues war, ging ich mit vollem Elan und Einsatz ans Werk. Da aber die Vielzahl der Masse an „Mitarbeitern" zu groß war, war zu Beginn des Projektes noch keine feste Position meines Arbeitens zu erkennen. Aus diesem Grund übernahm ich Hilfsarbeiten jeglicher Art, welche Maximilian Giebenrath mir zuteilte.

Als nun ein gewisser Fluss in das Projekt einzog, wurden die Arbeiten, welche ich übernahm, spezieller und ich übernahm eine Vielzahl an *Lötarbeiten* (Seite 19). Während zu Beginn nur die Verlängerung der Schiebregler-zur-Geschwindigkeitsendkontrolle auf dem Plan standen und die Überarbeitung mancher Carreraautos, auch mit Auswechslung der Schleifkontakte, übernahm ich später die Feinoptimierung der Lichtschranken. So überarbeite ich teilweise die Ständer der Lichtschranken, um ein bestmögliches Überprüfen der Strecke zu gewährleisten und ein sicheres Erkennen der Autos zu ermöglichen.

Später als Herr Althen die Idee einbrachte, die Lichtschranken mit Steckern zu versehen, um ein leichtes Austauschen derselbigen zu gewähren, übernahm ich die Lötarbeiten an allen Lichtschranken um Herr Althens Traum zu ermöglichen und meine Lötfertigkeiten auch auf Kleinteile zu skillen. Doch nicht nur Lötarbeiten waren meine Aufgabe, sondern engargierte ich mich auch bei der Streckenveränderung. Da die Neuplanung nun abgeschlossen war und wir die Strecke neu aufbauten, ergaben sich immer wieder Probleme, die passenden Streckenteile zu finden und einzusetzen. Immer wenn sich Probleme ergaben, auch bei anderen Mitarbeitern, versuchte ich Zeit zu finden um diese Probleme zu lösen.

1.2.5 Manuel Lambach

Beim Aufbau und Verkabelung der Lichtschranke beteiligte ich mich am Umbau. Die erste Verkabelung war leider fehlerhaft und so zogen wir im zweiten Anlauf eine Gesamtleitung, die die Lichtschranken in Reihe schaltet. Ausschließlich die Datenleitung mussten wir einzeln verlegen.

Um unseren Projekt ein cooles Image zu verleihen, kümmerte ich mich zusammen mit Stephen Beck und Mario Fidora um die Gestaltung eines geeigneten Logos.

Da die alte Strecke zu lang war und eindeutig zu viel Kurven besaß, entschieden Mario Fidora und ich uns für eine Neuentwicklung der Bahn. Wir setzten uns als Ziel eine schnelle und spannende Strecke zu entwickeln. Wichtig war, dass alle vier Spuren gleich lang sind und die Strecke sich auf unserem Fundament realisieren lässt. Ein faires Spiel stand im Vordergrund.

Bei der Befestigung der Bahn füllten Martin Ludwig und ich die Bahnteile mit Heißkleber auf. Besonders ordentlich und gewissenhaft musste diese Aufgabe erledigt werden, da am Ende keine Lücken entstehen durften. Ansonsten unterbricht die Streckenspannung an unterschiedlichen Stellen. Bahnreinigung, Tuning Autos

1.2.6 Martin Ludwig

Grundidee unseres Projektes war es das Projekt der letzten 13er weiterzuführen und zu optimieren. Zu Beginn unserer Arbeit war es so, dass noch nicht genau geklärt war, welche Aufgaben der Einzelne nun übernimmt. So begannen wir mit der ersten Aufgabe eine *Ampel* (Seite 15) und *Startfreigabe* (Seite 18) zu bauen, bei der Lichtschranken zum Einsatz kamen, welche ich teils zusammengelötet habe.

Des Weiteren mussten *Lötarbeiten* (Seite 19) für die Zeitmessung und *Verkabelung* (Seite 18) gemacht werden.

Später kümmerte ich mich dann um die Streckenplanung und die Umsetzung dieser in die Realität, was den Zusammenbau des Fundaments und die Befestigung der neuen Strecke mit einbezog.

Hierbei mussten die einzelnen Streckenteile mit Hilfe des Senkbohrers und Akkuschraubers befestigt werden.

Alles in Allem konnte ich durch dieses Projekt sehr viel mitnehmen, zum Beispiel, wie Software und Hardware miteinander kooperieren und welche Probleme hierbei auftreten können, allgemein welche physikalischen Schwierigkeiten in der Realität in Zusammenhang mit der Elektrotechnik zustande kommen und wie wir teils eigenständig, teils aber auch mit Hilfe des Lehrers, zu Lösungsansätzen kamen.

Ich denke, dass dieses Projekt die perfekte Abwechslung zum typischen Unterricht ist, da wir fast selbstständig zu Wissen kommen konnten, aber auch durch Exkurse und die entstehenden Probleme schwierige Dinge fast spielerisch lernten. Mein Fazit ist, dass auch die nächsten Jahrgänge ähnliche Projekte durchführen sollten und so ein praktischer Lernbereich in den Vordergrund rückt, welcher gewisse Vorteile mit sich bringt, welche der „normale Unterricht“ einfach nicht bieten kann.

1.2.7 Paul Rohmund

Bei dem Carrera-Projekt befasste ich mich größtenteils mit der Hardware. Dabei arbeitete ich hauptsächlich an den *Lichtschranken* (Seite 15) für die Zeitmessung, der *Ampel* (Seite 15), der *Streckenfreischaltung* (Seite 18) und der *Verkabelung* (Seite 18), und führte weiterhin kleinere *Lötarbeiten* (Seite 19) durch.

Anfangs bestand der Großteil meiner Arbeit aus dem Zusammensetzen und Löten der Lichtschranken, welche vorerst für die Rundenzeitmessung und später für das computergesteuerte Fahren nötigen waren. Des Weiteren war ich für das Verlöten der Ampel-LEDs und den dazugehörigen Transistoren zuständig. Durch Steckverbindungen an beiden Seiten der Transistor-Platine wurde die Verkabelung wesentlich übersichtlicher und geordneter.

Bei der Streckenfreischaltung lötete ich die Bauteile der Platine an und kümmerte mich um die Verkabelung für die Relais mit Lüsterklemmen. Darüberhinaus versuchten wir das Zusammenspiel von Lichtschranken und Arduino herzustellen, was uns allerdings erst nach Hilfe von Herrn Althen gelungen ist. Außerdem führte ich weitere kleinere Arbeiten bei der Befestigung der Bahn durch.

Das Projekt war sehr interessant und hat mir persönlich viel Spaß gemacht, da man hier die theoretischen Kenntnisse aus Datenverarbeitungstechnik und Elektrotechnik in der Praxis umsetzen konnte und man sich unvorhergesehenen Problemen stellen musste. Somit war es auch eine willkommene Beschäftigung während der Freistunden.

1.2.8 Jonas Stüber

Bei unserem Carrera-Projekt habe ich ausschließlich an der Hardware gearbeitet. Diese Hardwarearbeit beinhaltete unter anderem den Streckenaufbau. Dabei steckte ich die einzelnen Streckenteile - streng nach ausgefertigtem Streckenplan - zusammen und breitete sie passend auf den Holzplatten aus. Da wir die Bahn jedoch so konstruieren wollten, dass sie schnell abbaubar und schnell wieder aufbaubar ist, bauten wir sie an den Holzplattenübergängen noch auf kleinere Streckenteile um. So können die einzelnen „Streckenpuzzleteile” schnell zusammengesetzt werden. Danach fixierte ich mit der Unterstützung anderer Gruppenmitglieder die Streckenteile, die dauerhaft fest auf den Platten sein sollen. Dazu füllten wir eine kleine Kammer mit Heißkleber. Anschließend bohrten wir ein Loch durch den Heißkleber und das jeweilige Streckenteil und schraubten es danach auf der Holzplatte fest.

Beim Streckenaufbau versuchten wir keine stark verunreinigten Teile zu verwenden, jedoch war es nicht möglich die Strecke nur aus sauberen Teilen aufzubauen. Deshalb reinigte ich die verschmutzten Streckenteile bestmöglich, damit ein guter Kontakt zwischen Auto und Fahrbahn gegeben ist.

In der Stunde, in der ich krankheitsbedingt gefehlt habe, bauten meine Mitstreiter das Grundgerüst eines „Turmes“, in dem später das „elektronische Herz“ der Bahn ihren Platz finden sollte. Ich übernahm die Aufgabe eine Plexiglasscheibe, welche ich zuvor mit Maximilian auf die richtige Größe zugeschnitten hatte, in den Zwischenboden des Turmes einzupassen und zu fixieren.

Des Weiteren habe ich an der Elektrik/Elektronik mitgewirkt, u. a. legte ich diverse Stromkabel aus. Zum Beispiel verteilte ich die begrenzte Anzahl von Steckerleisten so im Raum, dass überall dort, wo Strom benötigt wird, auch welcher zur Verfügung stand. Nachdem wir uns dazu entschlossen hatten, mit den Lichtschranken die Geschwindigkeit des Autos zu steuern, benötigten wir Halterungen, an denen wir die Lichtschranken mit den dazugehörigen Platinen befestigen konnten. Daraufhin baute ich kurzerhand sieben Halterungen aus dem Metallbaukasten zusammen, welche aber später durch Aluminiumhalter ausgetauscht wurden, weil man diese einfach unter die Bahn schieben kann.

Des Öfteren hatte ich auch mit der Kabellache oder dem Lötkolben zu tun. Beispielsweise isolierte ich diverse Kabel ab, verzinnte sie und lötete sie dann zum Beispiel an die Lichtschrankenplatinen bzw. Lichtschranken. Außerdem verkabelte ich nach den Portanweisungen von Maximilan den Arduino, verklebte die LEDs in die Ampelanlage ein und verkabelte diese auch entsprechend. Falls ich kurzzeitig keine Hardwarearbeiten zu verrichten hatte, habe ich mit dem Staubsauger die Streckenteile von Abisolierresten oder Sägespäne befreit oder auch mal den Klassenraum gefegt.

1.2.9 Sören Wegener

Da ich bereits in der 11. Klasse für meine guten Kenntnisse in der Softwareentwicklung bekannt war, war es von Anfang an klar, dass ich diesen Teil übernehmen werde. Zu Beginn stand noch viel Recherche an und es wurden kleinere Testscripte geschrieben. Dazu gehörte beispielsweise, wie man einen Port des UE9 lesend oder schreibend verwendet, ob der Einsatz der Scriptsprache Python möglich ist oder ob auf „klassische” Programmiersprachen wie C zurückgegriffen werden muss, die das Erzeugen von maschinennahen Programmen erlauben. Da LabJack eine komfortable und performante Schnittstelle für Python anbietet, war nach wenigen Tests klar, dass die Kombination dieser Techniken miteinander problemlos möglich ist. Im Laufe der Entwicklung beschäftigte ich mich außerdem mit Matplotlib (http://matplotlib.sourceforge.net/), welches das Plotten von Graphen für unterschiedlichste Zwecke erlaubt, in unserem wird es zur Visualisierung der Messergebnisse genutzt.

Später, als wir das *eigenständige Fahren* (Seite 12) hinzufügen wollten, kam ich auf die Idee, dieses in Ermangelung eines weiteren UE9 mithilfe eines Arduino Uno (http://arduino.cc/en/Main/ArduinoBoardUno) zu implementieren. Nach einigen Besprechungen mit dem Team gelangten wir zu dem Schluss, dass ein Arduino Uno sowohl Hardware- als auch Softwareseitig unseren Anforderungen und finanziellen Mitteln entsprach. Außerdem sieht’s eh viel cooler aus, wenn wir sagen können, dass „der Chip da” fährt, und ich musste mich nicht damit beschäftigen, wie man mehrere angeschlossene UE9 voneinander unterscheidet. :)

Gegen Ende übernahm ich die Zusammenstellung und Strukturierung der Dokumentation. Das Grundgerüst für diese war bereits geschaffen, zum Einsatz kommt Sphinx (http://sphinx.pocoo.org), welches die Dokumentation der Software unterstützt und auch den restlichen Abschnitten ein professionelles Erscheinungsbild in verschiedensten Formaten wie etwa PDF oder HTML verleiht.

Durch die Kombination von Hardware und Software habe ich auch einiges über die Elektrotechnik gelernt.

Die Entwicklung der Software ist chronologisch auf Github (https://github.com/swege/Carrera/commits/master) festgehalten. Das Repository ist Teil der Dokumentation und wurde lediglich aus Übersichtlichkeitsgründen nicht abgedruckt.

1.2.10 Sebastian Wendorf

Zu Beginn des Projektes hatte ich ein mögliches Aussehen der Oberfläche für die Software skizziert. Gemeinsam mit Sören habe ich angefangen eine Oberfläche zu konstruieren. Da er sich besser mit der Programmierung auskennt, hat er sie später implementiert. Von da an habe ich an vielen Stellen mitgeholfen.

Um die Lichtschranken mit dem Arduino zu verbinden, verlöte ich die Lichtschranken mit Kabeln, die zum Tower führten, da aber die Fehlersuche umständlich war, entschied man sich für eine andere, bessere Lösung. Auch waren ein paar den Lichtschranken kaputt gewesen. Über die Leitungen liefen Masse, Stromversorgung und Daten, die Strom- und Masseleitungen wurden am Tower zusammengeführt, damit nur noch jeweils eine Leitung übrig blieb, mit dieser ersten Verkabelung begann auch dann das Programm zum Selbstfahren. Durch Messungen mit dem Messgerät konnten aber auch die defekten Lichtschranken lokaliesiert werden.

Nachdem das Relais das erstemal verlötet war, funktionierte dies nicht wie beabsichtigt. Ich machte mich daran, den Fehler zu suchen und fand ihn mit Hilfe des Schaltbildes. Es war die Masse war verkehrt angeschlossen. Neben der ersten Verkabelung der Lichtschranken und der Reperatur des Steuerungsrelais machte ich vorallem viele kleinere Korrekturen und Reperaturen. Verbaute ich das die Lichtschranken des Start/Ziel mit der Ampel richtig und isolierte die Kontakte damit es zu keinen Überbrückungen kommt.

Da ich in der anderen Technikgruppe bin, war es problematisch an dem Projekt zu arbeiten, ich nutze daher die Freistunde am Donnerstag, um dies Auszugleichen.

Auch war ich bei „Unterricht Live“ da gewesen.

1.2.11 Steffen Zindel

Bevor wir mit unserem Projekt starten konnte, mussten wir uns zunächst in kleinere Gruppen unterteilen. Da ich mich gerne mit Elektrotechnik beschäftige, wollte ich mich mit der Umsetzung der *Lichtschranken* (Seite 15), der *Streckenfreigabe* (Seite 18) und der *Schaltung der Ampel* (Seite 15) auseinandersetzen. Da die Ampel unserer erster Schritt sein sollte, setzten wir uns zusammen und planten eine Transistorschaltung für die LEDs der Ampel.

Diese praktische Arbeit war eine gelungene Abwechslung zum Unterricht, da wir nun das erste Mal wirklich löten und etwas konstruieren konnten. Natürlich mussten wir uns zuerst etwas einlesen, da wir die benötigte Spannung für die Transistoren und LEDs genau wissen mussten.

Nachdem wir auch handwerklich tätig werden konnten, indem wir die Halterung für die LEDs bauten, widmeten wir uns der Streckenfreigabe und den Lichtschranken. Dabei fiel mir auf, dass unsere Gruppeneinteilung nicht optimal war, da wir nun viel mehr Absprache mit Sören und der Software brauchten. Also haben Maximilian und ich versucht den Kontakt zwischen der Software und Hardware herzustellen. Dafür beschäftigten wir uns mit den Ausgängen des Arduinos und UE9 und verknüpften diese mit den Lichtschranken.

Dazu mussten wir auch eine Relaisschaltung aufbauen, welche uns durch die zuerst unübersichtliche Verkabelung mehrere Probleme bereitete. Die Arbeit mit der Elektrotechnik sowie die Verknüpfung der Hardware mit der Software war sehr vielseitig und bereitete mir sehr viel Spaß. Deshalb widmeten wir auch unzählige Freistunden um weiter an dem Projekt zu arbeiten, da die Verkabelung uns sehr lange aufhielt. Das war auch der Teil, bei dem es am schwierigsten war, sich zu motivieren, da durch die Unübersichtlichkeit der Kabelverlegung viele Probleme auftraten. Um diese zu lösen mussten wir mehrmals jedes Kabel und jede Lichtschranke einzeln überprüfen, was uns sehr schwer fiel, da wir am Anfang keine Steckverbinder benutzten.

Daher mussten wir später unser Konzept überarbeiten und somit auch die komplette Kabelverlegung auf die Steckverbinder anpassen. Das war eine mühselige Arbeit, allerdings lohnte es sich, da nun Probleme leichter erkannt und gelöst werden konnten. Alles in Allem machte die Arbeit an dem Projekt sehr viel Spaß, da man sich mit verschiedensten Themengebieten der Technik beschäftigen musste, um die Verknüpfung zwischen Hard- und Software so zu gestalten.

Software

Die Software besteht aus zwei Teilen: der in Python implementierten Messsoftware, welche mithilfe eines LabJack UE9 die Schnittstelle zur *Hardware* (Seite 15) bietet, und dem später hinzugefügten Autopiloten, welcher aus einem Arduino Uno besteht. Das Programm ist als „Arduino Sketch" geschrieben, einem vereinfachten C Dialekt, der speziell für Arduino Chips entwickelt wurde.

Die Software haben wir öffentlich zugänglich gemacht, sie kann von Interessierten von Github (https://github.com/swege/Carrera) bezogen werden.

2.1 Datenbank

Zu Beginn wurde über eine Datenbank nachgedacht, welche die durch die Software erfassten Renndaten permanent abspeichern sollte. Ein entsprechendes Layout dazu wurde bereits entworfen, später während der Entwicklung der Messsoftware jedoch vernachlässigt, da wir uns zunächst auf den reinen Rennbetrieb konzentrieren wollten.

Der bisherige Entwurf des Layouts wurde nicht verworfen und ist in den Pythonmodulen `models` und `utils` enthalten, da die Datenbank aktuell jedoch nicht zum Einsatz kommt und es somit „toter Code" ist, wird an dieser Stelle nicht weiter darauf eingegangen.

2.2 Zeitmessung

Dieses Kapitel beinhaltet Informationen über die Datenerfassungssoftware. Es beinhaltet beispielsweise die Beschreibung von erstellten Funktionen und Methoden, implementierten Schnittstellen und die Bedienung des Programmes.

2.2.1 Spielmodi

Basisklasse aller Spielmodi:

class `modes.`**`Mode`**(*device*, *player_num=2*)

Baseclass for all modes.

`cancel`()

Cancel a match.

`countdown`()

Handle the countdown for the start.

`poll`()

Do some "game logic". Needs to be called as often as possible.

run()
Countdown is over, start the race!

save()
Write the acquired data to the database.

start()
Start a new match.

Match

Bei einem Match fahren 2-4 Spieler eine bestimmte Anzahl an Runden. Nach dem Absolvieren aller Runden wird der Strom auf der entsprechenden Spur abgeschaltet.

class modes.**Match**(*device*, *player_num=2*, *rounds=5*)

check_conditions()
Check if a player made all rounds.

Turns off the track if a player made all his rounds.

Time Attack

Im Modus "Time Attack" wird eine bestimmte Zeit vorgegeben. Nach Ablauf der Zeit wird auf allen Spuren der Strom abgeschaltet. Sieger ist, wer die meisten Runden geschafft hat.

class modes.**TimeAttack**(*device*, *player_num=2*, *seconds=0*)

check_conditions()
Check if the time is over

Used to power off the tracks.

Knock Out

In einem "Knock Out" Rennen fliegt in jeder Runde der langsamste Spieler raus. Dadurch sind maximal drei Runden möglich, bei diesen stehen die Spieler aber unter hohem Druck und Nervenkitzel.

class modes.**KnockOut**(*device*, *player_num=2*)

2.2.2 Grafische Oberfläche

class gui.**Carrera**
Class which handles the GTK interface.

add_player()
Add a new player box to the player list.

clear_racewindow()
Clear the racewindow

Removes any infromation from the racewindow to provide a clean interface

num_players
Contains the total playernumber.

power_off(*track*)
Power the given track off. Pass -1 to power all off

power_on(*track*)
Power the given track on. Pass -1 to power all on

quit (**args*, ***kwargs*)
Quit the GUI and cancel any running race.

run ()
Display the GUI and start the mainloop.

start_knockout ()
Start a new "knock out" race.

Works the same like *start_match*, it just uses the gamemode "KnockOut"

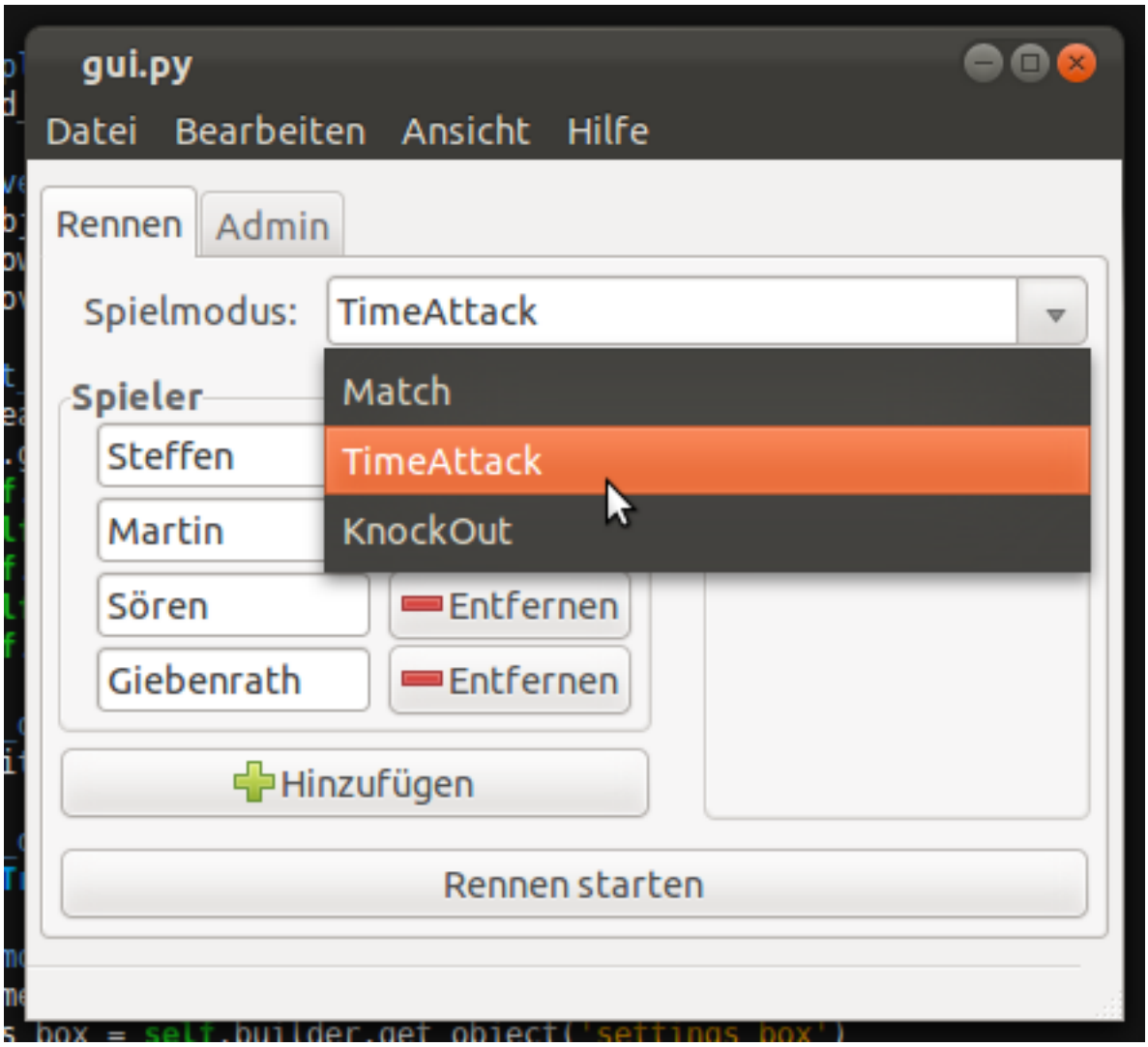
start_match ()
Start a new "match" race.

Fetches information from the main window to prepare the racewindow. It then goes in an infinite loop to poll the sensors. After all players finished the loop will be interrupted.

start_time_attack ()
Start a new "time attack" race.

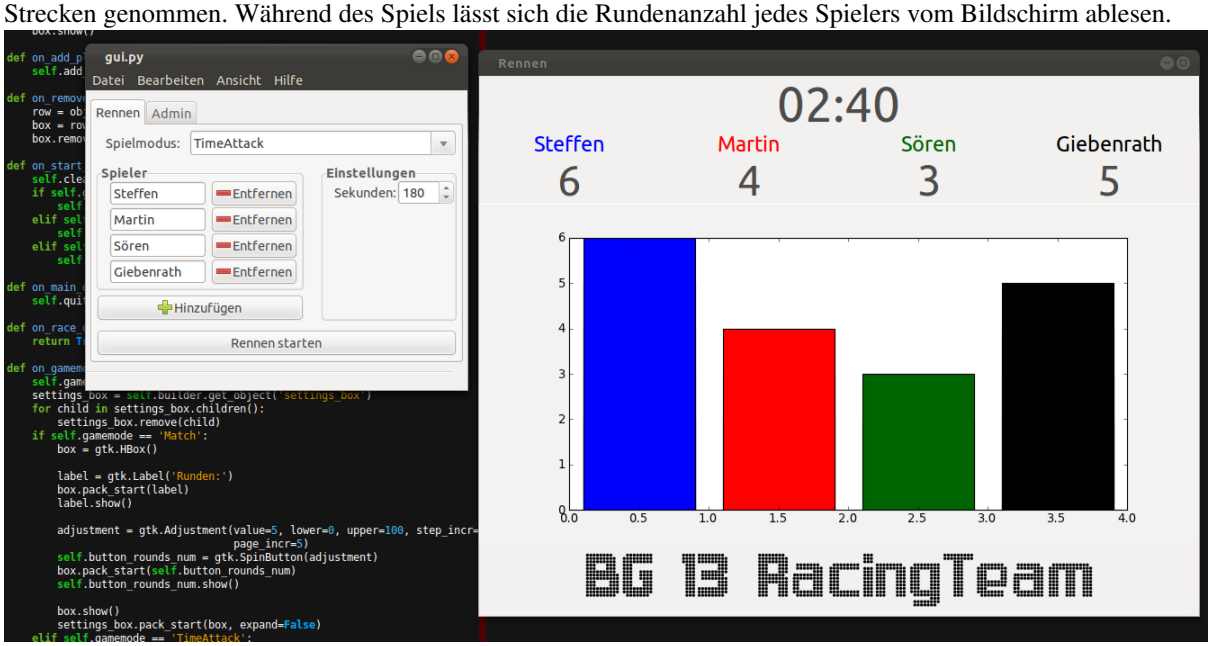
Works the same like *start_match*, it just uses the gamemode "TimeAttack"

Die verschiedenen Spielmodi können über das Hauptfenster angewählt werden. Der Einstellungsbereich ändert sich dabei passend zum Spielmodus.



Ist der „Match" Modus gewählt, lässt sich die Rundenanzahl einstellen. Während des Rennens werden die Rundenzeiten und am Ende die Gesamtzeit angezeigt. In Planung ist außerdem eine Zusammenfassung mit konkreten Zeitangaben, welche am Ende eines Rennens ausgegeben und gedruckt werden kann.

Im TimeAttack Modus lässt sich die Zeit für ein Rennen einstellen. Nach Ablauf der Zeit wird der Strom von allen Strecken genommen. Während des Spiels lässt sich die Rundenanzahl jedes Spielers vom Bildschirm ablesen.



Der KnockOut Modus erlaubt keine weiteren Einstellungen. Die Oberfäche zeigt während des Rennens alle Spielernamen an, und graut die Namen derjenigen aus, die aus dem Rennen ausscheiden.

2.2.3 Devices / UE9 Interface

LabJack bietet nicht nur Linuxtreiber für ihr UE9 an, sondern ebenfalls eine Schnittstelle zu Python (http://labjack.com/support/labjackpython). Damit ist es möglich, das gesamte Gerät vergleichsweise einfach zu nutzen.

Um die Nutzung für unsere Zwecke weiter zu vereinfachen, sind wir noch einen Schritt weiter gegangen: Mittels Vererbung haben wir die von LabJack bereitgestellte `UE9` Klasse um Methoden erweitert, welche speziell auf unsere Zwecke abgestimmt sind. Dazu gehören etwa Methoden, welche die *Ampelsequenz* (Seite 15) abspielen oder die *Lichtschranken* (Seite 15) an der Startlinie abfragen. Auf diese Weise kann das eigentliche Messprogramm frei von kryptischen Portabfragen bleiben und stattdessen Methoden mit aussagekräftigen Namen verwenden, welche das Programm lesbarer machen.

Weiterhin hat diese Abstraktion den zusätzlichen Vorteil, dass das Messprogramm auch mit anderen Geräten verwendet werden kann: Es kann selbst unverändert bleiben, lediglich die Abstraktionsschicht muss für andere Geräte neu geschrieben werden. Neben dem Interface für das UE9 wurde daher auch ein virtuelles Messgerät geschrieben. Das virtuelle Gerät ist sehr nützlich, wenn man die Software testen möchte, jedoch keinen UE9 zur Verfügung hat. Das virtuelle Gerät löst dabei zufällig unterschiedliche Aktionen aus, wie etwa das Passieren des Ziels, und zeigt Informationen an, die es erhält, etwa das ein- oder ausschalten der *Stromfreigabe* (Seite 18).

Standardmäßig versucht das Programm zunächst, eine Verbindung zu einem realen UE9 aufzubauen, wenn das fehlschlägt, wird auf das virtuelle zurückgegriffen. Geplant ist außerdem ein Einstellungsdialog, welcher die Möglichkeit bieten soll, während des Betriebes zwischen verschiedenen Geräten zu wechseln.

Im Folgenden sind die von LabJack bereitgestellten Methoden nicht aufgeführt.

class devices.**UE9**(**args*, ***kwargs*)

computer_speed
Deprecated. Selfdriving car is now controlled via the Arduino.

power_off(**tracks*)
Disables the power for the given tracks. Pass -1 to disable all.

power_on(**tracks*)
Enables the power for the given tracks. Pass -1 to enable all.

sensor_state(*num*)
Returns the state of the sensors at the finish line.

The state is returned as a list with four booleans, one for each sensor. It uses the ports EIO0 - EIO3.

traffic_lights
Set the state of the traffic lights.

- 0 = 0 red, 0 green
- 1 = 1 red, 0 green
- 2 = 2 red, 0 green
- 3 = 3 red, 0 green
- 4 = 0 red, 1 green

Uses ports FIO0 - FIO4

class devices.**Virtual**
Virtual device for testing purposes.

power_off(**tracks*)
Disables the power for the given tracks. Pass -1 to disable all.

power_on(**tracks*)
Enables the power for the given tracks. Pass -1 to enable all.

sensor_state(*num*)
Returns the state of the sensors at the finish line.

The state is returned as a list with four booleans, one for each sensor. It "activates" the sensors randomly.

traffic_lights
Set the state of the traffic lights.

- 0 = 0 red, 0 green
- 1 = 1 red, 0 green
- 2 = 2 red, 0 green
- 3 = 3 red, 0 green

- 4 = 0 red, 1 green

Die Dokumentation der Pinbelegung befindet sich im *Hardware Teil* (Seite 21).

2.3 Graphen

Für die Generierung der Graphen wurde Matplotlib (http://matplotlib.sourceforge.net/) verwendet.

class `graphs.`**Graph**(*num_players=2*)

draw()

show()

class `graphs.`**Match**(*num_players=1*, *rounds=1*)

add(*player*, *time*)

update_axis()

class `graphs.`**TimeAttack**(*num_players=2*)

add(*player*)

update_axis()

2.4 Autopilot / KI

Als zusätzliche Herausforderung haben wir dem Arduino beigebracht, die Strecke möglichst schnell zu fahren. Um die aktuelle Autoposition festzustellen, verwendet der Arduino die Daten der *Lichtschranken* (Seite 15). Zu jeder Lichtschranke ordnet der Arduino drei Arrays zu: Das erste Array enthält die Geschwindigkeit, mit der das Auto fahren soll, sobald es die Lichtschranke passiert. Das zweite Array beinhaltet einen Delay, also eine Zeitverzögerung (in Millisekunden). Nach Ablauf der dort angegebenen Zeit beschleunigt das Auto auf den Wert, welcher im dritten Array gespeichert ist. Dieser Ansatz wurde gewählt, da es so mit relativ wenig Lichtschranken möglich ist, ein gutes Fahrverhalten zu realisieren.

Der Arduino bietet zwar keinen analogen Ausgang, die Geschwindigkeit kann jedoch mittels *PWM-Signal* (Seite 22) gesteuert werden.

Das Programm ist in Form eines `Arduino Sketches` erhältlich.

2.4.1 Probleme

Aktuell ist die *Streckenfreigabe* (Seite 18) nur für die manuell gesteuerten Fahrzeuge umgesetzt, der Arduino lässt sich nicht stoppen. Grundsätzlich gibt es zwei Möglichkeiten, dieses Problem zu lösen. Zum einen elektrotechnisch, indem abhängig vom Signal des UE9 der Arduino physikalisch von der Strecke getrennt wird. Dies würde keine Änderungen der Software erfordern. Der andere Ansatz ist, ein einziges Kabel direkt zwischen Arduino und UE9 zu verlegen. Über dieses Kabel ist zwar keine physikalische Trennung möglich, wie sie bei den menschlichen Spielern angewandt wird, es kann jedoch eine “Bitte” in Form eines Signals (HIGH oder LOW) an den Arduino gesendet werden. Die Software des Arduinos müsste nun lediglich durch eine einfache Abfrage erweitert werden, welche abhängig von diesem Signal fährt oder stehen bleibt.

Ein größeres und für uns derzeit unlösbares Problem stellt die „Lernfähigkeit“ des Arduinos da. Aktuell müssen die Werte für die einzelnen Lichtschranken per Hand ausprobiert und eingegeben werden. Eine zunächst scheinbar naheliegende Lösung stellte sich nach den manuellen Tests als ungenügend heraus:

Man könnte doch einfach die gesamte Strecke zunächst langsam abfahren, und nach jeder Runde ein wenig beschleunigen. Fällt der Wagen nach einer bestimmten Lichtschranke aus der Bahn, kann an dieser Stelle nicht weiter beschleunigt werden. Dies wird solange durchgeführt, bis für jede Lichtschranke die Maximalgeschwindigkeit gefunden wurde.

Dies ist jedoch aus einigen Gründen nur schwerlich umsetzbar:

- Bei den manuellen Tests wurde auch die Position der Lichtschranken optimiert, da erkannt wurde, dass sie zu nah oder zu weit entfernt von den Kurven stehen. Diese Optimierung wäre auf diese Weise nicht möglich
- Da der Wagen mit einer gewissen Geschwindigkeit ankommt, ist diese auch für die folgende Kurve von Bedeutung, nicht nur die Geschwindigkeit, die der Wagen annehmen soll. Der Wagen kann nicht sofort auf die gewünschte Geschwindigkeit gebracht werden. Somit ist die Frage „fliegt er raus oder nicht?“ auch von der vorherigen Lichtschranke abhängig. Doch ist es nun sinnvoller, bei der aktuellen Lichtschranke stärker vom Gas zu gehen, oder bei der vorherigen? Das ist wiederum vom Aufbau der Strecke abhängig - wurden zuvor viele Kurven verwendet, war es eine lange Gerade, gab es Steigungen oder Gefälle? All dies kann nicht mithilfe der Lichtschranken erfasst werden, sondern erfordert manuelle Analysen und Tests oder mehr Daten.

Hardware

3.1 Lichtschranken

Die Lichtschranken verwendeten wir zunächst für die *Zeitmessung* (Seite 7) der einzelnen Runden bzw. Rennen. Hierfür verwendeten wir ein Gestell an dem für jede der vier Bahnen eine Lichtschranke befestigt war. Dieses Gestell war bereits vorhanden, sodass wir nur noch die Software anpassen mussten. Um die Daten der Lichtschranken am Computer auslesen zu können, verwendeten wir ein UE9, auf das später noch genauer eingegangen wird.

Zu den anfänglich vier Lichtschranken kamen im Laufe der Zeit acht weitere hinzu. Diese stellten wir vor jeder wichtigen Kurve entlang der Stecke auf, um ein computergesteuertes Fahrzeug mit der nötigen Spannung zu versorgen, sodass es die Runde optimal fahren kann.

Die Lichtschranken bestehen aus einer Platine und einem Reflexsensor; fährt ein Auto unter diesem Sensor hindurch, sorgt die Schaltung dafür, dass mittels eines Pull-Down Widerstandes der High-Pin des Arduinos auf Masse gezogen wird. Dabei machen wir uns die Eigenschaften des Arduinos zu Nutzen, da es bereits die Möglichkeit eines Pull-Down Widerstandes besitzt müssen wir lediglich die Lichtschranke durch ein Kabel mit dem entsprechndem Port verbinden.

Als Hilfe diente uns ein Tutorial von arduino.cc (http://arduino.cc/it/Tutorial/Button), in welchem am Beispiel eines einfachen Buttons die Schaltung und dazu nötige Software erläutert wurde.

3.1.1 Probleme

Gegen Ende des Projekts, nachdem wir uns lange mit dem selbständigen Fahren beschäftigt haben, schlossen wir die Zeitmessung, die bereits lange abgeschlossen war. und mussten uns einem neuem Problem stellen ein reproduzierbarer Fehler welcher dafür sorgte, dass einige Lichtschranken bei Aktivierung andere beeinflusst. Dieser Fehler kam Zustande, weil wir ein Verlängerungskabel verwendeten, welches den Ansprüchen nicht genügte, sodass wir es noch austauschen mussten. Das Kabel welches durch Quetschverbindugen an den Streckern befestigt war muss an diesen Quetschverbindungen nich perfekt isoliert haben, sodass es ständig zu Fehlsignalen kam.

3.2 Ampel

Um eine visuelle Anzeige für Rennstart und -stopp zur Verfügung zu stellen, entschieden wir uns dafür, eine Ampel im „Formel-1-Stil“ zu bauen. Diese Ampel besteht aus 8 großen LEDs, welche an einem rechteckigen Holzbrett angebracht sind. Sie befindet sich direkt an der Zeitmessbrücke und ist somit besonders gut bei Start- und Zieleinfahrt von den Fahrern einsehbar.

Die Startreihenfolge ist an jene der Formel-1 angelehnt und verläuft wie folgt:

- 1. Reihe rot

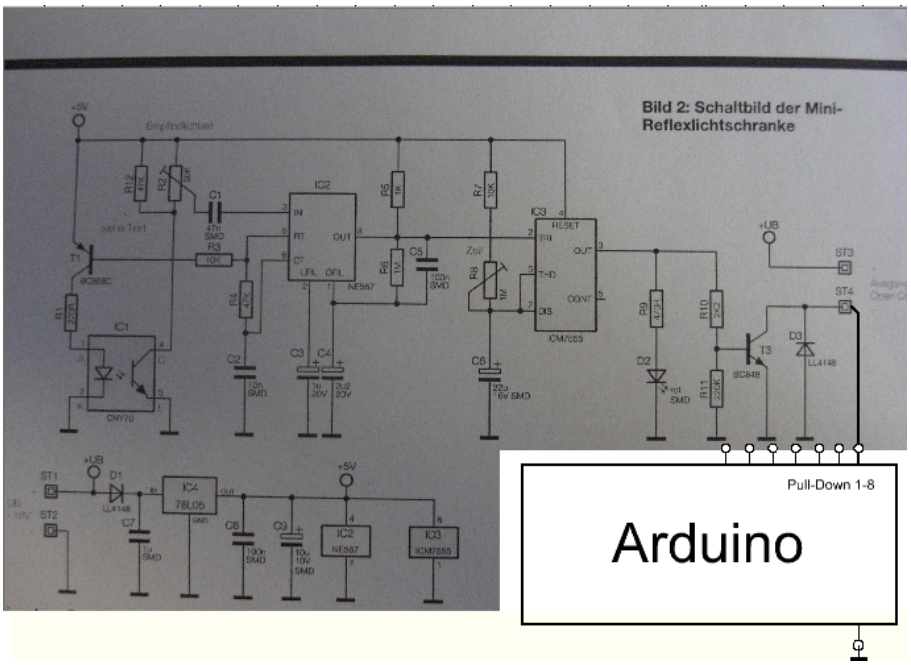


Abbildung 3.1: Schaltbild einer Reflexlichtschranke mit Anschluss an das Arduino

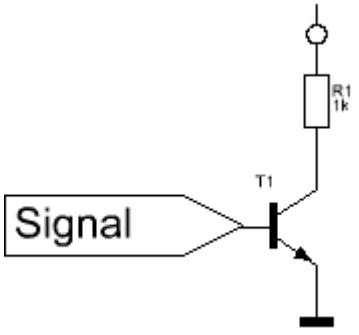
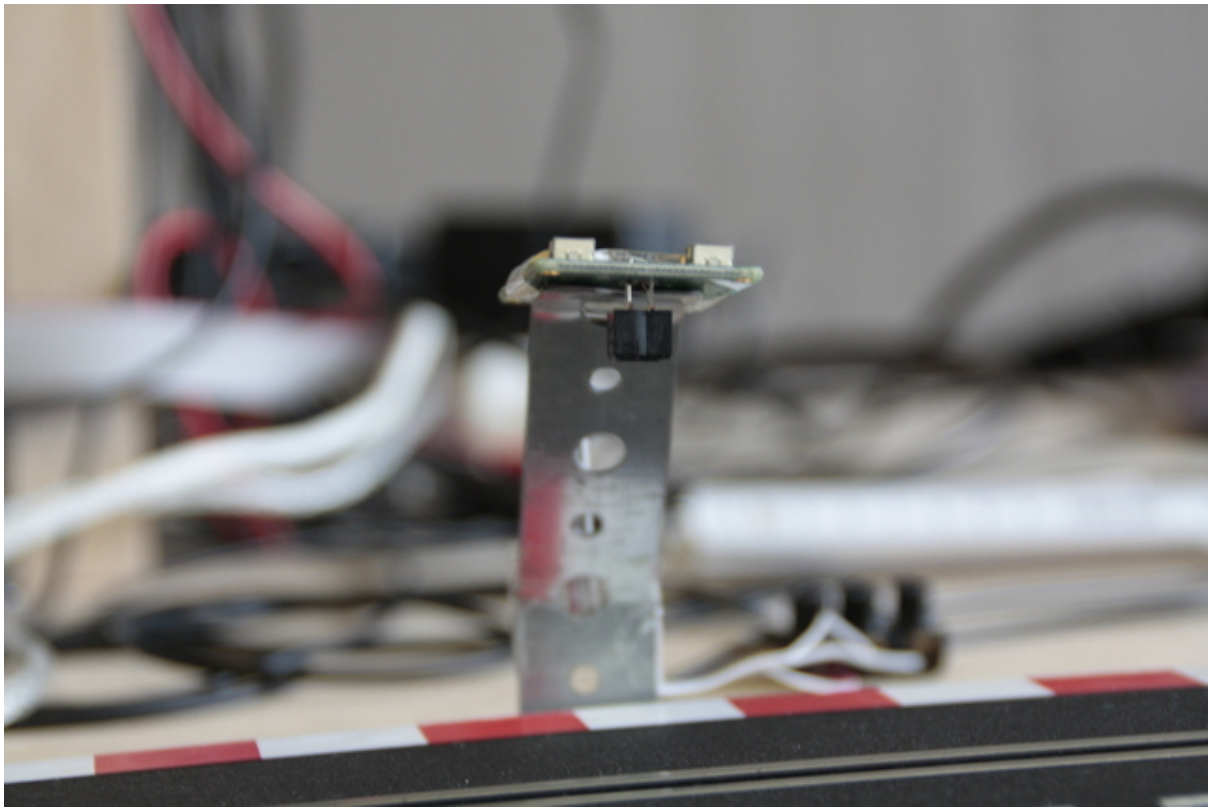
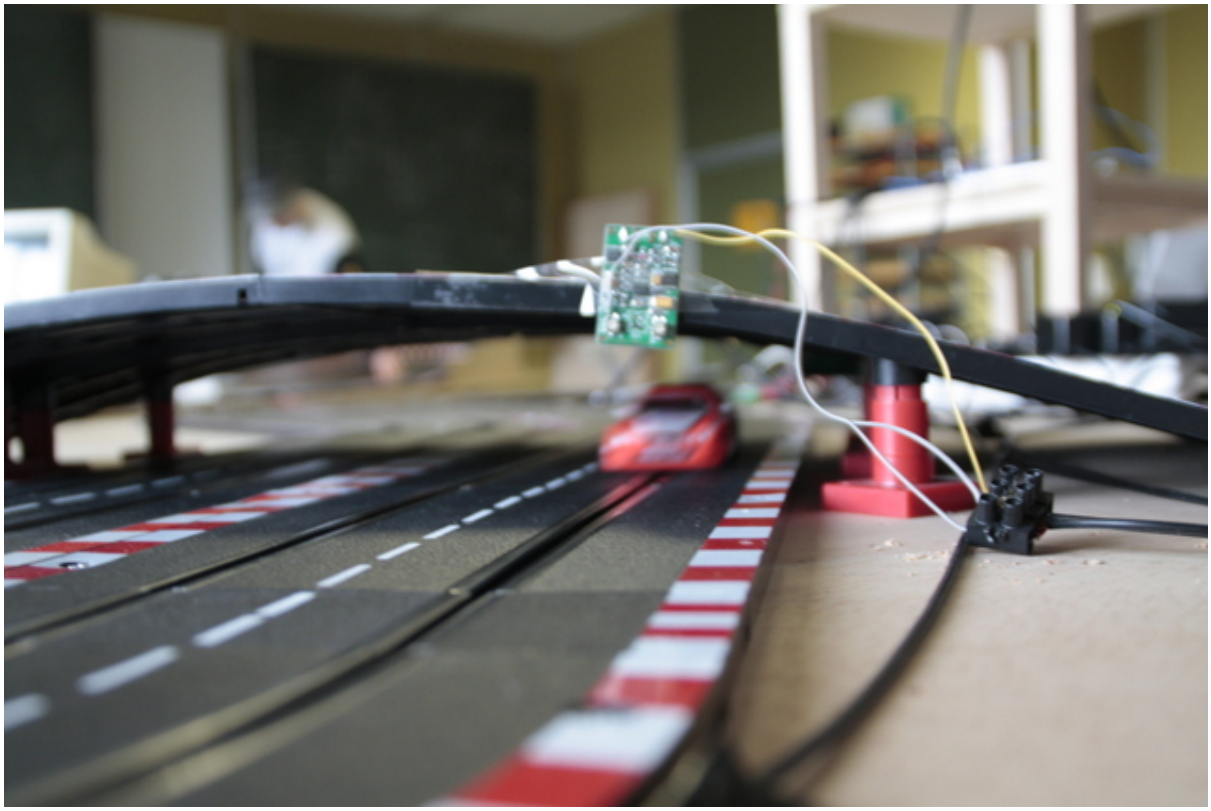


Abbildung 3.2: Prinzip eines Pull-Down





- 2. Reihe rot
- 3. Reihe rot
- Reihe 1,2 und 3 aus, 4. Reihe grün

Die LEDs werden dabei vom UE9 angesteuert, wobei die Leistungsverstärkung mittels Transistoren gewährleistet wird. Sie werden mit 12V Spannung betrieben.

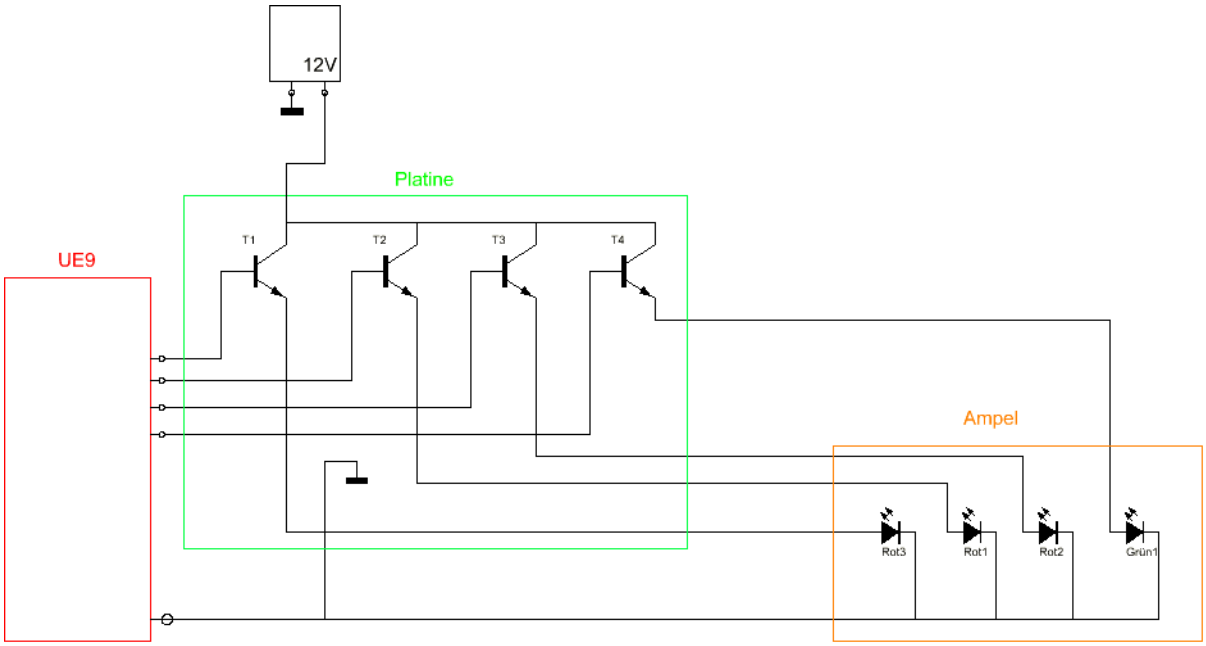


Abbildung 3.3: Schaltbild

3.2.1 Probleme

Unerwarteterweise bereitete uns die Ampel mit am meisten Schwierigkeiten, da es wiederholt Probleme mit der Verstärkerschaltung gab und die richtige Reihenfolge für das Aufleuchten der LEDs lange von Problemen verfolgt war. Dieses Problem konnten wir mit viel Geduld lösen, sodass die LEDs letztendlich in richtiger Reihenfolge aufleuchten.

3.3 Streckenfreischaltung

Wir wollten eine Möglichkeit schaffen den Strom der Strecken zu steuern, um in erster Linie Frühstarts ausschließen zu können. Des Weiteren wollten wir den Zustand der Fahrzeuge schützen, weil nach Beendigung eines Rennens kein weiteres Fahren mehr möglich sein sollte. Dies ermöglichte darüberhinaus weitere Rennmodi, wie etwa den `Knock-Out Modus` (Seite 8), bei dem der jeweils langsamste Fahrer einer Runde ausscheidet, indem die Stromzufuhr für seine Bahn unterbrochen wird.

Um dies zu ermöglichen, nutzen wir eine Platine, auf der Relais vorhanden sind, welche den Strom jeder Strecke freigeben, sperren und regulieren können.

3.4 Verkabelung

Die Verkabelung war zunächst nur provisorisch und recht unübersichtlich. Doch nach der Zeit bereitete uns dies verschiedene Probleme. Zum einem war es mit Schwierigkeiten verbunden die Übersicht zu erhalten um eventuelle Änderungen vorzunehmen oder eine Neuerung zu implementieren. Außerdem war der „Kabelsalat" eine zu

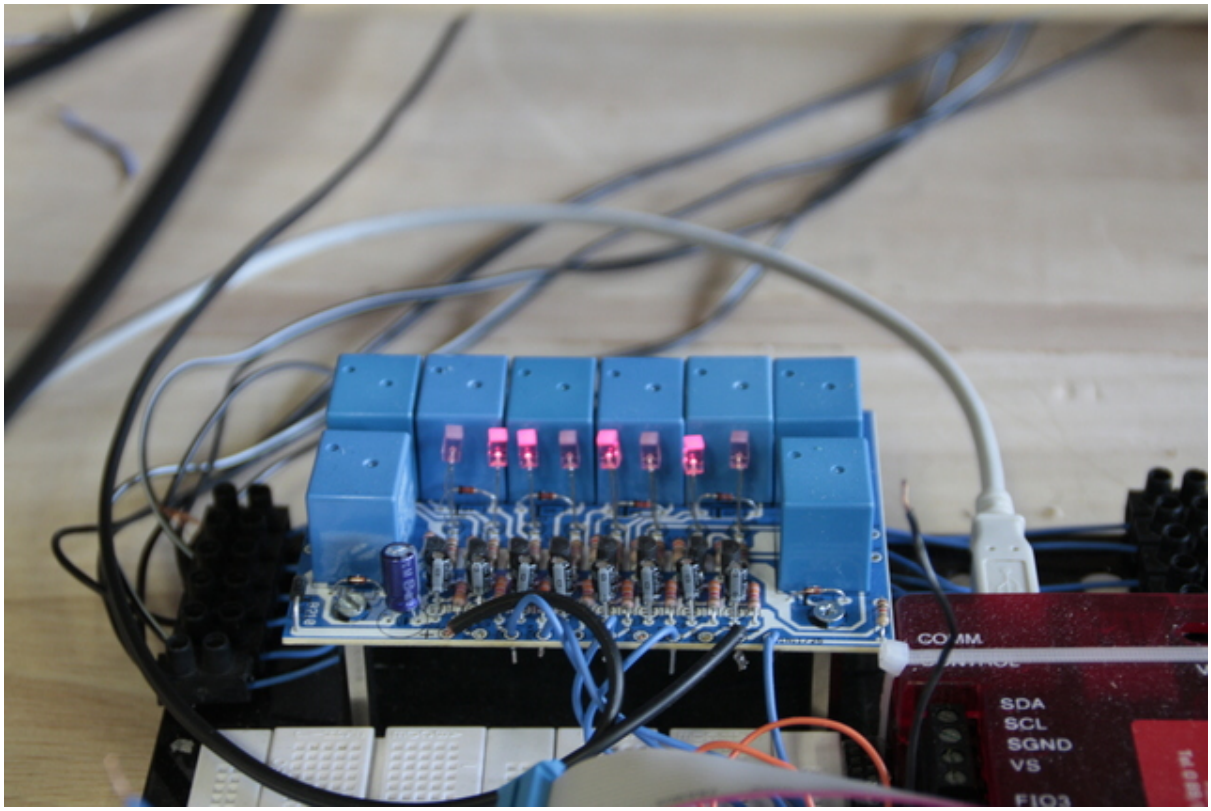


Abbildung 3.4: Relais in Betrieb

große Fehlerquelle, sodass die Verkabelung überdacht und anschließend überarbeitet wurde. Durch eine gelungene Strukturierung konnten wir fortan effizienter und leichter arbeiten.

3.4.1 Probleme

Kabelsalat, Fehlerquelle, Unordnung.

3.5 Lötarbeiten

Es gab immer genug zu löten, egal ob Platinen, Kabel, die verbunden werden mussten, oder Reperaturarbeiten. Jeder, außer Sören, hat wahrscheinlich früher oder später einmal gelötet.

3.5.1 Probleme

Oftmals standen zu wenig Lötkolben zur Verfügung, da die anderen Gruppen auch Anspruch auf die Koblen erhoben. Auch war teiweise das Lötzinn aufgebraucht.

Desweiteren ergaben sich Probleme bei der später hinzugefügten Steckverbindung zwischen den Lichtschranken und dem Datenkabel. Da die Stecker, welche uns Herr Althen zur Verfügung stellte, sehr klein waren ergaben sich Probleme beim löten. Durch die Kompaktheit der Stecker musste man diese erst mit Tesafilm fixieren, um diese sauber mit den *Lichtschranken* (Seite 15) verlöten zu können.

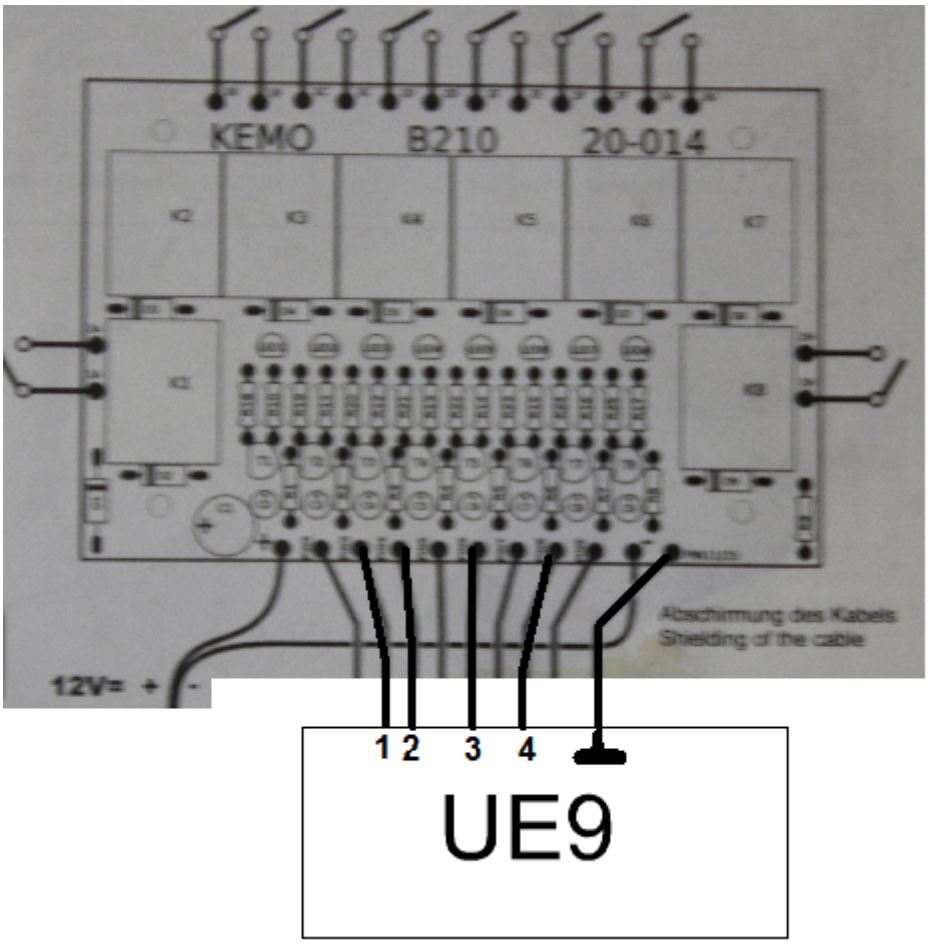


Abbildung 3.5: Schaltplan der Relais

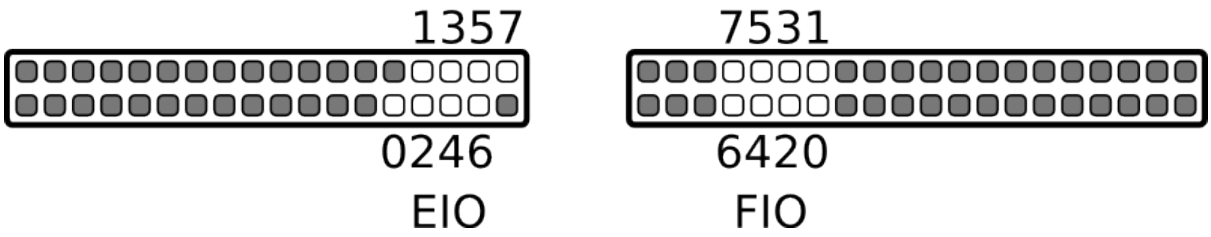
3.6 UE9

Das UE9 ist ein universell einsetzbares Mess-Labor, welches die Schnittstelle zur *Software* (Seite 10) darstellt und so ermöglicht, Daten auszulesen und verschiedene Vorgänge zu steuern. Wir verwendeten es zunächst nur für die Zeitnahme durch die Start/Ziel-Lichtschranken, später kamen die Steuerung der Relais für die *Streckenfreischaltung* (Seite 18) und die *Ampel-Schaltung* (Seite 15) hinzu.

Gesteuert werden die einzelnen Elemente durch Low bzw. High Pegel, welche verstärkt werden, um das dazugehörige Bauelement anzusteuern. Außerdem wird erfasst, ob gerade eine Lichtschranke aktiviert ist, also ein Auto die Start- / Ziellinie überquert.

Auf der Website von Meilhaus ist ein *Datenblatt des UE9 <http://www.meilhaus.de/index.php?id=26&L=0&user_produkte[PR]=85* erhältlich.

3.6.1 Schnittstellen



Ampel:

- Rot 1: FIO0
- Rot 2: FIO1
- Rot 3: FIO2
- Grün: FIO3

Streckenfreigabe (Seite 18):

- Strecke 1: EIO4
- Strecke 2: EIO5
- Strecke 3: EIO6
- Strecke 4: EIO7

Sensoren:

- 1: EIO0
- 2: EIO1
- 3: EIO2
- 4: EIO3

3.6.2 Probleme

Das UE9 selbst bereitete uns wenig Probleme, da es fehlerfrei und genau arbeitete, jedoch mussten klare Absprachen mit dem Programmierer getroffen werden, damit auch alles so funktioniert wie es soll. Meistens ging es um die richtige Pinbelegung.

3.7 Arduino

Das Arduino ist das „Gehirn“ unseres selbstgesteuerten Autos. Es ist mit einem 32Kb Flash-Speicher und einem 16MHz Prozessor ausgestattet. Das *Programm* (Seite 12) ist vollständig auf dem Chip gespeichert. Es benötigt lediglich eine Stromversorgung und ist dann sofort einsatzbereit. Sämtliche Lichtschranken, die zum selbständigen Fahren benötigt werden, sind an das Arduino angeschlossen. Auf Port 11 liegt ein PWM (Pulsweitenmodulation) Signal an, welches je nach Position des Autos auf der Strecke einen unterschiedlichen Aussteuerungsgrad hat.

Der Grund für die Nutzung des PWM-Signal ist die Einfachheit, da das Arduino einen eingebauten Modulator besitzt. Wir versuchten anfangs durch Verstärkerschaltungen das schwache Signal des Arduino zu verstärken, um die Autos anzutreiben. Obwohl eine geeignete Schaltung entworfen wurde, war es leider nicht möglich diese umzusetzen. Der Grund dafür waren die Stromquellen, die nicht fdafür geeignet waren eine Operationsverstärker-Schaltung zu betreiben, da die Carrera Trafos einen starken Spannungsabfall bei einer Last hatten.

Daten des Arduino Uno (http://arduino.cc/en/Main/ArduinoBoardUno)

3.7.1 PWM

Hier zunächst ein Beispiel um zu verdeutlichen wie wir das PWM nutzen:

```
t1 / T = 0.25 = 25%
```

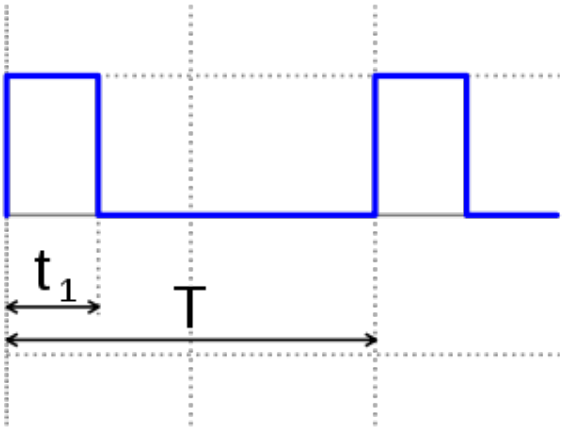


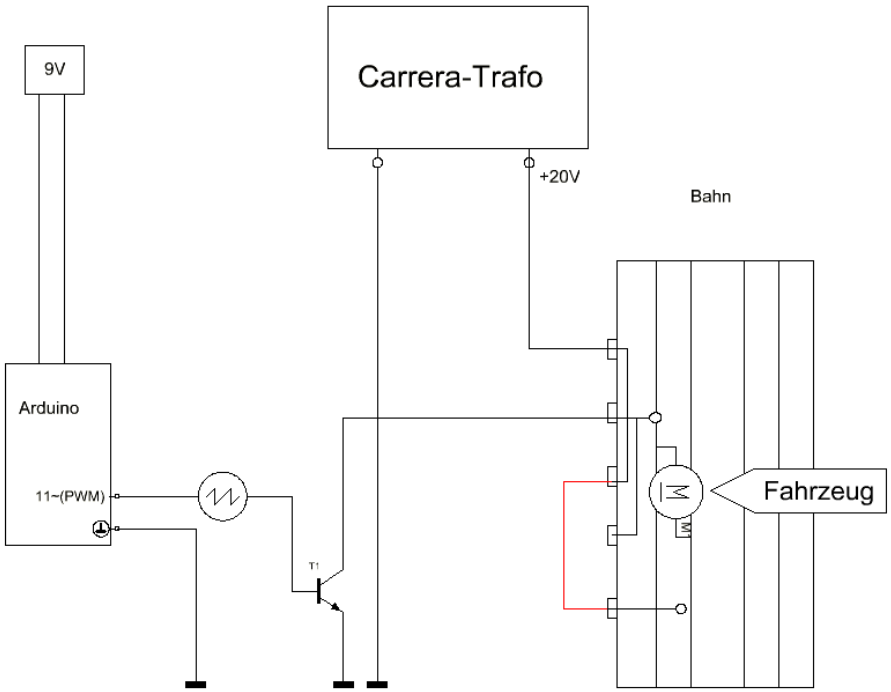
Abbildung 3.6: Quelle: http://de.wikipedia.org/w/index.php?title=Datei:Pulse_wide_wave.svg&filetimestamp=20080309132226

Durch den unterschiedlichen Aussteuerungsgrad des PWM-Signals fährt auch das Auto schneller. Je größer t1 ist desto stärker beschleunigt es. So ist es auch möglich zu bremsen. In den normalen Carrera Fahrtenregeler ist ein Bremse eingebaut, die, wenn man den Schieber gehen lässt, den Motor kurzschließt. Somit kann Strom fließen, was die Energie des Motors abbaut. Da dies durch die Arduino-Steuerung nicht möglich ist, geben wir einfach ein PWM-Signal mit einem sehr kleinen t1 an und können somit die Geschwindigkeit schwach vermindern. Um stärker abbremsen zu können müssten wir den Motor kurzschließen, wozu wir allerdings noch keine passende Möglichkeit gefunden haben.

Unten abgebildet ist die Schaltung, welche das PWM-Signal verstärkt. Wir verändern die Spannung die vom Trafo kommt. Somit brauchen wir keinen Schieberegler mit dem das Auto normalerweise gesteuert wird, sodass diese überbrückt werden müssen (rote Linie).

Probleme

Das größte Problem bereitet uns noch immer das Bremsen gefolgt von Schwierigkeiten bei den Lichtschranken, welche sich durch eine optimierte Verkabelung und der Reparatur der Lichtschranken durch Herr Althen lösten.


9V
Carrera-Trafo
+20V
Bahn
Arduino
11~(PWM)
T1
Fahrzeug

Carrera-Strecke

4.1 Grundzüge / Fundament

Hierbei ging es schlicht und einfach um die Platte, auf der sich die Strecke befindet. Diese mussten gedreht, so angeordnet werden, dass die Strecke mit dem Fundament optimal übereinstimmte und untereinander befestigt werden. Es wurden Aussparungen vorgenommen, um die Kabel unterhalb der Platte zu verlegen.

4.2 Streckenplanung / Streckenbau

Die bereits vorhandene Stecke musst aufgrund ihrer Größe und ihrer Kurvenanzahl leicht verändert werden. Dabei musste man jedoch beachten, dass alle Bahnen eine möglichst gleiche Länge und Fairness bieten sollten, da man beispielsweise eine Außenkurve schneller als eine Innenkurve fahren kann. Zielsetzung war es eine neue Strecke zu schaffen, welche eine ausgewogene Anforderung an die Fahrer stellt.

Die Strecke musste fair für alle Bahnen bleiben. Außerdem war das Programm, mit welchem man die Strecke anfertigen musste, nicht sehr einfach zu bedienen.

Pläne

Zu den weiteren Zielen gehört die Anbindung des Arduino an das Rennprogramm, sodass wir das computergesteuerte Fahrzeug definiert starten und stoppen können. Weitere Details und Überlegungen zu diesem Thema finden sich im Kapitel *Autopilot / KI* (Seite 12). Außerdem werden wir versuchen, ein geeignetes Bremssystem zu entwickeln. Unser erster Ansatz ist es, den Motor umzupolen, was eine Bremswirkung zur Folge haben sollte.

Erste Ordnung schaffen. Videos drehen, die unseren YouTube-Channel (http://www.youtube.com/Bg13Racingteam) füllen sollen, damit die Allgemeinheit von unserer Arbeit profitieren kann.

Stichwortverzeichnis

V