MCB Mobil

Marco Stauber
Lars Peter

Ben17A

Hard & Softwaretechnik

3. Lehrjahr Projektarbeit

Inhalt

Idee

Unsere Idee war zunächst das MCB Board zu mobilisieren. Doch einfach ein paar Motoren daran anzubringen wäre zu einfach. Darum wollten wir das Fahrzeug und den Controller selber herstellen. Mit einer normalen WIFI Verbindung wäre es zu schwach, um ein Signal durch Wände zu schicken. Hier kommt LoRa ins Spiel. Wir nehmen hierfür das Long Range Protokoll und können somit grosse Distanzen erreichen mit den kosten von Datendurchsatz.

Die eigentliche Idee war es eine mobile Plattform für verschieden Sensoren zu bauen. Das Fahrzeug sollte zunächst genügend gross werden, damit man zusätzliche Module dazu bauen kann. Wie zum Beispiel: einen steuerbaren Arm, irgendwelche Sensoren oder es sogar mit anderen Projekten zu verbinden, wie wir es teilweise mit Laurins Bildschirm gemacht haben.
Mit einem Stepper-Motor in der Mitte könnte man den Deckel aufmachen und schliessen, um die Akkus auszutauschen. Oder eine Seilwinde hinten am Fahrzeug mit einem genügend starken Motor, um sich selbst aus der Klemme zu helfen.

Arbeitsaufteilung

Marco hat von Anfang an die Führung übernommen. Und Lars hat mitgeholfen. Die Arbeitsaufteilung war unter uns sehr einfach. Marco übernimmt das LoRa Modul und das Chassis während Lars den Controller entwickelt. Jeder von uns hat sein eigener Zeitplan erstellt. Und war selbst für seinen Teil verantwortlich. Wir haben uns gegenseitig geholfen und über unsere Ideen diskutiert. Das Chassis und das LoRa Modul waren eigentlich der grössere Teil des Projekts jedoch hat Marco sich das so ausgesucht was sich als sehr gut herausgestellt hat. Lars hat gegen den Schluss ein wenig die Motivation verloren während Marco erst richtig losgelegt hat.

Funktion

Das Fahrzeug wird mit einem Controller über LoRa gesteuert. Auf dem Fahrzeug wird andauernd die Akkuspannung gemessen und schaltet die Motoren ab zum Schutz der Zellen, sobald diese eine gewisse Spannung unterschreiten. Ein LED Balken in Form einer Batterie zeigt den aktuellen Status der Batterie. Als erstes wird 100-mal jede Zelle gemessen und dann den Mittelwert davon errechnet. Dann bestimmt das Programm wie viele Zellen angeschlossen sind (4, 5 oder 6), indem er die 4. misst, wenn etwas vorhanden ist misst er die 5. Wenn dann dort keine Spannung angeschlossen ist, handelt es sich um ein 4 Zellen Akku. Danach misst er die einzelnen Zellen ob sie über dem Mindestwert sind und in welchem Fünftel (bis zur maximalen Zellenspannung) sich befinden. Entspricht die Anzahl Zellen die in Ordnung sind, der Anzahl Zellen die angeschlossen sind, wird das tiefste Fünftel auf dem LED-Balken angezeigt und an den Controller weitergeschickt.

Alle Uart Kanäle werden über Interrupts und Buffer Funktionen parallel eingelesen und bearbeitet. Dazu gibt es die Lora_BUFFER und Lora_HANDLER sowie die CH_BUFFER und CH_HANDLER Funktion. Mit der Buffer Funktion werden die einzelnen Zeichen abgespeichert bis zu einem gewissen Kontrollzeichen (\n bei Lora und ; bei den anderen Uarts). Danach wird der String im Handler abgespeichert. In der While-Schlaufe wird der Handler wieder erneut aufgerufen und der String wird durch eine Befehlsliste (viele stringcompares) durchgefragt und die dazugehörige Funktion wird ausgeführt.

Um die Motoren zu steuern werden drei Joystick Achsen von dem Controller auf den Chassis geschickt. Der Wandelt das in Vektoren um und rechnet das mit den entsprechenden Motoren zusammen. Der Motorenstrom ist über fünf Bits steuerbar.

Auf dem Print ist eine steckbare 20A Sicherung eingebaut, falls mal irgendein Kurzschluss geschieht, damit der Akku nicht kaputt geht. Die Motoren können zusammen Maximal 8A ziehen. Auf die Dauer erwärmt sich der DCDC auf etwa 70°C und deswegen ist darüber noch 120mm Lüfter eingebaut.

Abbildung 1:DCDC auf ungefähr 70°C

Auf den Seiten wurden LED Streifen befestigt die als Feedback dienen falls irgendetwas passiert. Vorne und Hinten hat es Schweinwerfer die aus den gleichen LEDs bestehen und alle werden über den Arduino gesteuert.

Aufbau

Chassis

Marco Stauber:

Die Idee war es das Chassis unseres Fahrzeugs möglichst flexibel zu gestalten. Dazu gehörte die Kompatibilität mit verschiedenen Akkus, ein zusätzlicher Motor um irgendetwas zu steuern und ganz viel Platz um die Erweiterungen zu verstauen.
Angefangen hat das Design mit einem etwa 40*30 cm grossen Fahrzeug, einem riesigen 6s 10AH Akku, den ich von einem Mitarbeiter geschenkt bekommen habe und eine Federung für die Räder und Motoren, dass der ganze Aufbau ein wenig geschützt ist gegen Aufschläge. Zusammen mit meinem Vater haben wir mehrere Versionen dieser Federung kreiert und diese immer weiter verbessert. Dazu befinden sich an dem Deckel des Fahrzeugs zwei Lüfter die das ganze System kühlen sollten. Der Deckel könnte man mit dem Stepper Motor Auf und Zu machen. Wir hatten damit ein Problem, dass die benötigten Arme, die die den Deckel aufstossen, zu lang wären. Der Akku würde recht viel von dem Print überdecken, somit war die Positionierung des Akkus auch ein Problem.

Über den Winterferien habe ich dann den Print gelayoutet, welches mir garantiert 12 Stunden Arbeit gekostet hat. Gegen Mitte Januar konnte ich dann auch den Print bestücken und den Aufbau des Prints vervollständigen mit den benötigten Litzen. Glücklicher Weise funktionierte der Print beim ersten Mal schon.
Ausser der Bootloader, der denn ST-Link ersetzt, hat irgendein Treiber Problem mit dem wir uns nicht weiter befassen werden. Aus diesem Grund haben wir auch gleichzeitig die 20-Pin Buchse mit auf dem Print befestigt.

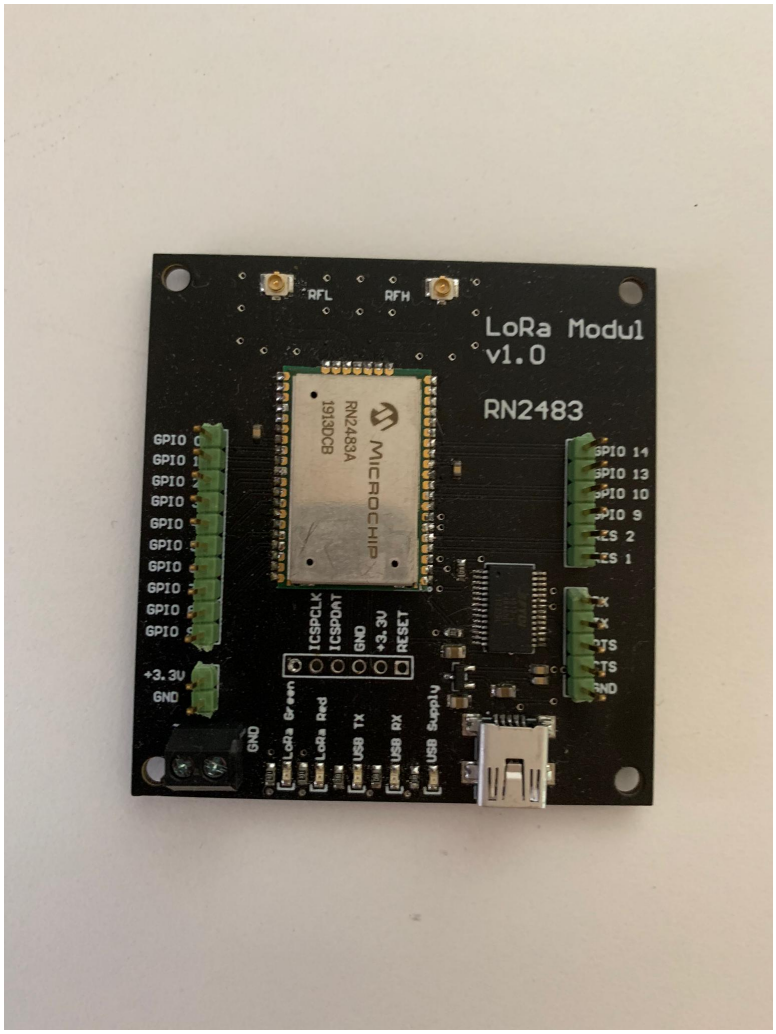
Nach langem überlegen habe ich mit dazu entschieden mit dem Zusammenbau des Fahrzeuges nochmal von vorne zu beginnen.
Neu wird das Fahrzeug etwa 24*20cm.
Als Grundplatte nahm ich eine 4mm dicke Aluminium Platte, die ich in der ETH mit dem Waterjet bearbeitete. Für die Wände nahm ich PMMA welches ich mit dem Lasercutter passend zuschnitt.
Für die Wände gab es ein Halter System aus einem 3D-Druck indem man die PMMA-Platten einfach nur noch hinein schieben kann. Nun wird nur noch ein Lüfter gebraucht, da der Platz nicht mehr vorhanden ist für ein zweiter Lüfter. Das LoRa Modul bekommt ein Halter an der Seite um Platz zu sparen. Hinten und Vorne befinden sich Scheinwerfer, zusätzlich nehmen wir einen kleineren Akku, damit auch der wieder Platz im Fahrzeug hat.
Unter dem Fahrzeug hat es RGB LEDs die den Status des Fahrzeugs wiedergeben als Feedback.
Offensichtlich sind die 4 riesigen DC- Motoren an der Unterseite mit den Mecanuum-Rädern für Bidirektionale Fahrtrichtung.

Im Code wird regelmässig die Akkuzellenspannung gemessen, um sicher zu sein, dass der Akku nicht kaputt gehen wird. Die Stromzufuhr für die Motoren kann man über den Motorentreiber mit 5 Bits steuern um die Geschwindigkeit zu regeln. Die Motoren haben eingebaute Encoder die man mit dem Onboard-Arduino ausmisst und über ein USART Kanal zu dem Hauptmikrokontroller schickt, um die Motoren auszugleichen. Zusätzlich verfügt der Mikrokontroller über drei weitere USART Kanäle: Kommunikation mit einem Computer für direkte Befehle, für das LoRa Modul, und einem Reserveanschluss für weitere Module.

Funk

Wir haben uns für LoRa entschieden, weil es eine riesige Reichweite hat auf einer tiefen Frequenz. Auch bei einer niedrigen Datenrate sollte dies möglich sein.
Microchip bietet ein LoRa Modul (RN2804) mit UART Interface und zwei verschieden Frequenzen an. Wir haben dazu ein Print erstellt um dieses Modul über den Computer oder dem USART Interface eines Mikrokontrollers zu steuern.
Wir hatten zunächst ein wenig Mühe um erfolgreich eine Datenübertragung zu führen. Nach vielen ausprobieren, haben wir verstanden welche Befehle man zusätzlich man eingeben muss.

Die Library die wir dazu haben mit einem eigenen Protokoll für Punkt zu Punkt Verbindungen, sollte den geregelten Ablauf zwischen dem Chassis und dem Controller regeln. Mit ständigem abwechselnden empfange und senden von Daten wir das Fahrzeug gesteuert.



Controller

Lars Peter:

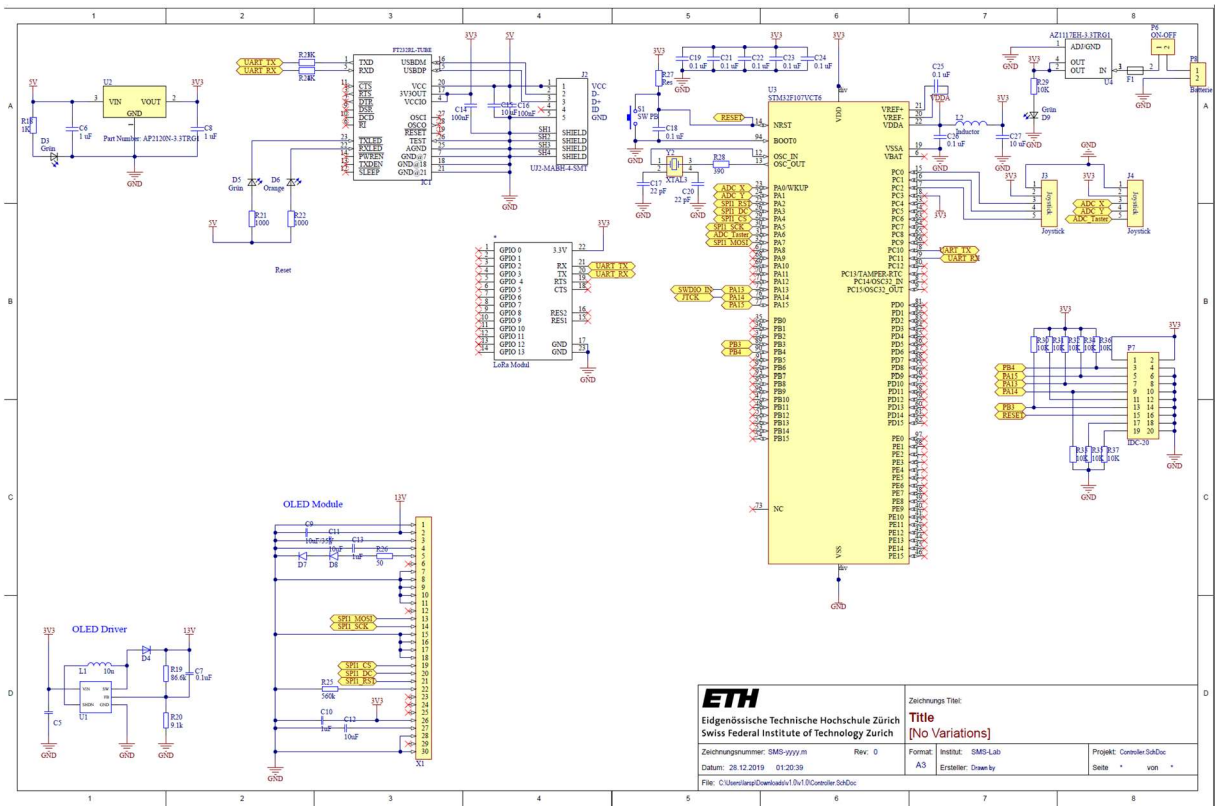
Der Controller sollte die meisten der Funktionen, welche Marco auf dem Chassis einbaut, steuern. Mit einem dem linken Joystick wird die Richtung bestimmt in welche sich das Fahrzeug bewegt, während mit dem rechten Joystick gedreht oder das Menu kontrolliert wird.

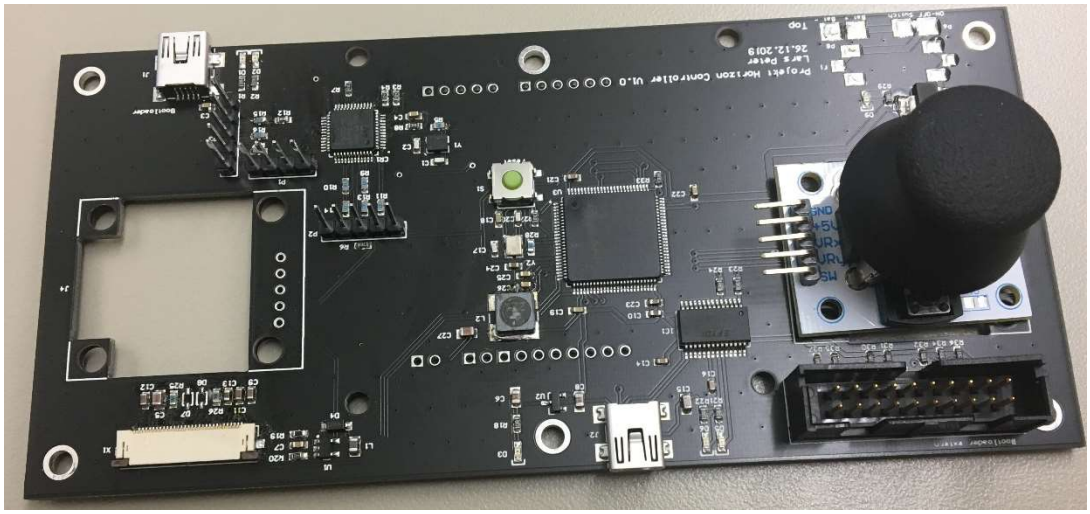
Das OLED Bildschirm zeigt Batteriestatus, Fahrzeugname, Geschwindigkeit und Verbindung an um den Überblick zu behalten während man fährt.

Für den Bildschirm konnte ich ein Teil der Schaltung und Code (GFX Befehle und Charset) von Laurin übernehmen und musste nicht alles neu machen.

Mit dem selbst gemachten LoRa-Modul schicke ich dann die Werte von drei Joystickachsen und eventuelle zusätzliche Funktionen an das Chassis.

Schema:





Schema des Controllers
Der Print läuft mit 3.3 V welche gleich am Anfang aus der 9 V Batterie oder direkt über den USB-Anschluss 5 V gemacht werden. Der OLED-Bildschirm braucht 13 V, diese werden mit einem OLED-Driver (Boost Converter) erreicht.

Mit den ADCs werden die zwei Joysticks eingelesen, welche danach über das LoRa Modul an das Chassis geschickt werden.
Über SPI kommuniziere der Microkontroller mit dem OLED-Bildschirm.
Der selbstgemache Bootloader sollte eigentlich den ST-Linkt ersetzen. Der JTAG Anschluss ist zur Sicherheit für den Fall, dass dieser nicht läuft auch eingebaut.
Über Uart Kommunizieren der uC mit dem LoRa-Modul welches die Daten an das Chassis weiter schickt.
Auch über Uart kann der uC mit einem PC kommunizieren.

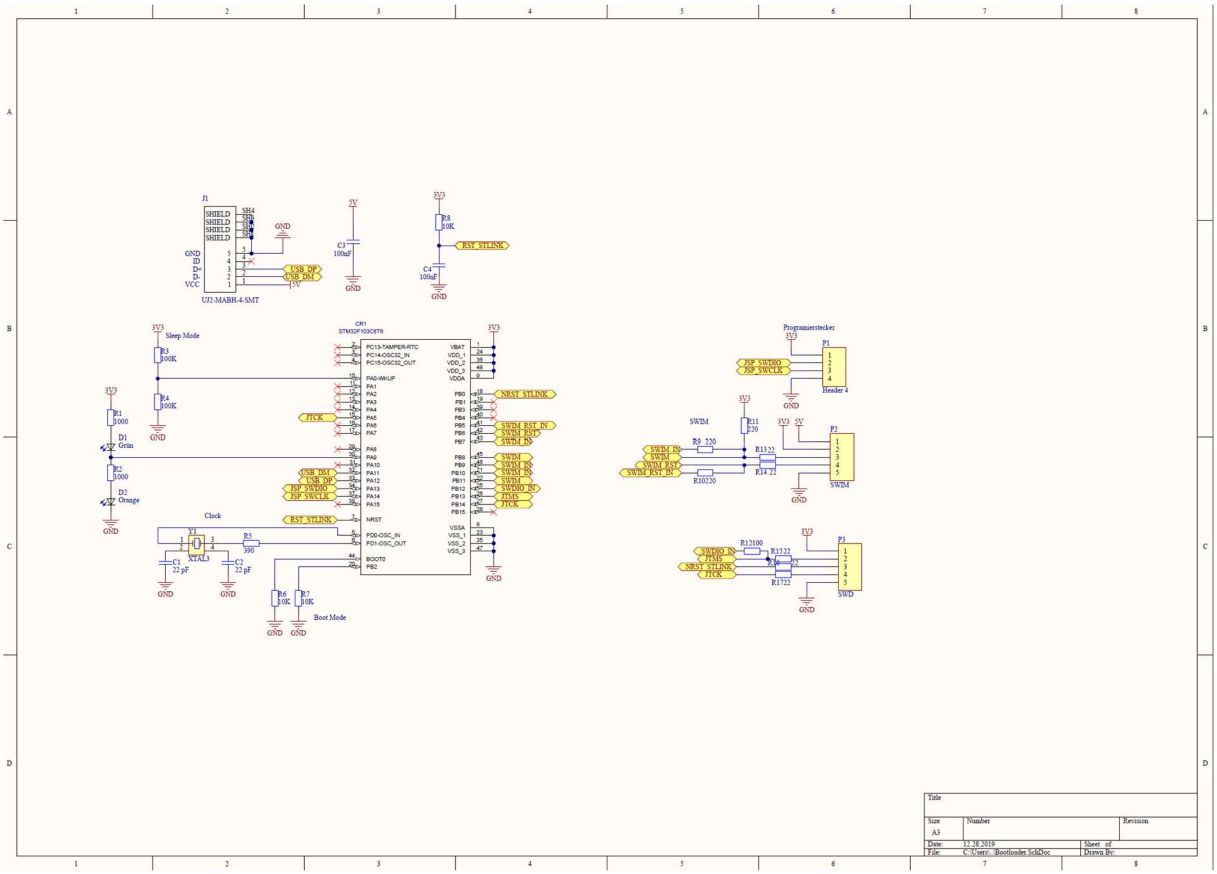
Software:
Den Code zum Einlesen der 2 Joysticks habe ich zuerst auf dem STM32 Board geschrieben um sie zu Testen. Der Plan war dieselbe Software für den eigentlichen Controller später zu gebrauchen. Jedoch kam dann noch der Bildschirm hinzu. Welcher nicht mit der TouchP0P1 programmiert werden kann. Darum brachte mir dieser Code am Schluss leider doch nur sehr wenig und es musste alles nochmals mit der hal Library neu geschrieben werden.

Joystick testen.MOV

Im Controller werden nur die Joysticks eingelesen und der Wert über das LoRa Modul zum Chassis geschickt.

Probleme:
Da nur Ich mein Schema überprüft habe und es nicht noch meinem Lehrmeister gezeigt habe, sind leider einige unnötige Fehler darauf:
Schutzdiode über Spannungsregler fehlt.
Das LoRa Modul und der COM-Port ist auf denselben UART Kanal geschalten.
Der Bootloader konnte hier nicht richtig programmiert werden, darum habe ich ihn in Ruhe gelassen um nicht zu viel Zeit dafür zu verschwenden.
Der Stecker für den Bildschirm befindet sich am falschen Ort und musste die SPI Pins auf die Bootloader Pins führen.
Diese Fehler kann ich aber zum Glück einigermassen gut korrigieren ohne allzu grossem Aufwand, ausser bei den SPI Leitungen, da diese einige Drähte brachten.

Bootloader:



Das Schema des Bootloaders hat Marco im Internet gefunden. Zu dem Schema gibt es einen ganzen Guide wie es gemacht werden sollte. Leider hatten wir trotzdem Probleme. Und wir konnten nicht mit ihm kommunizieren.
Zum Glück haben wir auch den JTAG Anschluss für genau diesen Fall mit eingebaut. Um einfach über den fertigen ST-Link vom MCB32 Board weiter zumachen ohne viel Zeit zu verschwenden.

Tagebuch

Datum	Tätigkeit
30.08.19	Flyer gestaltet und Terminplan erstellt
06.09.19	LoRa Modul Organisieren, Layout aus Datenblatt nehmen und Print selbst machen. Joystick und Schiebepotentiometer reserviert. Blackbox für Chassis erstellt. Bauteile zusammengesucht.
13.09.19	LoRa Modul austesten. Schema und Layout für eigenes LoRa Modul fertig und Malacarne gezeigt, kleine Änderung. Jetzt parat zum Bestellen.
20.09.19	Prints bestellt Problem mit Bezahlung gelöst. Bauteile bestellt
26.09.19	Prints bestückt und geprüft
27.09.19	Lora getestet -> funktioniert bis auf ein falschen vor widerstand einer Led.
04.10.19	Kontroller Planung. Visualisierung, Flussdiagramm Bauteile für Chassis und Kontroller gesucht
25.10.19	Joystick Ansteuerung getestet (Programmierung) Start Schema von Chassis, Lib zusammenstellen
1.11.19	Joystickprogrammierung mit TouchP0P1 Library fertig programmiert noch nicht getestet. Änderung von Powersupply im Chassis verursacht durch anderen Akku.
2.11.19	Joystickprogramm getestet und überarbeitete. Funktioniert.
8.11.19	Da wir das Display von Laurin benutzen und er mit der Hal Library arbeitet muss müssen die Joysticks anscheinen auch mit der Hal Library programmiert werden, weil die TouchP0P1 und die Hal sich nicht verstehen. An Chassis Schema weiter gearbeitet
15.11.19	Schema für Controller begonnen Arduino und Lüfter für Chassis hinzugefügt. Motoren, + deren Kühler und Räder herausgesucht.
22.11.19	Schema für Controller beinahe fertig. Es fehlt noch Layout und Stückliste. Servo für Chassisgehäuse hinzugefügt, Schema erweitert. Stromquelle für Controller ausgesucht. Planung angefangen für Chassis Aufbau. Allgemein: Bootloader hinzugefügt als Ersatz für St/linkv2.
29.11.19	Layout begonnen und Bauteile gezeichnet. Motor gesucht und Stückliste weitergeführt.
6.12.19	Layout Controller und Chassis
13.12.19	Layout Controller und Chassis
20.12.19	Layout Controller und Chassis
29.12.19	Bestellung von Bauteilen und PCBs
10.1.20	Bestücken von Controller und Chassis
17.1.20	Bootloader Setup fehlgeschlagen. Fehler davon suchen
24.1.20	Bootloader kann immer noch nicht programmiert werden.
31.1.20	Bootloader wird ignoriert, nur noch mit Stlink programmieren. Chassis LED programmieren.
21.2.20	Redesign von Chassis: Es wird kleiner. Entwicklungen von 3D Modell von Chassis: Aluplatte
28.2.2020	Entwicklungen von 3D Modell von Chassis: Wände

6.3.2020	Dokumentation von Schema und Layout
13.3.2020	Entwicklungen von 3D Modell von Chassis, Lora Halter
27.3.2020	Entwicklungen von 3D Modell von Chassis, Scheinwerfer
3.4.2020	Test Programm auf Windows für LoRa Modul: Reichweite und Datendurchsatz. Test fehlgeschlagen: Nicht jeder PC unterstützte das Programm. Programmierung von Funktion für Messung von Batterie Spannungswerte.
8.5.2020	Entwicklungen von 3D Modell von Chassis, Seilwinde. Ansteuerung von Arduino und Motor programmiert
15.5.2020	Programmierung von LoRa Initialisierung und paralleler UART Kommunikations Handler.
22.5.2020	Zuschneiden von Aluplatten für Chassis mit Waterjet.
29.5.2020	Entwicklungen von 3D Modell von Chassis, Deckel
5.6.2020	Beginn Zusammenbau von Chassis
12.6.2020	Acrylplatten zuschneiden für Wände von Chassis. 3D Druck von Wandhalter für Chassis.
19.6.2020	Weitere Dokumentation von Chassis und Controller
22.6.2020	Controller Print überarbeiten. Fehler beheben
26.6.2020	Einbau von OLED Bildschirm bei Controller und Ansteuerung.
3.7.2020	3D Druck von Gehäuse für Controller Feinschliff an Doku und Programmen

Anhang

Alle Dateien sind unter folgendem GITHUB verfügbar:

https://github.com/Monerfone/MCB-Mobil

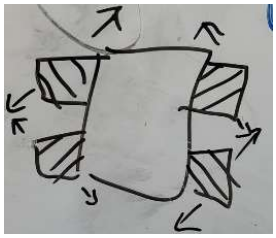


Abbildung 2: Prinzip der Mecanuum-Räder

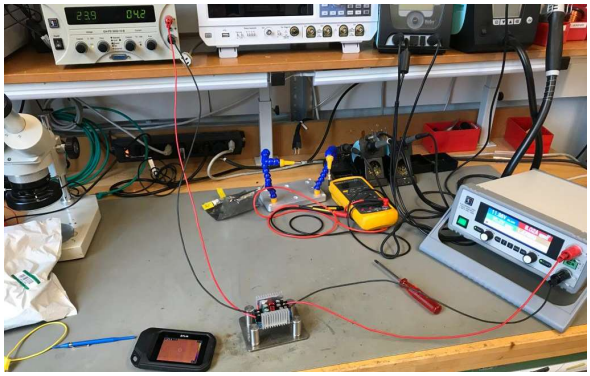


Abbildung 5: DCDC für das Chassis Messung

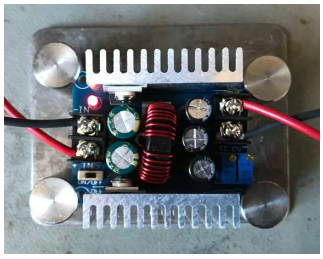


Abbildung 4:DCDC für das Chassis

Abbildung 3: DCDC auf ungefähr 80°C

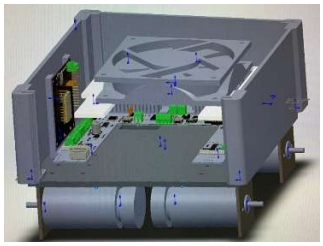


Abbildung 7: 3D Model vom Chassis

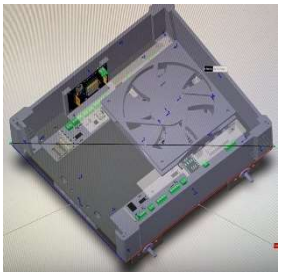


Abbildung 6: 3D Model vom Chassis

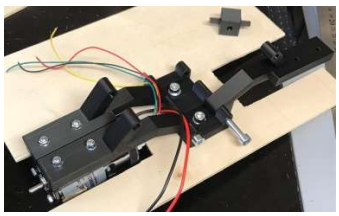


Abbildung 8: Federung des Fahrzeuges der ersten Idee

Endergebnis

Mit vielen Nervenzusammenbrüche auf Marcos Seite sind wir doch zu einem Ergebnis gekommen. Es hat zwar zeitlich nicht mehr für die Seilwinde gepasst trotzdem funktioniert das Fahrzeug und der Controller und das Fahrzeug sehen so aus wie erwartet.

Lars hat gegen den Schluss ein wenig nachgelassen, weil wir kein richtiges Zeitlimit hatten und die ganze Arbeit nach hinten geschoben hat. Marco hat als er so im Schuss war gleich noch einige von Lars Aufgaben übernommen.