Semesterprojekt

Rocky Hockey

Sommersemester 2019

betreuender Professor:
Prof. Dr. Arnulf Deinzer

Team

Name	Kontakt	Matrikelnummer
Oliver Berger	oliver.berger@stud.hs-kempten.de	341272
Gil De La Cruz	gil.l.cruz@stud.hs-kempten.de	342383
Arlind Gashi	arlind.gashi@stud.hs-kempten.de	338212
Philipp Möslang	philipp.moeslang@stud.hs-kempten.de	333030
Alexander Maier	alexander.b.maier@stud.hs-kempten.de	354709
Andreas Olf	andreas.olf@stud.hs-kempten.de	306809
Daniel Marko	daniel.g.marko@stud.hs-kempten.de	352135

Projekt Ist-Zustand

Das Projekt Rocky Hockey ist ein DV-Projekt das unter der Leitung von Professor Arnulf Deinzer für Studierende im 6. Semester des Studiengangs Informatik angeboten wird.
Ziel des Projekts ist es einen Roboter zu Entwickeln, der gegen einen Menschen in einem Airhockey „Match“ antreten kann.

Der Roboter verwendet eine über dem Tisch angebrachte Playstation 3 Kamera, um die Bewegung des Spiel-Pucks zu erkennen. Abgestimmt auf die Bewegungsrichtung des Pucks wird der Schläger des Roboters mithilfe zweier Schrittmotoren in die Richtige Position gebracht, um den Puck zu stoppen und zurück zum Menschlichen Spieler zu schlagen.

Zusätzlich ist noch ein Bildschirm am Aufbau vorhanden über den ein Spiel gestartet werden kann. Dieses Spiel zählt den Punktestand mit, der durch Lichtschranken in den Toren des Spielfelds ermittelt wird. Als Zusatz ist noch ein Lautsprecher eingebaut, der entsprechende Informationen zum Spiel ausgibt.

Aktuelle Hardware

Raspberry Pi:
Für die Darstellung der GUI und zur Ermittlung der erzielten Tore mittels Lichtschranken.
Playstation Eye Kamera:
Die Kamera zum Motion Tracking des Pucks auf dem Spielfeld.

Nema 17 und Nema 23 Stepper Motoren:
Um den Schläger des Roboters an die Berechnete Position zu bewegen:
Arduino Uno mit CNC Shield incl. Stepper Treiber:

Soundboard + Lautsprecher:
Zum Abspielen der Spielsounds
5“ Touch Display:
Darstellung der GUI für Punktestand und Start des Spiels
90W ebm-papst Industrie Lüfter:
Sorgt für den Luftstrom, damit der AirHockey Puck auf dem Spielfeld gleitet

GIT Repository

In der Planungsphase zu Beginn des Projekts wurde durch das Team ein Versionsverwaltungssystem in Form eines GitLab-Repository aufgesetzt.

Das GitLab wurde auf einem Debian-Homeserver mit einer i5 6400 CPU und 16 GB DDR4 RAM in einem Docker-Container aufgesetzt. Die Daten wurden Redundant auf zwei 4TB HDD Festplatten im Raid-1-Verbund auf einem NAS im selben Netzwerk gespeichert. Die Datenübertragung wird mittels iSCSI vorgenommen. Die Übertragung zu Nutzern erfolgt mittels einer 1-GBit Internet-Verbindung unter Verwendung des SSH-Netzwerkprotokolls. Der Zugriff durch den Nutzer kann sowohl über den Web-Browser als auch über das Terminal des Betriebssystems mit den standard Git-Befehlen erfolgen.

Über dieses Repository wurden im Verlaufe des Projekts sämtliche Planungselemente wie Codeverwaltung, Meilensteinverwaltung, etc. ausgeführt. Außerdem wurden dort auch die zuvor erstellten Tickets, nach jedem erfolgreichen Abschluss als “Erledigt” markiert. So konnten wir uns an eine Liste halten anstatt alles zufällig zu machen. Dies hat dazu beigetragen, dass das Team viel organisierter an die offenen Aufgaben herangegangen ist.

Arbeitsweise/Projektmanagement

Zu Beginn des Projekt wurde im Team zusammen ein Katalog ausgearbeitet, der die zu erledigenden Aufgaben enthält. Diese Aufgaben wurden Teilweise in kleinen Gruppen innerhalb des Teams verteilt. Nach einer anfänglichen Analyse- und Bearbeitungsphase, stellte sich zeitnah heraus, dass ein großteil der Aufgaben stark miteinander verbunden und abhängig voneinander ist. Dieser Umstand führte dazu, dass aus den Anfänglich gebildeten Gruppen im Laufe der Zeit eine große Gruppe wurde und somit fast alle Aufgaben im Kollektiv als Gruppe bearbeitet wurden.
Diese Arbeitsweise führte dazu, dass die vorher definierten Aufgaben nach kurzer Zeit in vier große Meilensteine verpackt wurden, die dann in der großen Gruppe gemeinsam erfüllt wurden. Diese vier Meilensteine werden im Folgenden genannt.

- System in Betrieb nehmen
- Hardwareumbau / 3D-Teile drucken / Code-Optimierung
- Neues System in Betrieb nehmen
- System optimieren und testen

Anfänglich hat sich die Gruppe einmal die Woche, meistens Freitags, da an diesem Tag jeder Zeit zur Verfügung hatte, getroffen, um am Projekt zu arbeiten und zu besprechen, welcher Fortschritt in der vergangen Woche erfolgt ist. Einige Wochen vor Projektende wurde die Frequenz der Treffen erhöht und das Team hat sich somit wöchentlich zwei bis drei Mal, Freitags, Samstags und Sonntags, getroffen, um die Bearbeitung des Projekts voranzutreiben bzw. abzuschließen.

Da bei jedem Treffen fast immer alle Mitglieder des Teams da waren, konnte man immer Ideen und Vorschläge von einzelnen Mitgliedern in der Gruppe schnell analysieren und entweder als gut oder schlecht einstufen, ohne unnötig Zeit zu verlieren, indem man auf die Meinung der fehlenden Mitglieder warten müsste.

CODING - CODE Refactoring

In der ersten Phase des Projekts mussten wir uns mit dem bereits vorhandenen Code der vorherigen Projektgruppen vertraut machen. Dies hat anfänglich eine gewisse Zeit in Anspruch genommen, da bis auf ein Teammitglied niemand aus dem aktuellen Team vorher wesentliche Erfahrungen mit C# gesammelt hatte.
Doch dank der betreffenden Person, die dem Rest den Code erklärt hat, wurde die erste Hürde somit relativ schnell überwunden und jeder konnte sich mit dem vorhandenen Code entsprechend auseinandersetzen.

GUI in Java

Das Graphical User Interface mit Soundausgabe wurde in einem Relaunch in der Programmiersprache Java programmiert. Hierbei wurde ein rein Objekt-Orientierter Ansatz verfolgt.

Unter Anderem wurden für die Problemstellungen Programmier-Muster, wie etwa das Singleton-Pattern und Model-View-Controller-Pattern, verwendet. Das Programm selbst ist in 4 Threads aufgeteilt, um einen reibungslosen Ablauf mit hoher Frames-per-Second-Rate zu erzielen.

Die Ansteuerung der Hardware General Purpose Input/Output wurde auch in Java umgesetzt, um eine ungehinderte Kommunikation zwischen der GUI und der Hardware zu gewährleisten. Hierfür wurde eine Eigenschaft des Betriebssystems Debian, welches auf dem Raspberry-Pi verwendet wird, genutzt. Die GPIOs werden nämlich als System-Files hinterlegt und können somit gelesen und manipuliert werden. Dies ermöglichte die Implementation einer eigenen Java-Library.

Die Software wurde unter verwendung des JDK11 (Java Development Kit 11) in eine Binary compiliert und auf einem Raspberry Pi unter der Debian Distribution “Raspberry-Stretch” in der Java Virtual Machine ausgeführt. Der Java-Code selbst wurde in der open-Source IDE “Eclipse” erstellt und umfasst etwa 1.300 Source-Code Zeilen.

Hardware Änderung

Eine wesentliche Aufgabe des Projekts bestand darin neue Hardware Teile zu beschaffen und in Betrieb zu nehmen. Anfangs haben wir im Team viele Teile auf unsere Einkaufsliste geschrieben und diese dann auch an die zuständigen Personen weitergeleitet, die sich wie vereinbart um die Beschaffung kümmern.

Leider wurden nicht alle unsere Vorschläge vom Haushaltsamt genehmigt. Letztendlich haben wir von unseren angefragten Objekten nur einen Teil gestellt bekommen. Darunter zählt der Zahnriemen, welchen wir um die Motoren und die Schlägerbasis montiert haben, da auch hier die alten Zahnriemen langsam an Stabilität eingebüßt haben und die weitere Verwendung als nicht sinnvoll erschien.
Aus diesem Grund mussten wir ein wenig improvisieren und die restlichen Teile in Baumärkten und Elektrofachgeschäften besorgen. Aufgeteilt in zwei Teams haben wir uns auf den Weg gemacht um die Einkäufe zu erledigen.

Team 1 ist zum Elektrogeschäft gegangen, Team 2 zum Baumarkt. Dort haben die Teams ihre jeweiligen Einkaufslisten versucht abzuhaken. Letzten Endes konnten wir abgesehen von einem Netzteil und diversen Kleinteilen, fast alles besorgen. Die Kassenzettel wurden dann an die zuständige Person übergeben und das Team hat die Kosten erstattet bekommen.

Abgesehen von der besseren Optik, die der Tisch erhalten hat, wurden auch mehrere wichtige Features dazu gewonnen. Der fest verbaute Schläger ist nun in der Höhe verstellbar und kann leicht rein- und rausgedreht werden. Des Weiteren kann man die sechs Seitenteile die am Tisch angebracht sind separat rausschrauben und Änderungen an diesen vornehmen, falls das Gerüst nicht in der Waage steht. Die einzelnen Bauteile sind aufgrund ihrer Modularität schnell und einfach zu ersetzen. Des Weiteren sind schnelle Hardware-Anpassungen lediglich in ihrer Druckzeit begrenzt. Dies führt dazu, dass der Tisch universell abänderbar ist.

Dank der neuen 3D-gedruckten Teile haben die Motoren nun viel weniger Probleme die schwere Last der Puck-Schläger Konstruktion in Richtung der X- und Y-Achsen zu bewegen. Das verringert die Gefahr, dass die Motoren überhitzen, ausfallen und ersetzt werden müssen.

Unnötige Bauteile, die einen hohen Energiebedarf und keinen nennenswerten Nutzen haben, wurden entfernt. Des Weiteren wurde die Beschaltung der Lichtschranken optimiert und diverse Leistungs-Abnehmer waren nicht mehr nötig.

3D-Drucke

Da sich der Tisch schon in der dritten Generation befindet, wurden mit der Zeit die vorhandenen Holzteile „Marode“. Dadurch fehlte die nötige Stabilität, die der Schlitten benötigt. Die Keilriemen überspringen mehrmals ihre Zahnräder, die Stangen verbiegen sich und viele weitere Folgen.

Aus diesem Grunde entschlossen wir uns die wichtigen Teile neu herzustellen. Dafür bieten sich 3D-gedruckte Modelle an. Da ein Mitglied schon viel Erfahrung mit 3D-Drucker gesammelt hat und selbst im Besitz eines 3D-Druckers ist, haben wir uns dafür entschieden, dass diese Person den 3D-Druck leitet.

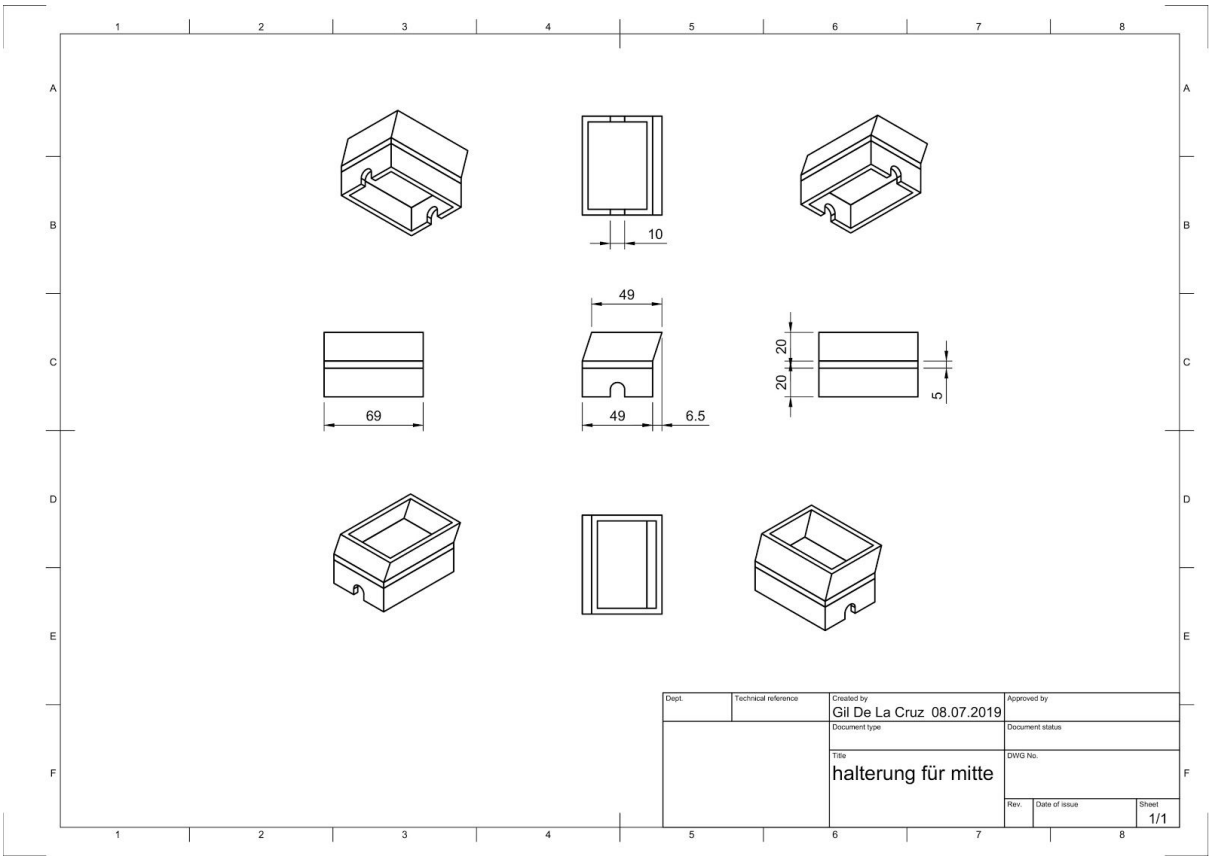
Alle Teile wurden dank der kostenlosen Studentenversion von Fusion 360 der Firma Autodesk modelliert. Alle Holzteile wurden zuerst nachmodelliert und danach verbessert. Durch Ideen aller Beteiligten wurden einige Verbesserungen eingeführt. Die Modellierung nahm sehr viel Zeit in Anspruch. Alle Teile, wie Stangen, Lager, Schlitten, Schrauben, Muttern und viele weitere, mussten genau passen. Da der 3D-Drucker auf 0,2mm genau druckt, mussten alle Gegenstände zuvor präzise abgemessen werden, um einen tagelangen Druck nicht in die Tonne werfen zu müssen. Leider wurde durch den späten Einkauf im Baumarkt diese Phase ausgebremst.

Da die 3D-gedruckten Teile nur mit einem kleinen prozentualen Anteil mit Material gefüllt sind, können nicht wie bei Holz einfach Schrauben reingedreht werden. Diese würden nach kurzer Zeit rausfallen. Dadurch wurde die gesamte Konstruktion mit Gewindestangen realisiert. An jedem Ende der Gewindestangen wurde eine Mutter festgeschraubt und somit eine stabile und feste Verbindung hergestellt. Bei dieser Art der Modellierung musste an jede einzelne Gewindestange und jede Mutter gedacht werden, um einen tadellosen Zusammenbau bereitzustellen.

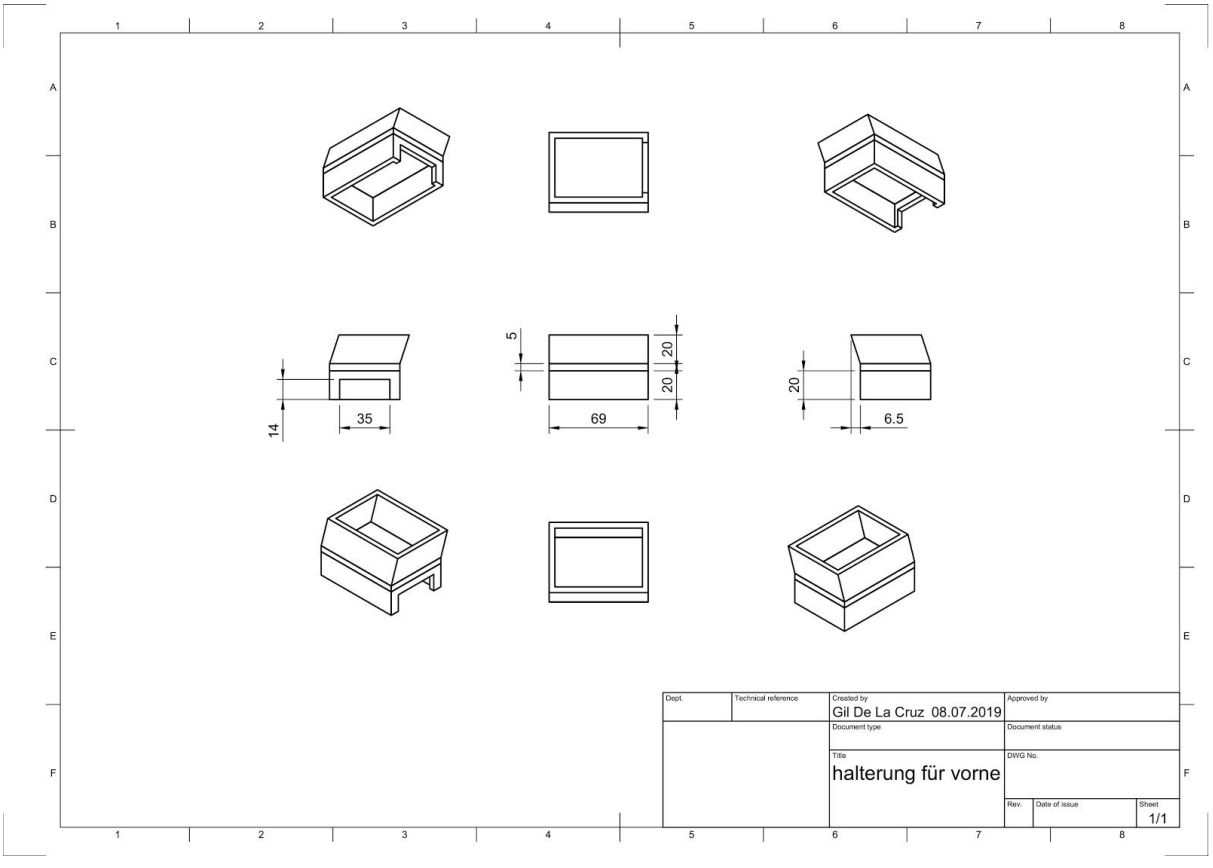
Durch die Software Ultimaker Cura 4 konnten alle 3D-Modelle problemlos und optimal in das 3D-Druckformat gcode übersetzt werden. Der Druck selbst dauerte mehrere Tage lang und verbrauchte mehr als zwei Kilogramm an Filament trotz des neuen Anycubic Mega-S Druckers.

Der Zusammenbau selbst ging durch die penible Planung ohne weitere Hindernisse vonstatten. Da ein Teammitglied glücklicherweise viel Werkzeug besaß, konnten wir an einem Tag durch Flexen, Bohren und Schleifen den Tisch zusammenbauen. Alle Gewindestangen wurden für jedes einzelne Loch auf seine benötigte Länge geflext und geschliffen, um jedes Bauteil optimal zu halten. Auch alle 8mm Stangen mussten auf die perfekte länge zugeschnitten werden, um zwischen den 3D-Teilen eingeklemmt zu werden. Durch den guten Zusammenhalt und der guten Aufgabenverteilung wurde der Tisch schnell zusammengebaut.

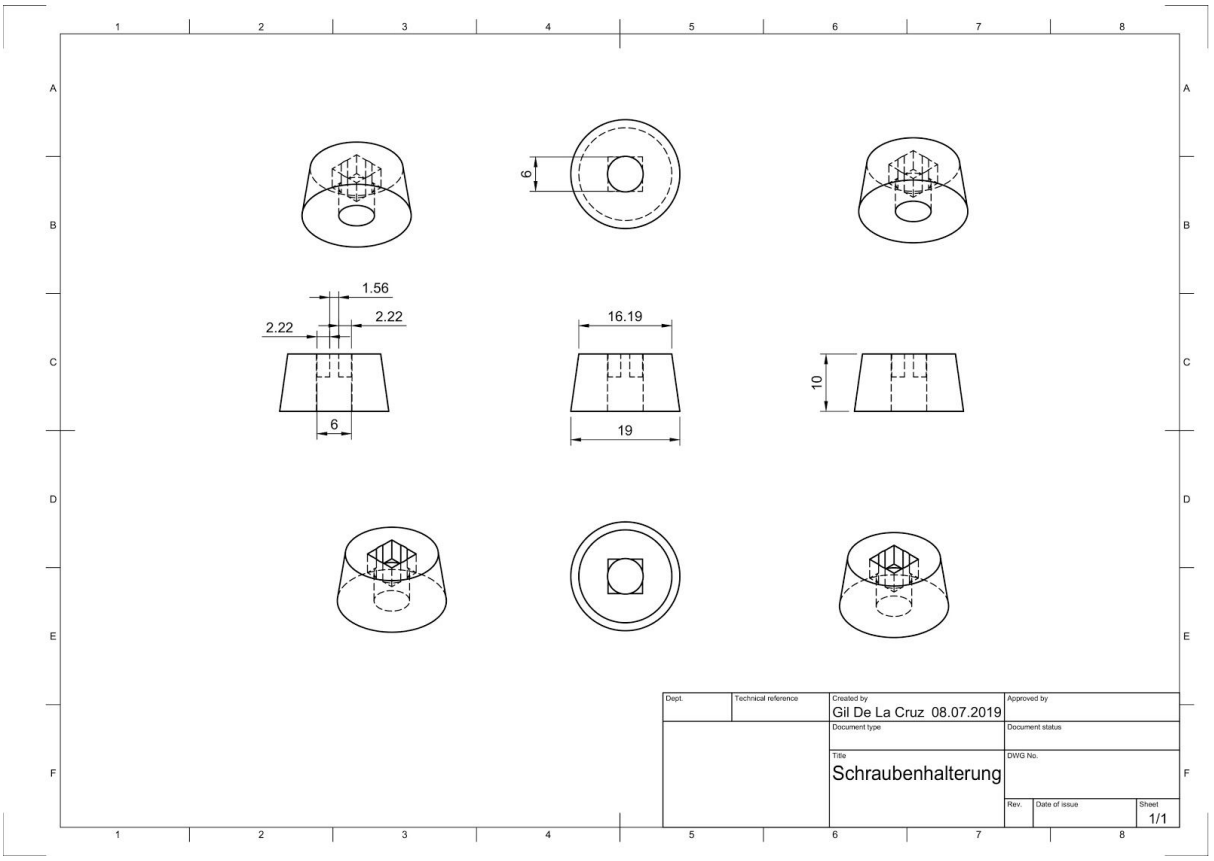
Im Folgenden ist ein Ausschnitt der modellierten 3D-Teile zu sehen.



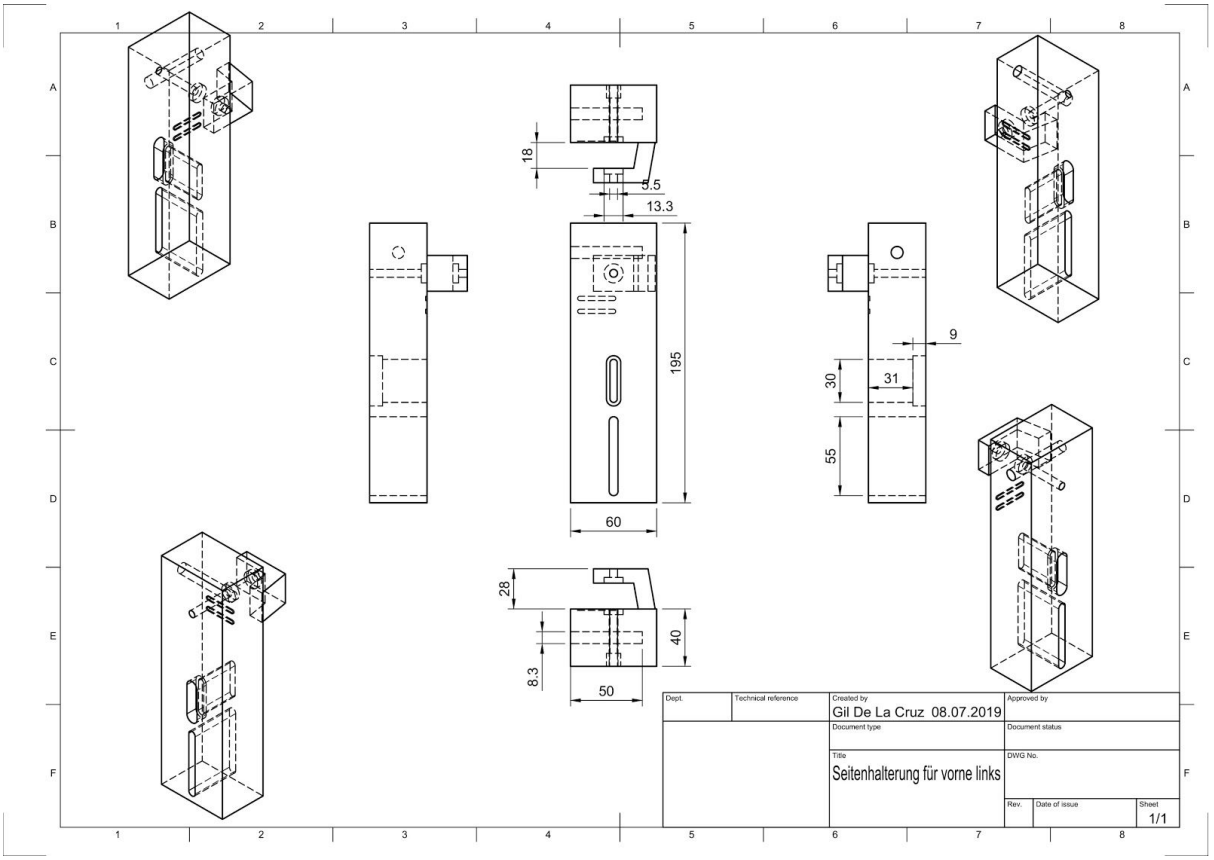
Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: halterung für mitte	DWG No.
			Rev. / Date of issue / Sheet: 1/1



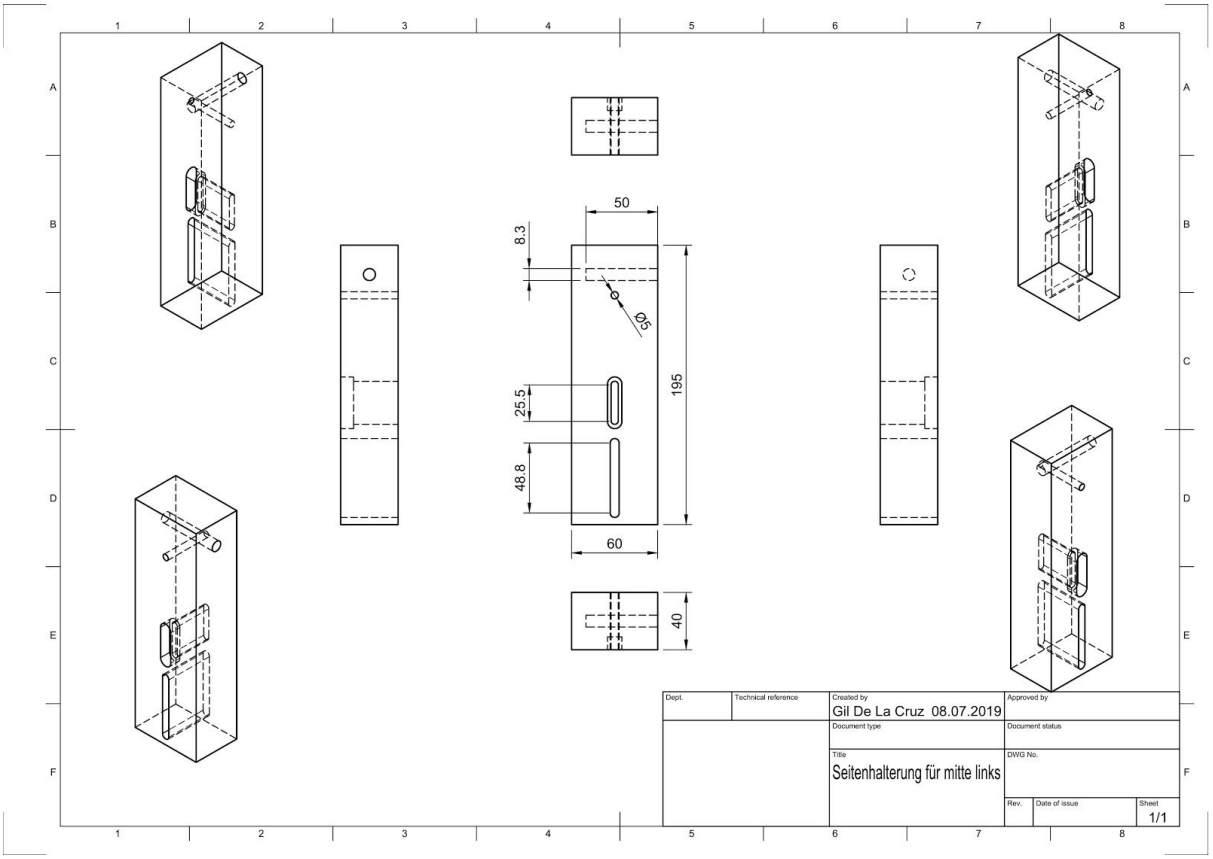
Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: halterung für vorne	DWG No.
			Rev. / Date of issue / Sheet: 1/1



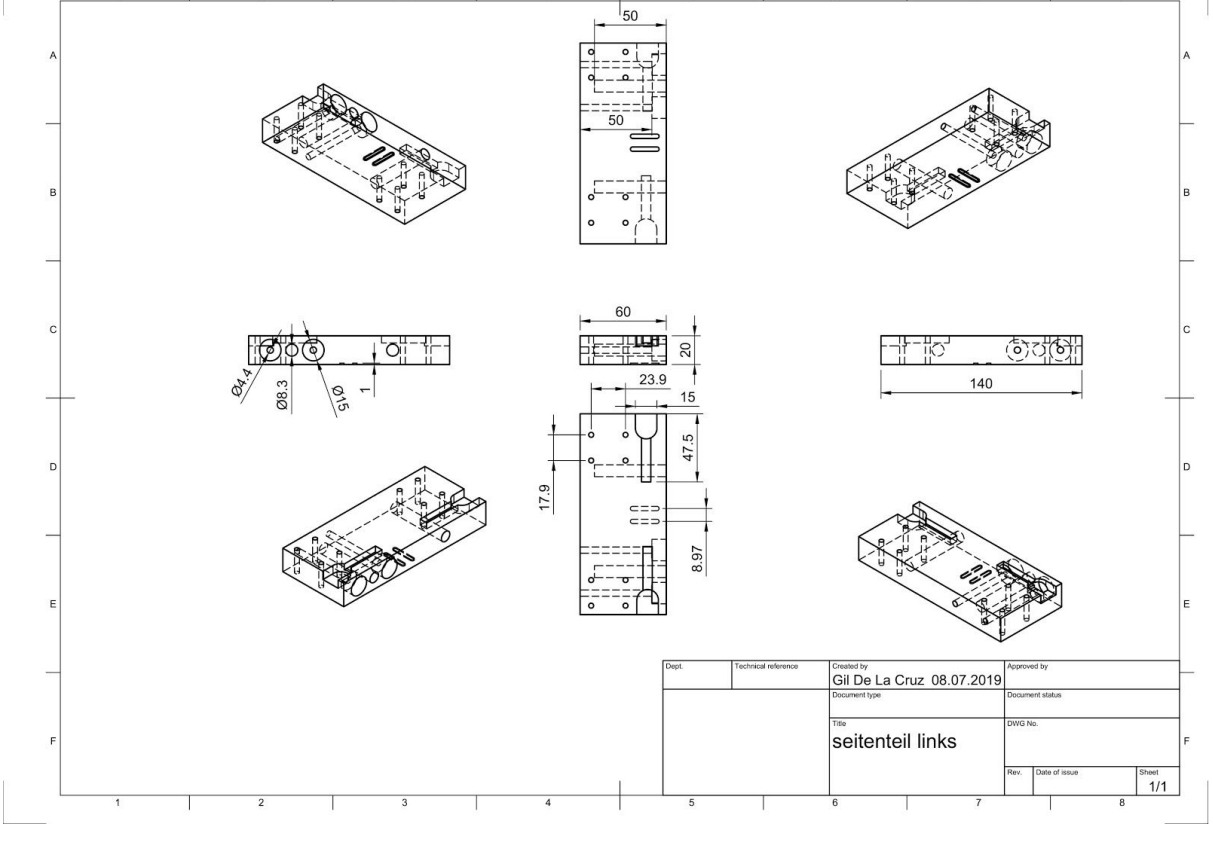
Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: Schraubenhalterung	DWG No.
			Rev. / Date of issue / Sheet: 1/1



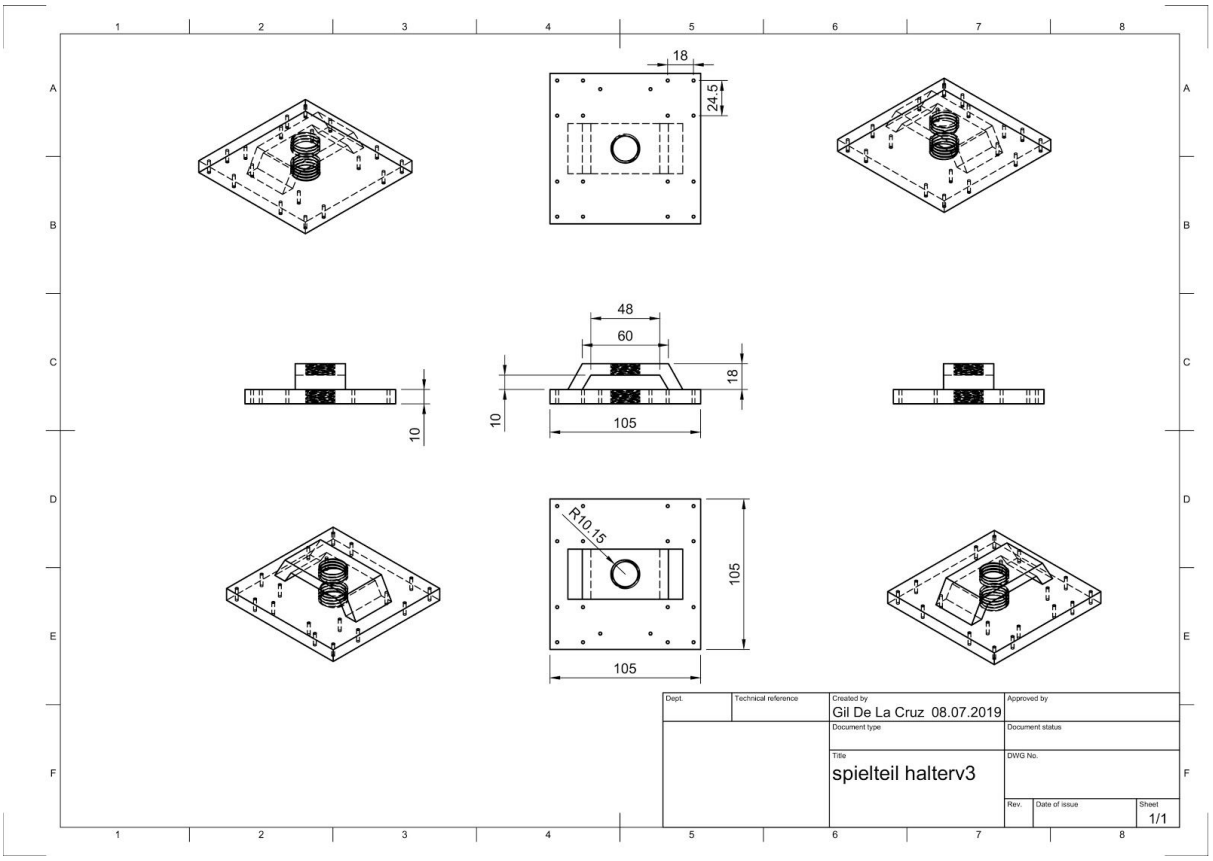
Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: Seitenhalterung für vorne links	DWG No.
			Rev. / Date of issue / Sheet: 1/1



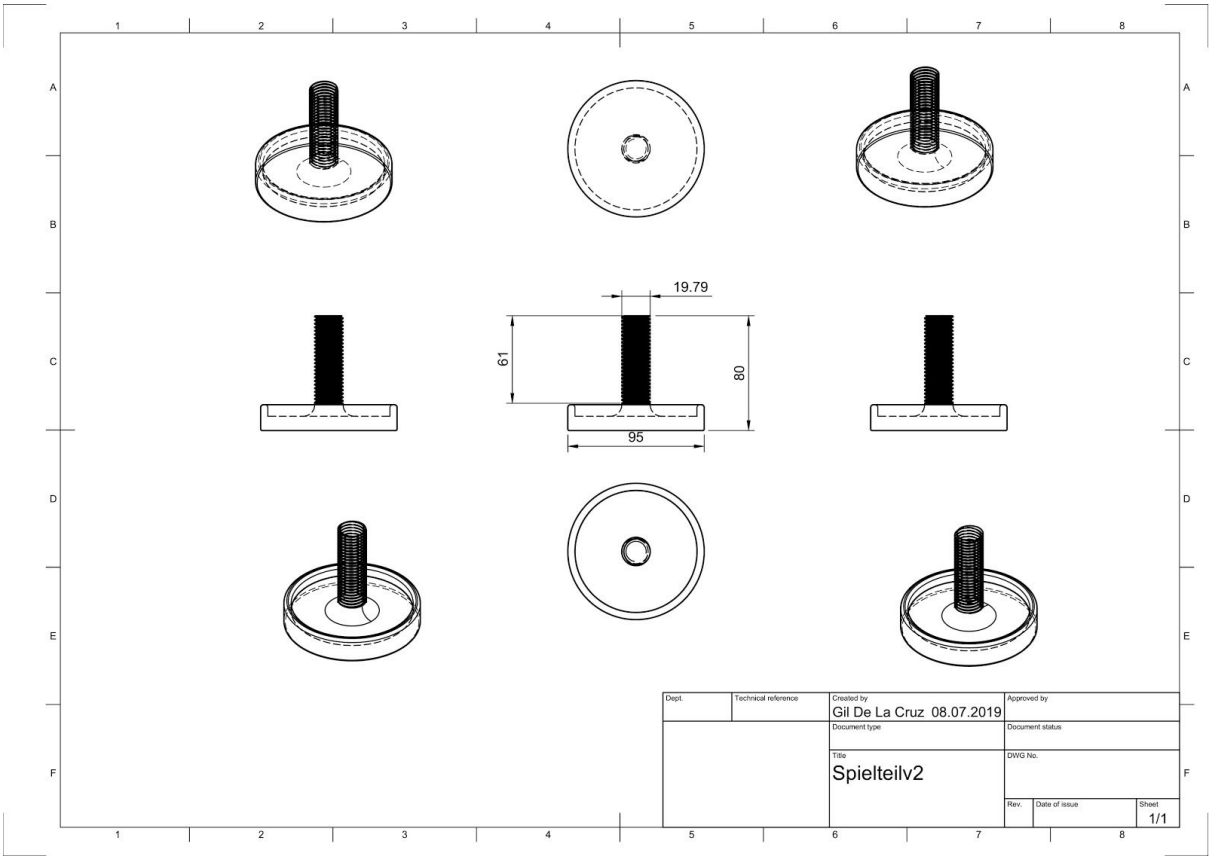
Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: Seitenhalterung für mitte links	DWG No.
			Rev. / Date of issue / Sheet: 1/1



Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: seitenteil links	DWG No.
			Rev. / Date of issue / Sheet: 1/1



Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: spielteil halterv3	DWG No.
			Rev. / Date of issue / Sheet: 1/1



Dept.	Technical reference	Created by	Approved by
		Gil De La Cruz 08.07.2019	
		Document type	Document status
		Title: Spielteilv2	DWG No.
			Rev. / Date of issue / Sheet: 1/1

Ansatz mit Nvidia Jetson TX1 development Kit

Seit der ersten Projektgruppe stand auch ein Nvidia-Board zur Verfügung. Das Jetson TX1. Dieses ist ausgerüstet mit einem Quad-Core ARM Prozessor, sowie einer Nvidia Maxwell GPU, welche 256 Cuda-Cores hat. Das Board ist für Deep Learning und grafische Berechnungen gedacht. Außerdem ist es mit mehreren Ausgängen ausgestattet. Dazu sind USB-Anschlüsse verfügbar, auch kann man via LAN oder WLAN das Board in ein Netzwerk verbinden. Von Haus aus ist Ubuntu 16 installiert.

Die vorherigen Gruppen hatten bereits mit dem Jetson TX1 experimentiert und auch verschiedene Profile eingerichtet. Das Board musste hauptsächlich neu aufgesetzt werden, da das Passwort für den Superuser nicht bekannt war.

Dazu kann man, wenn man einen Nvidia Developer Account erstellt, über die Nvidia Webseite das passende Programm herunterladen, um vom Host aus den Flash zu starten (https://developer.nvidia.com/embedded/downloads/archive). Herunterzuladen ist eine Jetpack-Version. Den Host selbst stellte eine Ubuntu 16 Distribution dar. Mit Ubuntu 18 wollte das Jetpack-Programm sich nicht ausführen lassen. Des Weiteren ist zu beachten, falls man auf seinem Rechner Windows installiert hat und in einer VirtualMachine Ubuntu laufen lässt, dann darf man höchstens Jetpack 3.0 installieren. Bei neueren Version (3.1 und höher) tritt ein Fehler während des Beschreibens des Boards auf.

Generell kann man sich für die Installation von Ubuntu auf dem Jetson TX1 an diesen Link halten: http://www.slothparadise.com/setup-cuda-7-0-nvidia-jetson-tx1-jetpack-detailed/

Falls beim Ausführen des Jetpack-Programms der Fehler auftritt, dass die Manifest-File fehlerhaft sein sollte, kann man das Programm mit folgendem Zusatz starten:

```
NV_DEVTOOLS_FORBID_MULTIPLE_DOWNLOAD_THREADS=1 ./JetPack-L4T-3.2-linux-x64_b196.run
```

(https://devtalk.nvidia.com/default/topic/1033069/jetpack-installation-the-manifest-file-is-broken-/).

Unglücklicherweise hat der Jetson TX1 nur 15GB internen Speicher. Allein durch das OS selbst sowie die Nvidia Software, die beim Flashen mitinstalliert wird, werden ca 10GB des Speichers belegt (unter Umständen weniger, wenn man manche Pakete auslässt, was trotzdem nicht ausreicht).
https://devtalk.nvidia.com/default/topic/1021520/the-space-of-jetson-tx1-is-not-enough/

Als Lösung dafür kann man das alles auf eine SD-Karte oder eine SSD schreiben lassen, die mindestens 64GB hat.
https://www.jetsonhacks.com/2017/01/26/run-jetson-tx1-sd-card/
https://www.jetsonhacks.com/2017/01/28/install-samsung-ssd-on-nvidia-jetson-tx1/

Wenn der Flashvorgang über eine VM vorgenommen wird, kann es dennoch passieren, dass es einen Fehler gibt und nicht funktioniert. Es wirkte, dass jedes zweite Mal ein Fehler aufgetreten ist. Deshalb mindestens ein weiteres Mal probieren, falls es nicht klappen sollte. Um auch auf den Jetson TX1 Remote Zugriff zu erhalten, müssen das Board und der zugreifende Rechner im gleichen Netzwerk angemeldet sein. Dann kann man im Board über `ifconfig` die IP-Adresse auslesen. Mit folgendem Befehl greift man dann auf das Board zu:

```
ssh ubuntu@192.168.1.3
```

Die IP-Adresse stimmt natürlich nicht und muss angepasst werden. Danach wird man aufgefordert das root-Passwort einzugeben, welches bei Standardeinstellungen “ubuntu” ist (https://devtalk.nvidia.com/default/topic/1036488/jetson-tx1/remote-control-of-the-jetson-tx1-without-using-an-hdmi-input/).

Zusätzlich kann man beispielsweise einen DHCP-Server einrichten um einen erleichterten Zugriff zu erhalten, falls man zukünftig das Board nicht mehr mit einem grafischen Output betreiben möchte (https://elinux.org/Jetson/Remote_Access).

Als nächsten Schritt wurde das Mono Framework installiert, welches das Kompilieren von C# Source-Code ermöglicht. Dazu empfiehlt es sich an diesen Guide zu halten:

https://www.mono-project.com/download/stable/#download-lin

Am besten installiert man das Package `mono-complete`, um später nicht auf unvorhergesehene Fehler zu treffen. Außerdem wird auch die IDE MonoDevelop mit installiert. Mit dieser kann man, im Falle von Fehlern beim Kompilieren oder Ausführen, das Debugging durchführen.

Wenn die Installation abgeschlossen ist, kann man das Projekt auf den Jetson TX1 aufspielen und kompilieren lassen. Auch Visual Studio Projekte lassen sich dort kompilieren.

Mit `msbuild visualstuidoproject.sln` kann man das Projekt kompilieren.
Ausgeführt wird es mit `mono visualstudioproject.exe`.

https://stackoverflow.com/questions/8264323/how-to-compile-a-visual-studio-c-sharp-project-with-mono

https://stackoverflow.com/questions/54790/is-it-possible-to-build-msbuild-files-visual-studio-sln-from-the-command-line

Nachdem ein paar Fehler bei den Projektdateien behoben wurden, konnte alles fertig kompiliert und das Hauptprogramm auch erfolgreich ausgeführt werden. Die Ausführung der Calibration-Software hat leider nicht funktioniert und hielt direkt mit einem Fehler an. Die Fehlermeldung lautete `System.BadImageFormatException`. Leider konnte der Fehler nicht behoben werden. Er hat etwas mit der Library EmguCV zu tun. Möglicherweise hängt der Fehler damit zusammen, dass die Library in einem 32-Bit System gebaut wurde, aber versucht wird, diese in einem 64-Bit System auszuführen. Leider waren die Kenntnisse rund um EmguCV in Kombination mit C# auf Ubuntu nicht ausreichend um den Fehler beheben zu können.

Probleme

Erfassung Ist-Stand vorgänger Team:
Da es üblich ist, dass das Vorgänger-Team den Nachfolgern erklärt und demonstriert welche Dinge sie das letzte Semester alles getan haben, war dies nun auch bei uns der Fall. Leider gab es ein paar zeitliche Schwierigkeiten mit unseren und deren Mitgliedern, weshalb sich die ganze Sache um ein paar Wochen gezogen hat. Beim ersten Treffen hat es widerwillen ein gutes Stück gedauert bis das System wieder funktioniert hat und der Tisch am laufen war.

Kamera Kalibrierung:
Die Position und der Erfassungswinkel der Kamera machen die Kalibrierung um einiges schwieriger als man es meinen mag. Grund hierfür ist, das die Kamera leider nur die Hälfte des Tisches erfassen kann. Des Weiteren ist die Befestigung nicht immer so akkurat und präzise auf die Spielfläche gerichtet. Aus diesen Gründen muss man in der Kalibrationssoftware das Bild so anpassen, dass die Spielfläche genau erfasst wird, ohne die Umgebung im Bild zu haben.

Holzplatte:
Durch die Form der Holzplatte, die nicht überall gleich hoch und somit uneben ist, kann es dazu führen, dass der Schläger des Roboters nicht überall auf der Platte aufliegt. Dies führt dazu, dass der Puck unter dem Schläger durchrutschen kann.

Nvidia Jetson:
Der Speicherplatz des Jetson TX1 Boards ist mit ca. 15 GB doch sehr beschränkt. Der dadurch entstehende Platzmangel erforderte es, das Board mehrmals neu aufzusetzen. Wobei zuerst versucht wurde Platz zu schaffen indem nicht benötigte Pakete entfernt oder gar nicht erst installiert werden, doch letztendlich hat nur ein größeres Speichermedium abhilfe geschaffen. Mit 64 GB SD-Kartenspeicher ist garantiert ausreichend Speicherplatz um alle Steuerprogramme auf dem Jetson auszuführen.

ToDo’s nächstes Projektteam

- Fangt am besten sofort an und baut einen kleineren Tisch, einfach die hälfte von dem jetzigen. Dann müsst ihr den Code nur darauf anpassen und habt kein stress mit dem Auf- und Abbau.

- Kamera Anpassung vornehmen oder vielleicht sogar mit zwei Kameras arbeiten, was natürlich ein wenig Arbeit erfordert.

- Sämtliche Kabelstrukturen, die sich im Tisch befinden, sollten erneuert und übersichtlicher verlegt werden.

- Falls ihr vorhabt die LED´s wieder anzubringen, dann beachtet das folgende nicht: Anstatt dem großen Gerüst, das man oben aufsetzen kann, bringt lieber nur einen metallstab an eine der lagen Seiten an, welcher die Kamera/s halten kann. So spart ihr euch das wackelige Gerüst oben und könnt die Kamera/s leichter positionieren.

- Ihr braucht DRINGEND einen E-Techniker im Team, sonst bekommt ihr echt Schwierigkeiten, wenn etwas ausfällt oder ihr in Sachen Elektrik etwas ändern wollt.

- Versucht eine Art Federvorrichtung zu bauen, damit der schläger nicht immer an einer höhe festsitzt, sondern sich frei hoch und runter bewegen kann, sodass er nicht auf der Platte hängen bleibt. Grund hierfür ist, dass die Platte sich verzogen hat und nun uneben ist.