Mechatronik Trinational
Mécatronique Trinationale

Fachhochschule Nordwestschweiz

DHBW Duale Hochschule Baden-Württemberg

IUT Mulhouse Institut Universitaire de Technologie UNIVERSITÉ HAUTE-ALSACE

Autonomous Beer-Cooler

Semesterarbeit

Muttenz, Januar 2023

Studenten	Max Knauber Matthias Gass Fabian Schenker
Fachbetreuer	Silvan Wirth

Fachhochschule Nordwestschweiz, Hochschule für Technik

Zusammenfassung

Deutsch

Für das mechatronische Projekt im 5. Semester haben wir uns einen Beercooler mit Follow-Funktion vorgenommen. Dieser soll sich per Bluetooth mit dem Smartphone verbinden und dem User hinterherfahren, bis er die Follow-Funktion ausschaltet. Der Beercooler soll eine Kühlbox tragen, die Platz für mindestens zwölf 0.5L Dosen hat.

Diesee Dokumentation beschreibt den Prozess vom Erstellen des Konzepts über die Konstruktion und Programmierung bis hin zum Erstellen eines funktionsfähigen Prototyps.

Français

Pour le projet mécatronique du 5ème semestre, nous avons décidé de créer un Beercooler avec fonction Follow. Celle-ci doit se connecter au smartphone via Bluetooth et suivre l'utilisateur jusqu'à ce qu'il éteigne la fonction Follow. Le Beercooler doit porter une glacière pouvant contenir au moins douze canettes de 0,5 litre.

Cette documentation décrit le processus de création du concept, de la construction et de la programmation jusqu'à la création d'un prototype fonctionnel.

English

For the mechatronic project in the 5th semester, we have planned a beercooler with a follow function. It should connect to the smartphone via Bluetooth and follow the user until he switches off the follow function. The beercooler should carry a cooler that has space for at least twelve 0.5L cans.

This documentation describes the process from creating the concept, through design and programming, to creating a working prototype.

Vorwort / Dank

Dieses Projekt hat gezeigt, wie wichtig eine gute Zusammenarbeit und Einsatz der fachlichen und sozialen Kompetenzen von jedem ist, die zum Erfolg von diesem Projekt beigetragen haban.

Der FHNW danken wir für die Arbeitsmöglichkeiten im Labor, der Werkstatt und den Gruppenräumen.

Herrn Silvan Wirth danken wir für die Begleitung während des Projekts und für die Beantwortung bei allfälligen Fragen.

Wir danken uns gegenseitig für die gute Zusammenarbeit.

Weiter geht unser Dank an alle weiteren Personen, welche hier nicht namentlich aufgelistet sind und uns bei dieser Projektarbeit ebenfalls unterstützt haben.

Rahmenbedingungen

Übersicht

- Zu realisierende Projekte können vorgeschlagen werden (Berücksichtigung gewisser Rahmenbedingungen)
- Bearbeitung in Zweier- oder Dreiergruppen
- Vorgabe SGL: lauffähige Produkte mit Aussenwirkung für Marketing
- Zuweisung von max. zwei Gruppen auf ein Projekt
- Austausch zwischen Gruppen möglich (Lösungsvarianz höher, Chance, dass ein lauffähiges Produkt entsteht, ist höher)

Rahmenbedingungen Projekte

- Enthält Merkmale eines mechatronischen Systems (Sensorik, Aktorik, Informationsverarbeitung (Rechner / Steuerung))
- Aktorik: physische Bewegung muss vorhanden sein
- User-Interface: optional, nicht zwingend
- Energieversorgung sinnvoll gelöst

Qualifikationsziele und Kompetenzen

- Sachkompetenz:
  - Verstehen und praktisches Anwenden der mechatronischen Modellbildung
- Selbstkompetenz:
- – Eigenständige Auswahl und Einsatz von Aktoren, Sensoren, Mikrorechner
- Sozial-ethische Kompetenz:
  - ein System vom Konzept bis zum funktionierenden Produkt entwickeln.
  - mechatronisches Projekt im Team erfolgreich planen und durchführen.
  - gruppendynamische Prozesse bei der Bearbeitung größerer Aufgaben innerhalb von Projektgruppen erfahren

Unterstützung des Lernprozesses

- Aufwand gemäss Modulhandbuch: 15 h Präsenz, 45 h Selbststudium
- Anwenden des theoretischen Wissens aus der Vorlesung «Mechatronische Systeme» (und aller anderer vorgängiger Vorlesungen)
- Praktische Realisierung wird begleitet durch Dozenten.

Verknüpfung in Modulgruppe Mechatronik III

- Fach «Mechatronische Systeme» wirkt auf Fach «Mechatronisches Labor»
- Inhalte und Methoden aus Vorlesung sollen angewendet werden
- Unterstützung bei Realisierung eines konkreten Projektes

Prüfungsbedingungen

- Leistungserfassung durch Präsentation und Bewertung des Projektes
- Termin: 10.01.2023 08:30 – 12:15
- Gewichtung: Schlussnote gemäss Bewertungsraster

Youtube-Videos / Mechatronik-Trinational Channel

Zu den Projekten werden Videos gestaltet. Diese fliessen in die Bewertung der Projekte mit ein (siehe Bewertungsraster).

Das Video Ihres Projektes muss «Youtube-konform» sein, z.B. sind die Musik-Copyrights zu beachten und wenn Sie (aus welchen Gründen auch immer) nicht im Video zu sehen sein wollen, dann sollten Sie sich auch nicht selbst aufnehmen.

Projektrahmen

- Kostendeckel CHF 200.-pro Projekt
- Kann ggf. bei marketingtechnischem Nutzen erhöht werden (nach vorheriger Absprache)
- Materialbeschaffung selbst zu organisieren
- Abrechnung über Sekretariat/Studiengangsleitung zum Ende des Semesters
- Aufstellung mit Belegen und Kontoverbindung (IBAN) gemäss Formular
- Projekt muss vorgängig durch Dozenten genehmigt werden
- Arduino und/oder RaspberryPi wird abgegeben durch Labor (muss nicht in den Projektkosten berücksichtigt werden)

Laborzugang und Werkstattbenutzung

- Die Studierenden dürfen aus Sicherheits-und Versicherungsgründen das Labor Mechatronik Trinational / Campus Muttenz nicht unbegleitet (ohne Dozierenden) nutzen
- Zugang Labor Mechatronik Trinational gem. Unterrichtszeiten
- Werkstattaufträge (extern oder Hochschulen) sind vorgängig mit dem Dozierenden abzusprechen
- Kosten sind vorgängig zu klären und fliessen ins Projektbudget ein

Inhaltsverzeichnis

1 Einleitung

1.1 Sinn und Zweck der Dokumentation

Diese Dokumentation des Projekts hält die einzelnen Phasen und Arbbeitsschritte fest. Es soll unsere Gedankengänge und Entscheidungen nachvollziehbar aufzeigen und begründen. Es beinhaltet die Anforderungen, welche an die Projektarbeit des 5. Semesters gestellt werden.

1.2 Vision (Inhalt und Ziele)

Der Following Beercooler soll den Transport von Bier zu einem Tag am Strand oder auf der Wiese etwas angenehmer gestalten, indem er das Schleppen der Getränke selbst übernimmt. So besteht die Möglichkeit alle restlichen Sachen einfacher tragen zu können und die schweren Bierdosen fahren mit den Beercooler ohne weiteren Aufwand hinter einem her. Am Zielort angekommen sorgt der Beercooler dafür, dass die Getränke noch deutlich länger als in einer normalen Kühlbox kalt bleiben.

Die Basis kann indes auch für den Transport anderer Dinge benutzt werden, da die Kühlbox im finalen Design des Systems demontierbar ist.

1.3 Definitionen und Abkürzungen

Bzw.	Beziehungsweise
Resp.	Respektive
MDF	Mitteldichte Faserplatte
GPS	Global Position System
SDA	System Data
SCL	System Clock
PWM	Pulsweitenmodulation
UART	Universal Asynchronous Receiver Transmitter
I2C	Interner Integrated Circuit Bus
Akku	Akkumulator
DC	Gleichstrom
LiPo	Lithium Polymer
MISO	Master in Slave out
MOSI	Master out Slave in
IDE	Integrated development environment

1.4 Ablage, Gültigkeit und Bezüge zu anderen Dokumenten

Das Dokument bezieht sich auf die Vorlesung «Mechatronisches Labor» und deren Bewertungskriterien, welche für das Semesterprojekt der Mechatronik Trinational gelten. Alle verwendeten Quellen sind im Quellenverzeichnis/Literaturverzeichnis angegeben.

1.5 Verteiler und Freigaben

Tabelle 1.1: Verteiler und Freigaben

Rolle / Rollen	Name	E-Mail
Projektleiter Hardware	Gass Matthias	matthias.gass@students.fhnw.ch
Projektleiter Software	Knauber Max	max.knauber@students.fhnw.ch
Projektleiter Konstruktion	Schenker Fabian	fabian.schenker@students.fhnw.ch
Kunde	Vertreten durch Silvan Wirth	silvan.wirth@fhnw.ch
Anwender	Robert Alard	robert.alard@fhnw.ch

2 Aufgabenanalyse

2.1 Systemvoraussetzungen

Unser System ist in sich geschlossen. Einzig für die Kommunikation mit dem Arduino werden wir uns einer App bedienen, die als User-Interface dienen soll. Auch soll das System Akkubetrieben sein, was einen vollen Akku und dementsprechend ein passendes Ladegerät voraussetzen.

2.2 Problemdefinition

Das schwere Schleppen von einer Kühlbox, soll mithilfe eines mechatronischen Systems erleichtert werden. Die Idee war, die Schlepparbeit vom Auto zu einem Ort, an dem man sich mit der Familie oder mit Freunden niederlässt, etwas leichter zu gestalten. Dies soll auch als Problemlösung für Personen dienen, die nicht mehr schwere Dinge tragen dürfen oder können. Dabei soll auf etwas Lustiges und Nützliches zurückgegriffen werden können.

2.3 Systemabgernzung

Um die Aufgaben für das Projekt einzugrenzen und eine Übersicht über die zu erreichenden Punkte zu erhalten, wurde eine Systemabgrenzung, zur von uns selbst gestellten Aufgabenstellung, erstellt. Diese Systemabgrenzung, angefangen mit dem Umsystem nach den PESTEL-Guidelines zeigt an sich weder Details noch Unerwartetes auf:

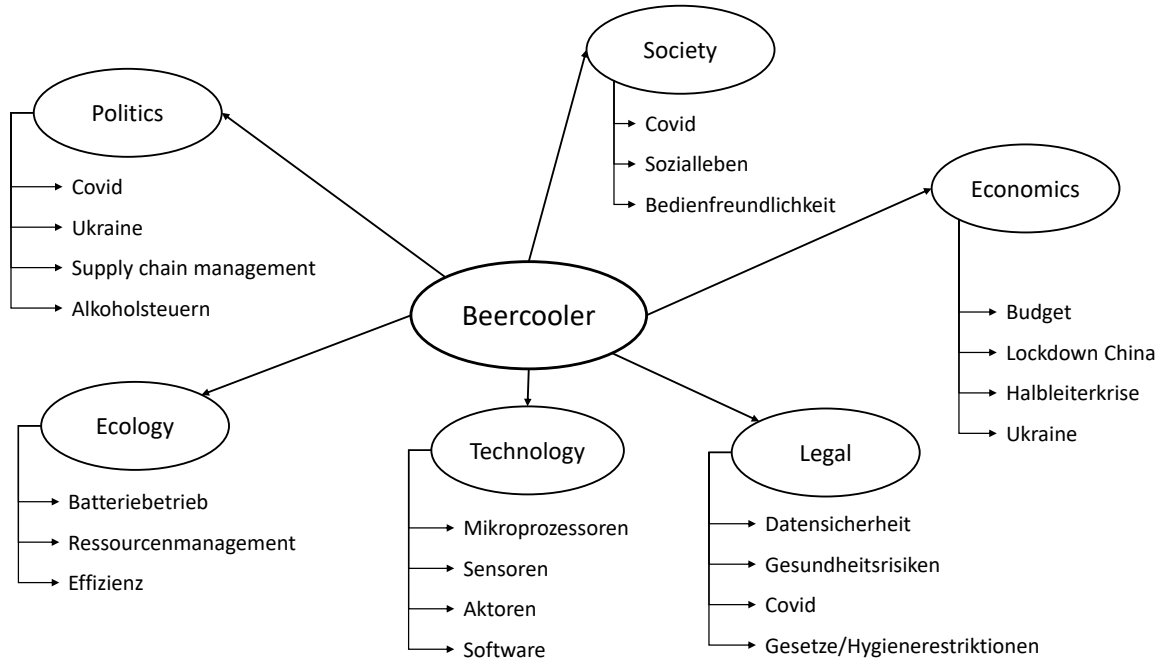

Abbildung 2.1: Umfeld der Systemabgrenzung nach PESTEL

Bei genauerer Betrachtung des Umsystem kommt bereits etwas mehr zum Vorschein:

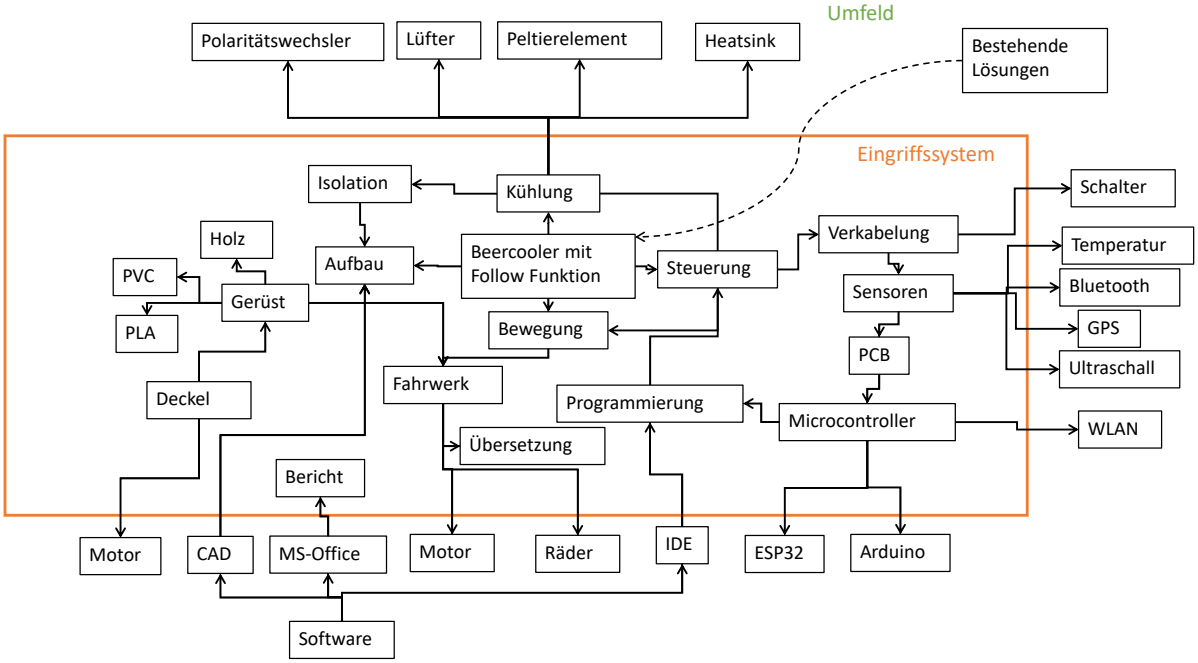

Abbildung 2.2: Systemabgrenzung

Im Eingriffssystem sind alle Punkte enthalten für welche Eingriffe, Veränderungen und neue Konzepte im Rahmen der Aufgabenstellung zu erledigen sind. Das Umfeld beinhaltet die relevanten Teile für die Erarbeitung der Punkte im Eingriffssystem.

2.4 Stärken- und Schwächenanalyse

Tabelle 2.1: Stärken- und Schwächenanalyse

Erfolgsfaktor	**--**	**-**	**0**	+	++	**Begründung**
Arbeitszeiten		X				Zugang zum Labor ist eingeschränkt
Betreuung			X			Nur während Laborzeiten
Abgabetermin				X		Im Januar nach Ferien
Innovation			X			Viele ähnliche Projekte im Internet
Infrastruktur				X		Mechatronik Labor und FHNW Werkstatt
Informatik Know-How		X				Kein Teammitglied aus diesem Fachgebiet
Kommunikation				X		Ohne Probleme
Kosten			X			Eher knapp
Materialbeschaffung				X		Selbstversorgung, Abrechnung am Ende des Projekts
Motivation					X	Sehr hoch, da eigen ausgewähltes Projekt
Projekterfahrung				X		Nach Semester 4 und Stage 2 vorhanden
Sicherheit			X			LiPo Akku
Technisches Know-How				X		Unterschiedliche Stärken der Teammitglieder
Zusammenarbeit					X	Sehr gut privat befreundet

3 Zielformulierung

3.1 Ziele und Nutzen des Auftraggebers

Der Auftraggeber wünscht sich ein lauffähiges Produkt, welches alle typischen Eigenschaften eines mechatronischen Systems (Sensorik, Aktorik, Informationsverarbeitung (Rechner / Steuerung)) enthält. Dabei sollen die Kenntnisse aus den früheren Semestern des Studiums angewendet werden.

3.2 Ziele und Nutzen des Anwenders

Der Anwender erwartet vom System eine simple Bedienung und eine unkomplizierte Benutzererfahrung. Er möchte möglichst wenig Schritte ausführen, um den Roboter zum Folgen zu bewegen. Wo immer ein Mensch mit Flip-Folps geht, ausser ins Wasser natürlich, sollte der Roboter folgen können.

3.3 Anforderungen

Anforderungen an das System sind, dass sich eine Kühlbox auf Rädern selbst fortbewegen kann, wobei es leichtes Gelände bewältigen können muss, wie zum Beispiel eine Wiese. Des Weiteren sollen die darin aufbewahrten Getränke deutlich unter Lufttemperatur temperiert werden.

3.4 Zielkatalog

Tabelle 3.1: Zielkatalog

Objekt	Eigenschaft	Ausmass	Zeitpunkt	Zielart	Priorität
Level 1					
Gerüst	Aufbau des Roboters mit Fahrwerk, Halterung für Kühlbox und Elektronik	zusammengebaut	KW 46	M	-
Follow-Funktion	Roboter kann autonom jemandem nachfahren	erreicht	KW 50	M	-
isolierte Box	Eine Kiste, welche die Innentemperatur von der Aussentemperatur isoliert.		KW 44	M	-
Kühlen	Die isolierte Box soll gekühlt werden	< 5 °C	KW 46	R	100
Akku	Der Akku soll für 3h aktive Kühlung und 1h Fahren ausreichen	erreicht	KW 48	M	-
Kapazität	Es soll mindestens Platz für 12 0.5 L Dosen haben	erreicht	KW 44	M	-
Level 2					
Wechsel Akku	Akku soll auswechselbar sein	erreicht	KW 48	W	40
Deckel	Der Deckel soll sich automatisch öffnen können	erreicht	KW 49	W	30
All-Terrain	Der Roboter soll auch über kleinere Hindernisse fahren können	5cm Schwelle	KW 49	O	30
Federung	Einbau einer Federung/Dämpfung	erreicht		W	20
Level 3					
Leine	Leine, an der der Roboter gezogen werden kann, falls der Akku leer ist.	erreicht		W	30
Variable Temp.	Die Temperatur in der Kühlbox soll variabel geregelt werden	erreicht		W	10
Wärmen	Die Isolierte Box soll heizbar sein	Warm 50° C		W	40

3.5 Benutzer / Zielgruppe

Tabelle 3.2: Benutzer / Zielgruppe

Zielgruppe	Name	Beschreibung
Stakeholder	FHNW, Silvan Wirth	Ansprechperson / Auftraggeber
Anwender	Prof. Dr. Robert Alard	Kunde
Zielgruppe	Studenten/Familien/Marketing	Potentielle Studieninteressierte

4 Konzepterarbeitung

4.1 Morphologischer Kasten

Teilfunktion \ Lösungsvarianten	1	2	3	4	5	6	Bemerkungen
Folgemechanissmus	Blynk (App und GPS/Kompass)	GPS Modul & Kompass	Bluetooth Tag (ESP32)	Infrarotsensor (Cam + Reflector	Objekterkennung		
Antrieb	2 DC Elektromotoren	Zentraler Antrieb und Getriebe	4 DC Elektromotoren				Freilaufdiode
Kühlmechanissmus	Peltier Element	Kompressor	gekaufte Kühlbox				
Mikrocontroller	Arduino	Raspberry Pi	ESP32				
Rahmen	Kuststoff	Alu	Holz				
Fahrwerk	2 Räder Angetrieben 1 lose	2 Räder Angetrieben, 2 lose	4 Räder, 1 Achse angetrieben, 2 Lenkung	4 Räder angetrieben, Steuerung über Ansteuerung			
Federung	keine	Blattfeder	aktiv	Gummidämpfung	Hydraulikelement		
Verkleidung	Kunststoff	Alu	Holz				
Energiesource	Akku	Brennstoffzelle					Energierechnung
Transport	Tragegriff	Box herausnehmbar					
Sekundärantrieb	Leine	Führungsstock					

Abbildung 4.1: Morphologischer Kasten

Im morphologischen Kasten wurden drei verschiedene Lösungsvarianten für das Projekt eingetragen. Nach Abwägung der Schwierigkeit in der Herstellung und der Zuverlässigkeit in der Bedienung mithilfe des Zielkatalogs, wurde die blaue Lösung weiterverfolgt.

4.2 Ressourcen Personell und Materiell

Als intellektuelle Ressourcen gelten die Vorkenntnisse von Matthias Gass als Automatiker, Max Knauber als Motorradmechaniker und Fabian Schenker als Polymechaniker, wodurch die mechanische Seite des Projekts gut abgedeckt ist. Die Seite der Informatik bedarf etwas Einarbeitung, aber dank des bereits absolvierten Teils des Studiums der Mechatronik trinational gibts es auch dort einige Vorkenntnisse.

Für technische Ressourcen und Fertigung von Kleinteilen seien folgende Werkstätten erwähnt:

- 3D-Drucker
- Trinat-Labor
- Mechanische Werkstatt der FHNW-Muttenz

4.3 Projektablauf

In der folgenden Tabelle wird der Ablauf des Projekts inklusive Terminierung der Arbeitsschritte beschrieben.

Tabelle 4.1: Projektablauf

Vorbereitungsphase	
Grobkonzept erarbeiten	bis 21.10.2022
Freigabe Pflichtenheft	**21.10.2022**
Erster Prototyp	
Design erstellen	bis 18.11.2022
Nötiges Material bestellen/erstellen	bis 18.11.2022
Erste mechanische Tests	bis 18.11.2022
Erste Tests mit Elektronikkomponenten	bis 18.11.2022
Meeting mit Stakeholder	**18.11.2022**
Zweiter Prototyp	
Design verbessern	bis 16.12.2022
Weitere mechanische Tests	bis 16.12.2022
Kühlbox anpassen	bis 16.12.2022
Meeting mit Stakeholder	**16.12.2022**
Finales Design	
Finales Design	30.12.2022
Elektrik verlegt und angeschlossen	30.12.2022
Software einsatzbereit	30.12.2022
Voraussichtliche Beendigung	08.01.2023
Präsentation und Bewertung	**10.01.2023**

4.4 Grobschätzung des Aufwands

Der Aufwand, in Stunden, wurde in der untenstehenden Tabelle grob geschätzt und pro Person aufgeteilt.

Tabelle 4.2: Grobschätzung des Aufwands

Thematik	**Matthias**	**Max**	**Fabian**
Grundkonzept erarbeiten	4	4	4
Pflichtenheft	4	8	4
Materialbeschaffung	4	4	3
Erster Prototyp	5	2	26
Zweiter Prototyp	4	4	22
Finales Design	16	16	8
Kühlbox-Umbau	12	-	-
Elektronik verkabeln	8	6	-
Software	-	60	15
Dokumentation	4	6	12
Präsentation	6	2	2
Testen	2	20	10
Video	36	1	1
Total	**105**	**133**	**107**

4.5 Budget

Tabelle 4.3: Budget (maximal 200.00 CHF)

Gegenstand	Rechnung (in CHF)	Datum Rechnung	Lieferant	Rechnungsnummer
Bollerwagen - Räder	20.00	10.10.2022	Tutti	1
Kühlbox elektrisch	20.00	20.10.2022	Tutti	2
GPS Modul: Zhiting Neo 7m	11.73	24.10.2022	Amazon	3
Bluethooth Modul: HC-06	8.78	24.10.2022	Amazon	3
Kompass Modul: AZDelivery GY-271	6.21	24.10.2022	Amazon	3
2Pcs Schrittmotortreiber BTS7960	15.77	24.10.2022	Amazon	3
Spannungsregler: LM2596HV	6.90	24.10.2022	Amazon	3
Zylinderschrauben lang M6x60	11.74	15.11.2022	Hornbach	4
Sechskantschraube M8x140	3.90	15.11.2022	Obi	5
Step-Up 12-35V 150W	8.86	18.11.2022	Amazon	6
Deans T-Plug Steckverbindungen	7.55	18.11.2022	Amazon	6
Transport Lenkrolle 125mm 100kg	10.60	03.12.2022	Galaxus	7
Rundrohr Alu	6.10	31.12.2022	Jumbo	8
Holzzuschnitt	9.05	20.12.2022	Jumbo	9
PLA/PET Teile gedruckt	52.80		Fabian/Max/Matthias	99
Batterie:		Gestellt von der FHNW	FHNW	0
Arduino:		Gestellt von der FHNW	FHNW	0
			Fabian	
Motoren:		Gestellt von Fabian		0
Holzreste:		Gestellt von Matthias	Matthias	0
Gesamte Rückzahlung	**199.99**			

5 Konzeptbeschreibung

5.1 Systembeschreibung

Das System besteht zentral aus einem Mikrokontroller, welcher sich über Bluetooth mit einer App auf dem Smartphone verbinden lässt. Von dieser App erhält der Mikrokontroller GPS-Koordinaten, welche er dank eines eigenen GPS-Sensors vergleichen kann. Über einen Kompass und diese beiden Koordinatensätze ist das System in der Lage die verbauten Motoren so anzusteuern, dass die Distanz zwischen den beiden Koordinatensätze sich verringert.

Des Weiteren verfügt das System über eine Box, welche für mindestens 12 Bierdosen à 0.5L Platz hat und diese aktiv kühlt.

Die Energieversorgung wird über einen verbauten Akku und eine Custom-Spannungsversorgung gewährleistet. Es handelt sich um einen LiPo Akku mit einer Kapazität von 16000 mAh.

Über eine Bluetooth Verbindung soll der Roboter GPS Koordinaten vom Smartphone erhalten. Er soll eine Distanz und einen Winkel zwischen diesen GPS-Koordinaten und denen des sich an Bord befindenden Sensors errechnen. Über den Kompass soll er stets die eigene Ausrichtung kennen und den errechneten Winkel in Relation setzen. Durch kontinuierliche Wiederholung dieses Prozesses soll der Roboter in der Lage sein, sich dem Smartphone anzunähern.

Davon unabhängig soll sich die Kühlbox von der Batterie aus mit Strom versorgen und die ausreichende Kühlung der Getränke sicherstellen. Dieser Prozess findet ohne Überwachung statt.

5.2 Grundsätzlicher Aufbau (Blockschaltbild)

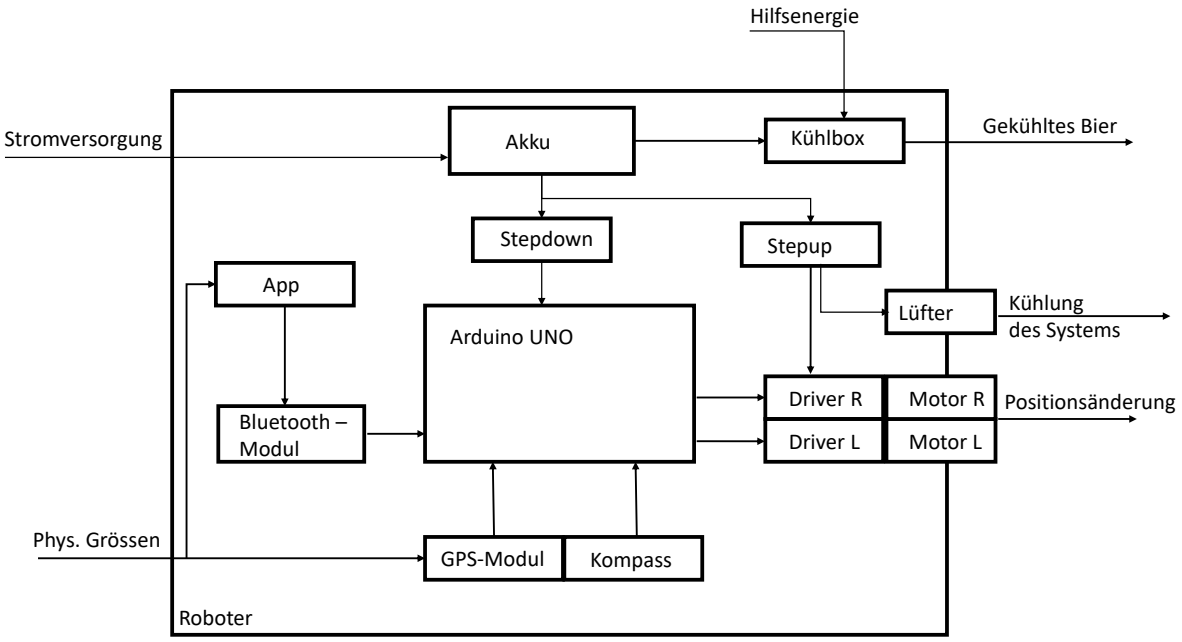

Abbildung 5.1: Blockschaltbild

5.3 Use-Case Übersicht

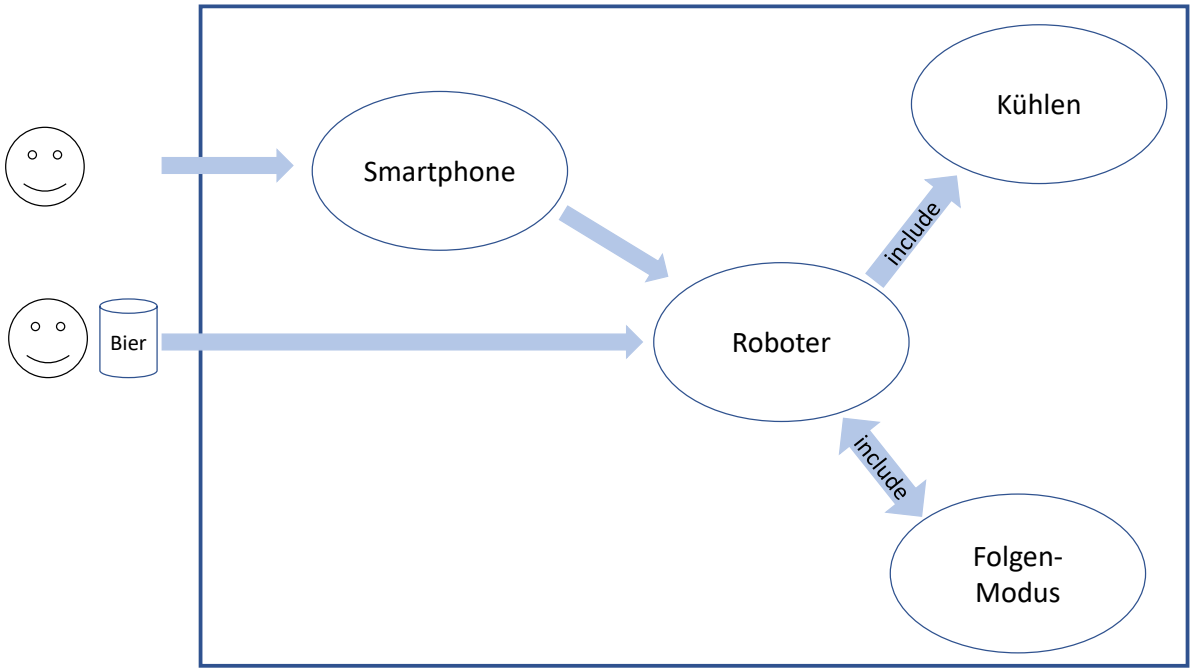

Abbildung 5.2: Use-Case Diagramm

Tabelle 5.1: Use-Case Übersicht

Nr.	Titel	Beschreibung
1	Folgen-Modus	Der Benutzer kann über eine Applikation auf seinem Smartphone den Folgen-Modus aktivieren und deaktivieren, wodurch der Roboter dem Smartphone des Benutzers folgt.
2	Kühlen	Der User kann mindestens 12 Bierdosen in der Kühlbox platzieren und diese mit deutlich unter Lufttemperatur wieder aus der Kühlbox entnehmen.

5.4 Vergleich mit bestehenden Lösungen

Es existiert bereits mindestens ein solches, sehr gut dokumentiertes Projekt, an welchem wir uns orientiert haben. Es soll aber kein kompletter Nachbau werden, weshalb unter anderem die aktive Kühlfunktion integriert wurde.

5.5 Nicht-funktionale Anforderungen

Mit Bezug auf die Bedienbarkeit soll das ganze System intuitiv bedienbar sein.

Bei der Leistung soll mit vollgeladenem Akku der Roboter 3 Stunden kühlen und eine Stunde fahren können.

In Betrachtung der Sicherheit und Zuverlässigkeit sind sämtliche Elektronikkomponenten vor Witterung, Sonneneinstrahlung und Überhitzung geschützt. Stromschläge werden dadurch vermieden, dass nur Niederspannungskomponenten ohne Schwierigkeiten mit blossen Fingern erreichbar sein sollen. Ein Hauptschalter soll ausserdem in der Lage sein, die Stromzufuhr zum gesamten System zu unterbrechen.

6 Schnittstellen

6.1 Übersicht

Da in diesem Projekt nur ein Mikrokontroller verwendet wird, halten sich die Hardwareschnittstellen in Grenzen.

6.2 Hardwareschnittstellen

Als Hardwareschnittstellen kommen lediglich Jumper-Kabel zum Einsatz, sowohl für die Datenübertragung als auch für die Versorgungsspannung. Dafür soll möglicherweise ein Kabelbaum zum Einsatz kommen.

Wir führen hier auch noch die Motorentreiber auf. Sie nutzen ein PWM-Signal, durch welches sie die ihnen zur Verfügung gestellte 24V Spannung in unterschiedlicher Stärke an die Motoren weitergeben.

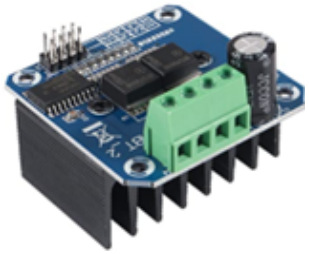

Abbildung 6.1: Motorentreiber

6.3 Softwareschnittstellen

Die Kommunikation zwischen den Sensoren und dem Arduino soll über das UART- und das I2C-Protokoll laufen. Das I2C-Protokoll benutzt die SCL- und SDA-Pins (A4 und A5 auf dem Arduino UNO), welche vom Kompass-Modul benutzt werden soll.

Das GPS-Modul und das Bluetooth-Modul sollen über ein UART-Protokoll mit dem Arduino kommunizieren. Dafür sind alle Pins mit PWM-Funktion auf dem Arduino ausreichend.

Die Motorentreiber erhalten ein 5V PWM-Signal, welches sie in ihrer Ausgangsspannung, in unserem Fall 24V, weitergeben.

7 Mechanik

7.1 Mechanische Struktur

Das Grundgerüst des Beercoolers besteht aus 2 MDF-Holzplatten, welche über Stützen miteinander verschraubt werden, sodass zwischen ihnen ein Hohlraum entsteht. In diesem Hohlraum ist Platz für sämtliche Elektronik, wie den Motoren, den Mikrokontroller und dem Akku. Auf der Rückseite des Roboters befindet sich an der oberen Platte ein Rad mit neutraler Lenkung, am vorderen Ende befinden sich zwei angetriebene Räder. Zuerst war geplant mit 4 starren angetriebenen Rädern zu fahren. Da dieses System aber im beladenen Zustand schlecht lenkbar war, haben wir uns für die Lösung mit einem neutralen Lenkrad entschieden.

Auf der oberen Platte befinden sich 4 Führungen, in welchen die Kühlbox zu platzieren ist. Die elektrische Verbindung ist in die Platte eingearbeitet und stellt eine einwandfreie Verbindung sicher. Die Kühlbox lässt sich dann auf Heizen oder Kühlen einstellen.

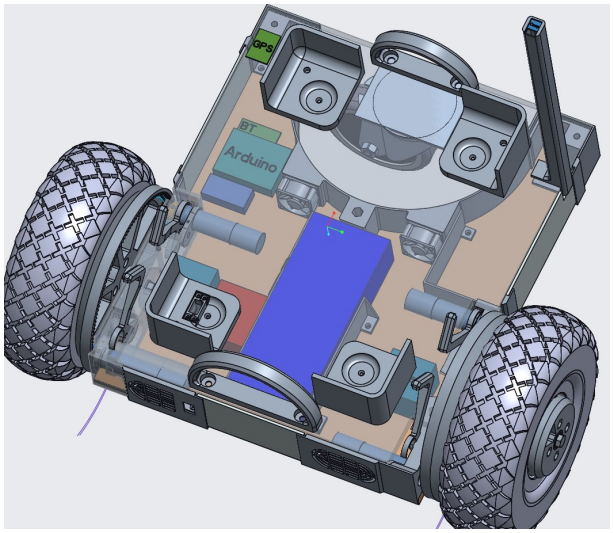

Abbildung 7.1: Aufbau im CAD

7.2 Führungen / Getriebe

Die Fortbewegung wird über 4 Brushless DC-Motoren von Faulhaber und ein Zahnradpaar gewährleistet. Der Kraftschluss geschieht über ein direkt auf der Motorwelle montiertes kleines, und über ein innenverzahntes grosses Zahnrad, welches mit dem Rad verschraubt ist.

Abbildung 7.2: 3D gedruckte Zahnräder

Die Motoren sind zudem über einen Hebelmechanismus zurückziehbar, wodurch es möglich ist den Roboter auch ohne den Einsatz der Motoren so reibungsfrei wie möglich zu bewegen.

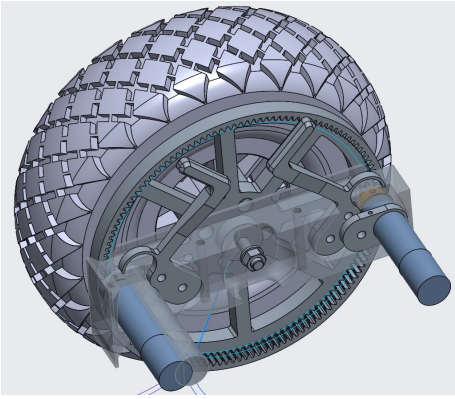

Abbildung 7.3: Rad mit Aufnahme und Hebelmechanismus

Um eine reibungsfreie Bewegung zu Erreichen, wurden bei den angetriebenen Rädern Kugellager eingebaut. Hier ist es wichtig, auf den korrekten Einbau der Kugellager zu achten, damit durch den Zusammenbau der Baugruppe keine überhöhten axiale Kräfte auf das Lager wirken. Dies kann zum Versagen der Lager führen. Weil wir zu Beginn die Lager nicht richtig abgestützt haben, hat sich ein Kugellager während den Testversuchen in Einzelteile zerlegt. Dieses Problem konnte mit einer Hülse, welche die Innenringe der Kugellager abstützt, gelöst werden, so dass keine axiale Last durch das Anziehen der Schraube auf den Kugellagern lastet.

8 Sensoren

8.1 Neo 7M

Das Neo 7m Modul ist ein sehr kleiner GPS-Sensor, welcher mittels einer Antenne die eigenen GPS-Koordinaten auslesen kann. Es wird in unserem Fall über eine UART-Schnittstelle angesteuert.

Die Genauigkeit ist leider stark abhängig vom Gelände, dem Wetter und ob die Antenne "Sichtkontakt" zum Himmel hat. Ausserdem wird der Empfang von GPS-Koordinaten nicht funktionieren, wenn das Modul sich nicht unter freiem Himmel befindet.

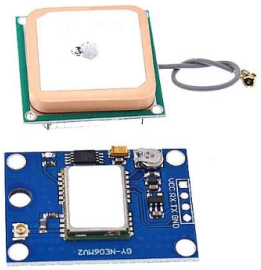

Abbildung 8.1: GPS Modul

8.2 QMC5883L

Das QMC5883 Kompass Modul ist ein Magnetometer, welches mit der richtigen Bibliothek in der Lage ist, einen Kompass zu simulieren. So können Azimut und Himmelsrichtungen direkt ausgelesen werden. Es ist allerdings Vorsicht geboten, der Kompass ist sehr empfindlich auf Magnetismen und Metalle. Derartige Einflüsse können die Zuverlässigkeit der Daten des Kompasses stark beeinflussen.

Abbildung 8.2: Kompass Modul

8.3 HC-05

Das HC-05 ist ein Bluetooth-Modul mit sowohl Slave- als auch Masterfähigkeiten. Es soll die Kommunikation sowie Datenübertragung zwischen dem Handy und dem Arduino ermöglichen.

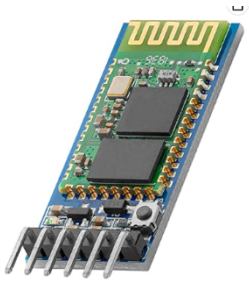

Abbildung 8.3: Bluetooth Modul

9 Aktoren

9.1 Motoren

Als einzige steuerbare Aktoren dienen uns DC-Motoren von Faulhaber mit einem integrierten Reduktionsgetriebe. Es werden 2 pro Rad zum Einsatz kommen, um ausreichend Drehmoment erzielen zu können. Bei der ursprünglichen Variante sollten alle vier Räder je mit einem Motor angetrieben werde. Um beim finalen Design die gleiche Ausgangsleistung zu erhalten haben wir die zwei Motoren der entfernten Rädern den anderen zwei hinzugefügt.

Abbildung 9.1: Motor

9.2 Peltier Element

Das Peltier-Element in der Kühlbox erzeugt eine Temperaturdifferenz, sobald man eine Spannung daran anlegt. Mit 2 Lüftern und 2 Kühlelementen, welche in der gekauften Kühlbox bereits vorhanden sind, ist es möglich sowohl zu heizen als auch zu kühlen.

Da die Endtemperatur von der Aussentemperatur abhängt, lässt sich hier lediglich eine Temperaturdifferenz sinnvoll definieren, welche sich auf 10-15 Grad Celsius festlegen lässt.

Abbildung 9.2: Kühlbox

10 Elektronik

10.1 Messwertverarbeitung

Die Messwertverarbeitung findet direkt auf dem Mikrokontroller statt.

10.2 Leistungsteil

Als Leistungsteil soll ein Step-Up-Modul die Spannung des 4S-LiPo's (Nennspannung 14.8 V) auf 24 V hoch transformieren, um die Motoren mit der maximal verträglichen Spannung zu versorgen.

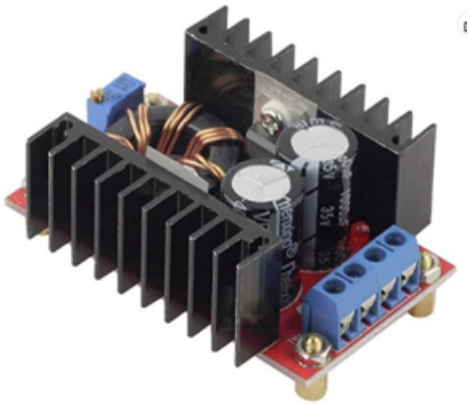

Abbildung 10.1: Step-Up

Des Weiteren soll auch ein Step-Down Modul zum Einsatz kommen, um die 5V Versorgerspannung für den Microkontroller bereitzustellen.

Abbildung 10.2: Step-Down

Da die Kühlbox auf die Nennspannung einer Autobatterie ausgelegt war, welche durchschnittlich 12-14 V liefert, haben wir von einer weiteren Spannungstransformation abgesehen und die Kühlbox mit der Batteriespannung versorgt.

11 Informationsverarbeitung

11.1 Digitalrechner

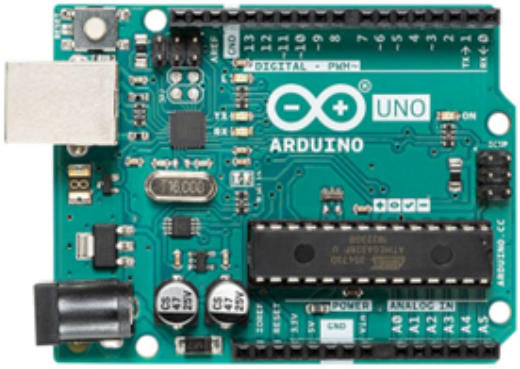

Abbildung 11.1: Arduino

Auf dem Arduino ist ein Atmega328 Mikroprozessor verbaut. Dieser ist auf einer Platine beschaltet. Das Programm kann mit der Arduino IDE direkt über eine serielle Schnittstelle auf den Arduino geladen werden. Es wird also kein externes Kompiliergerät benötigt. Mit der Arduino IDE lassen sich auf dem Arduino UNO rund 20 Pins programmieren. Bei einem Arduino UNO sind folgende Anschlüsse vorhanden:

- 13x digital Pins
- 3x Timer
- 6x PWM Pins
- SDA und SCL Pins
- 6x analog Pins
- MISO MOSI Pins

Benutzt werden hierbei 5 der PWM-Pins, 4 der normalen Digital-Pins, die SDA- und SCL-Pins. Der USB-Anschluss wird lediglich zur Übertragung des Programms und zur Überwachung der korrekten Funktion während der Entwicklung benötigt.

11.2 Steuerung

Die Steuerung des Roboters erfolgt lediglich über die Motoren, welche mit unterschiedlichen PWM-Signalen neben vorwärtsn nach links und nach rechts gesteuert werden.

11.3 Regelung

Die Regelung erfolgt über eine Kombination aus Kompass, den GPS-Koordinaten des Handys und den GPS-Koordinaten des GPS-Moduls, welches mit dem Arduino verbunden ist. Mittels der 2 Koordinatensets soll die Distanz zwischen den beiden Punkten errechnet werden, der Kompass gibt schliesslich die Richtung an, in welche es sich zu drehen gilt. Sobald die Distanz unter einen gewissen Schwellenwert fällt, wird die Position des Handy-GPS wieder überprüft und der Prozess beginnt von vorne.

Das Fundament für die Berechnung bildet die Natur des GPS-Koordinatensystems. Dieses hat seinen «Nullpunkt» am Schnittpunkt zwischen Äquator und dem Null-Meridian, der Nord-Süd

Achse durch Greenwich, Vereinigtes Königreich. Ein GPS-Standpunkt ist stets in Latitude und Longitude unterteilt. Die Latitude gibt den Winkel zwischen der Äquatorlinie und dem Standpunkt an und die Longitude gibt den Winkel zwischen Nullmeridian und dem Standpunkt an.

Dank der Haversine Formel

$$a = sin^2\left(\frac{p_2.lat - p_1.lat}{2}\right) + cos\left(p_1.lat\right) * cos\left(p_2.lat\right) * sin^2\left(\frac{p_2.lon - p_1.lon}{2}\right)$$

$$b = 2 * atan2\left(\sqrt{a}, \sqrt{1-a}\right)$$

$$d = R * b$$

lässt sich die Distanz zwischen 2 GPS-Punkten relativ leicht berechnen. R ist der Erdradius von 6371 km.

Über folgende Formel:

$$\beta = atan2(sin\left(p_2.lon - p_1.lon\right) * cos\left(p_2.lat\right), cos\left(p_1.lat\right) * sin\left(p_2.lat\right) - sin\left(p_1.lat\right) * cos\left(p_2.lat\right) * cos\left(p_2.lon - p_1.lon\right))$$

lässt sich indes der Winkel zwischen der Nord-Süd-Achse durch Punkt 1 und der Achse, welche durch die beiden Punkte geht, berechnen. Man nennt dies die Peilrichtung, oder Bearing.

Wenn man nun den Winkel, resp. die eigene Ausrichtung gegenüber dem Nordpol kennt, zum Beispiel durch einen Kompass, kann man diesen Winkel von der Peilrichtung abziehen und erhält den Winkel, um den man sich drehen muss, um in Richtung des zweiten Punktes zu zeigen.

12 Software

12.1 Softwareverweise

Als Grundlage haben wir uns für die Software des Projekts von Hackster.io bedient. Es handelt sich hierbei um zwei Herren, die dasselbe Projekt bereits realisiert hatten und eine Dokumentation dazu online zur Verfügung gestellt haben. Darunter auch ihre Software. Die Idee war, das Projekt mit dieser Vorlage zum Laufen zu bringen und von dort aus Verbesserungen vorzunehmen.

Nachdem Anfangs die Bibliotheken der Arduino IDE für den Kompass und das GPS-Modul ausgetauscht werden mussten, stellte uns die Blynk-App, resp. Blynk Plattform vor das nächste Problem, indem sie die Möglichkeit der Bluetooth Konnektivität seit Mai 2022 aus dem Sortiment genommen haben. Wir mussten also eine Legacy-Version der App suchen, welche funktionierte. Das klappte auch, bis wir kurz vor dem Jahreswechsel freundlicherweise von der Blynk-App darauf hingewiesen wurden, dass die Legacy-Plattform, welche für das Login in der App benötigt wird, per 01. Januar 2023 vom Netz genommen wird.

Nach anfänglicher Beunruhigung, ob das Projekt mit dieser Steuerung zu finalisieren sei, haben wir es jedoch geschafft, auf Hetzner.de einen eigenen Legacy-Server einzurichten, wodurch wir unseren Stand zu diesem Zeitpunkt nicht verwerfen mussten.

13 Benutzerinterface

13.1 Layout

Das Layout beruht auf der gegebenen Blynk Oberfläche und einer Auswahl an verfügbaren Widgets. Wir mussten dabei auf eine Legacy Version der App zurückgreifen, da die Möglichkeit zur Verbindung mit Bluetooth eingestellt wurde.

Abbildung 13.1: Layout in der Blynk App

Es soll mindestens ein Killswitch, eine Bluetooth-Verbindung, ein GPS-Stream und ein Terminal darin enthalten sein.

13.2 Funktionen

Über den Killswitch soll der Roboter in den «Fahren»-Modus versetzt werden können und diesen wieder verlassen. Das Bluetooth- und GPS-Widget dienen lediglich dem Informationsaustausch zwischen dem Arduino und der App. Das Terminal soll schliesslich die Möglichkeit bieten, GPS-Koordinaten direkt dem Arduino zuzuführen, zu welchen er fahren soll.

14 Schlussbemerkungen

Hier ist deutlich hervorzuheben, dass uns eines unserer Hauptziele, das autonome Folgen, in der Finalisierung des Projektes nicht gelungen ist. Der Roboter fährt zwar selbstständig, aber nicht autonom per Appsteuerung dorthin, wo er hin soll. Zudem aktualisiert sich die Position des Smartphone-GPS auch nicht selbstständig. Er fährt zum Punkt, an dem die Verbindung hergestellt wurde und bleibt dann aber dort stehen. Wenn man ihn versetzt, fährt er nur wieder an den Punkt zurück.

Um uns den Transport und die Kontrolle zu erleichtern, entstand aus einem Versuch der Blynk-App zu entkommen ein Joystick in der Dabble-App, welcher hervorragend funktioniert und mit welchem wir den Betrieb bis auf weiteres empfehlen würden. Leider erlaubt die Dabble App nicht mehr als seine Kommunikation über das UART-Protokoll. Da unser GPS-Modul auf einem für UART ausgelegten Board zu uns kam, hätten wir ohne Erfolgsgarantie die Pins am Chip umlöten müssen, was uns ein zu grosses Risiko darstellte.

Auf das Design des Roboters sind wir hingegen stolz. Wir konnten sehr viel selbst mit den 3D-Drucken herstellen, was alle 3 Teilnehmer zu ihren Hobbys zählen, und hatten auch sonst viel Spass beim mechatronischen Zusammenbauen des Fahrgestells.

Abschliessend lässt sich sagen, dass die prognostizierte Schwachstelle der Software sich bewahrheitet hat.

Die Blynk App, auf die wir aufgrund der Vorlage in gewissem Masse gebunden waren, war schlicht nicht stabil genug für unsere Voraussetzungen. Zum einen hatte sie stets Probleme die Bluetooth-Verbindung zuverlässig aufrecht zu erhalten, zum anderen ist die genaue Funktionsweise der App und ihrer Bibliotheken schwer zu durchschauen. Eventuell wäre auch ein Wechsel auf eine andere Datenübertragungsmethode wie ein Wifi-Modul eine Option gewesen. Hier hätten wir früher reagieren können, als wir bemerkten, dass es mit der Blynk App nicht wie gewünscht funktionieren wird. Als Lösung hätte man womöglich einen zweiten Arduino mit Bluetooth und GPS-Modul verwenden können. Diesen hätte dann der Benutzer auf sich getragen und der Arduino hätte über das BT-Modul mit dem Arduino des Roboters kommunizieren können.

Die Sensoren stellten uns vor ein weiteres grosses Problem, da der Kompass sich als ausgesprochen sensibel herausstellte und das GPS-Modul an Präzision zu wünschen übrig liess. Eine anfängliche Idee das Ganze über einen Apple-Airtag ähnliches Gerät aufzusetzen oder die Followfunktion mittels Objekterkennung zu gestalten, hätte uns eventuell zu mehr Erfolg verholfen. Die Elektronik und Mechanik von diesem Projekt funktionieren aber sehr gut. Der Roboter fährt mit dem Joystick sehr zuverlässig. Die Konstruktion und Planung des Konzepts liessen sich, bis auf die Softwareschwachstelle, gut umsetzten.

Wir sind trotzdem stolz auf unsere Leistung und haben es am Ende geschafft, zumindest die Rückenbelastung durch das Schleppen einer Kühlbox und anderer Gegenstände zu mindern. Die Motoren verfügen über ausreichend Drehmoment, um den voll beladenen Roboter über eine Wiese und abschüssiges Gelände hochzubewegen und die Kühlleistung reicht aus um 3 Stunden Kühlung zu gewährleisten.

Quellenverzeichnis

Amazon. (2023a). Bild BT-Modul [Online; Abgerufen 09.01.2023]. https://m.media-amazon.com/images/I/71QXqTfBE5L._SL1500_.jpg

Amazon. (2023b). Bild GPS-Modul [Online; Abgerufen 09.01.2023]. https://m.media-amazon.com/images/I/41GUl9XQ3dL._AC_.jpg

Amazon. (2023c). Bild Kompass-Modul [Online; Abgerufen 09.01.2023]. https://m.media-amazon.com/images/I/51SzxbmwiLL._SL1010_.jpg

Amazon. (2023d). Bild Motor-Driver [Online; Abgerufen 09.01.2023]. https://m.media-amazon.com/images/I/71LH36g3MnL._SL1500_.jpg

Amazon. (2023e). Bild Step-Down [Online; Abgerufen 09.01.2023]. https://m.media-amazon.com/images/I/71F4Yi0PFCL._AC_SL1500_.jpg

Amazon. (2023f). Bild Step-Up [Online; Abgerufen 09.01.2023]. https://m.media-amazon.com/images/I/519GtclDOUL._AC_SL1000_.jpg

BlynkMobilde. (2022). *Blynk-Android-App* [Online; Abgerufen 09.01.2023]. https://github.com/BlynkMobile/Blynk-Android-App

ddf3d.com. (2023). Karabiner STL-Teile [Online; Abgerufen 10.01.2023]. https://www.thingiverse.com/thing:1819242

Distrelec. (2023). Bild Arduino UNO [Online; Abgerufen 09.01.2023]. https://www.distrelec.ch/de/mikrocontroller-board-uno-arduino-a000066/p/11038919

Hacker Shack. (2017). *Make an Autonomous "Follow Me"Cooler* [Online; Abgerufen 09.01.2023]. https://www.hackster.io/hackershack/make-an-autonomous-follow-me-cooler-7ca8bc

item Industrietechnik GmbH. (2023). Lenkrolle D125 140x110 ESD CAD-Datei [Online; Abgerufen 10.01.2023]. https://www.traceparts.com/de/product/item-industrietechnik-gmbh-lenkrolle-d125-140x110-esd?CatalogPath=TRACEPARTS%3ATP05010003&Product=30-07072021-084271&PartNumber=0.0.667.23

Kai. (2023). LM2596HV Buck Converter Housing STL-Modell [Online; Abgerufen 10.01.2023]. https://www.printables.com/de/model/219074-lm2596hv-buck-converter-housing

Kaufland. (2023). Bild Kühlbox [Online; Abgerufen 09.01.2023]. https://www.kaufland.de/product/438115140/?id_unit=385239511000

Martinfun1987. (2023). Arduino Uno R3 Clip Base Holder Stand STL-Modell [Online; Abgerufen 10.01.2023]. https://www.thingiverse.com/thing:222680

Peterkn2001. (2022). *blynk-server* [Online; Abgerufen 09.01.2023]. https://github.com/Peterkn2001/blynk-server

Tisserand, K. (2023). CAD-Modell von Bollerwagenrad [Online; Abgerufen 10.01.2023]. https://grabcad.com/library/roue-gonflable-260-x-85-1

Ehrlichkeitserklärung

Hiermit erklären wir, die vorliegende Projektarbeit, selbständig und nur unter Benutzung der angegebenen Quellen verfasst zu haben. Die wörtlich oder inhaltlich aus den aufgeführten Quellen entnommenen Stellen sind in der Arbeit als Zitat bzw. Paraphrase kenntlich gemacht. Diese Projektarbeit ist noch nicht veröffent-licht worden. Sie ist somit weder anderen Interessierten zugänglich gemacht noch einer anderen Prüfungs-behörde vorgelegt worden.

Muttenz, 10. Januar 2023

Name: Max Knauber

Unterschrift:

Name: Matthias Gass

Unterschrift:

Name: Fabian Schenker

Unterschrift:

Abbildungsverzeichnis

Tabellenverzeichnis

A Anhang / Ressourcen

A.1 Impressum

Datum der Erstellung der Dokumentation: Herbst und Winter 2022/2023

A.2 Quellcode

Listing 1: Programm des Roboters

```
#define BLYNK_USE_DIRECT_CONNECT
#define BLYNK_PRINT Serial

// Imports
#include <Wire.h>
#include <QMC5883LCompass.h>
#include <Servo.h>
#include <SoftwareSerial.h>
#include <BlynkSimpleSerialBLE.h>
#include <TinyGPSPlus.h>
#include "./CoolerDefinitions.h"

// GPS
TinyGPSPlus gps;

// Master Enable
bool enabled = false;

// Serial components
SoftwareSerial bluetoothSerial(BLUETOOTH_TX_PIN, BLUETOOTH_RX_PIN);
SoftwareSerial nss(GPS_TX_PIN, 255);// TXD to digital pin 6

/* Compass */
QMC5883LCompass mag; //Max

GeoLoc checkGPS() {
  Serial.println("Reading onboard GPS: ");
  bool newdata = false;
  unsigned long start = millis();
  while (millis() - start < GPS_UPDATE_INTERVAL) {
    if (feedgps())
      newdata = true;
  }
  if (newdata) {
    return gpsdump(gps);
  }

  GeoLoc coolerLoc;
  coolerLoc.lat = 0.0;
```

```
  coolerLoc.lon = 0.0;

  return coolerLoc;
}

// Get and process GPS data
GeoLoc gpsdump(TinyGPSPlus &gps) {
  float flat, flon;

  GeoLoc coolerLoc;
  coolerLoc.lat = gps.location.lat();
  coolerLoc.lon = gps.location.lng();

  Serial.print(coolerLoc.lat, 7); Serial.print(", ");
  Serial.println(coolerLoc.lon, 7);

  return coolerLoc;
}

// Feed data as it becomes available
bool feedgps() {
  while (nss.available()) {
    if (gps.encode(nss.read()))
      return true;
  }
  return false;
}

//enabling joystick
bool isButtonPressed;
BLYNK_WRITE(V5) {
  int buttonState = param.asInt();
  if(buttonState == 1){
    isButtonPressed = true;
    Serial.print("Joystick enabled");
  } else {
    isButtonPressed = false;
    Serial.print("Joystick disabled");
  }
}

//Joystick
BLYNK_WRITE(V0) {
  if (!isButtonPressed) {
    return;
  }

    int x = param[0].asInt(); // x-coordinate of joystick
    int y = param[1].asInt(); // y-coordinate of joystick
```

```
    // Set speeds to 0 if y < 128
    if (y < 128) {
      analogWrite(MOTOR_A_EN_PIN, 0);
      analogWrite(MOTOR_B_EN_PIN, 0);
      return;
    }

    // Calculate full speed
    int fullSpeed = y - 128;
    // Constrain fullSpeed to the range [0, 128]
    fullSpeed = constrain(fullSpeed, 0, 128);

    // Calculate autoSteer values
    float autoSteerA = 1.0;
    float autoSteerB = 1.0;

    if (x < 128) {
      autoSteerA = 1-((128 - x) / 128.0);
    } else if (x > 128) {
      autoSteerB = 1-((x - 128) / 128.0);
    }

    // Calculate speeds for motors A and B
    int speedA = fullSpeed * autoSteerA;
    int speedB = fullSpeed * autoSteerB;

    // Constrain speeds to the range [0, 255]
    speedA = constrain(speedA, 0, 128);
    speedB = constrain(speedB, 0, 128);

    // Set motor speeds
    analogWrite(MOTOR_A_EN_PIN, speedA);
    analogWrite(MOTOR_B_EN_PIN, speedB);

    // Output values to serial
    Serial.print("x: "); Serial.println(x);
    Serial.print("y: "); Serial.println(y);
    Serial.print("fullSpeed: "); Serial.println(fullSpeed);
    Serial.print("autoSteerA: "); Serial.println(autoSteerA);
    Serial.print("autoSteerB: "); Serial.println(autoSteerB);
    Serial.print("speedA: "); Serial.println(speedA);
    Serial.print("speedB: "); Serial.println(speedB);
}

// Killswitch Hook
BLYNK_WRITE(V1) {

  int buttonState = param[0].asInt();
  if(buttonState == 1){
    enabled = true;
```

```
    Serial.print("autodrive enabled");
  } else {
    enabled = false;
    Serial.print("autodrive disabled");
    //Stop the wheels
    stop();

  }
}

//displayCompassDetails
BLYNK_WRITE(V4) {
  displayCompassDetails();
}

// GPS Streaming Hook
BLYNK_WRITE(V2) {
  nss.listen();
  GeoLoc coolerLoc = checkGPS();
  bluetoothSerial.listen();
  GeoLoc phoneLoc;
  phoneLoc.lat = param[0].asFloat();
  phoneLoc.lon = param[1].asFloat();
  Serial.print("phoneLoc.lat: "); Serial.print(phoneLoc.lat,7);
  Serial.print(" , "); Serial.print("phoneLoc.lon: ");
  Serial.println(phoneLoc.lon,7);
  Serial.print("distancev2: ");
  Serial.println(geoDistance(coolerLoc,phoneLoc));
  if(enabled == true && geoDistance(coolerLoc,phoneLoc) > 10){
    driveTo(phoneLoc, GPS_STREAM_TIMEOUT);
  }
}

// Terminal Hook
BLYNK_WRITE(V3) {
  Serial.print("Received Text: ");
  Serial.println(param.asStr());

  String rawInput(param.asStr());
  int colonIndex;
  int commaIndex;

  do {
    commaIndex = rawInput.indexOf(',');
    colonIndex = rawInput.indexOf(':');

    if (commaIndex != -1) {
      String latStr = rawInput.substring(0, commaIndex);
      String lonStr = rawInput.substring(commaIndex+1);
```

```
      if (colonIndex != -1) {
         lonStr = rawInput.substring(commaIndex+1, colonIndex);
      }

      float lat = latStr.toFloat();
      float lon = lonStr.toFloat();

      if (lat != 0 && lon != 0) {
        GeoLoc waypoint;
        waypoint.lat = lat;
        waypoint.lon = lon;

        Serial.print("Waypoint␣found:␣"); Serial.print(lat);
        Serial.println(lon);
        driveTo(waypoint, GPS_WAYPOINT_TIMEOUT);
      }
    }

    rawInput = rawInput.substring(colonIndex + 1);

  } while (colonIndex != -1);
}

void displayCompassDetails(void)
{

  char myArray[3];
  Serial.println("--------------------------------------");
  mag.read();
  Serial.print  ("GetX:␣␣␣␣"); Serial.print(mag.getX());
  Serial.println("␣uT");
  Serial.print  ("GetY:␣␣␣␣"); Serial.print(mag.getY());
  Serial.println("␣uT");
  Serial.print  ("GetZ:␣␣␣␣"); Serial.print(mag.getZ());
  Serial.println("␣uT");
  Serial.print  ("Azimuth:␣␣␣"); Serial.print(mag.getAzimuth());
  Serial.println("␣degrees");
  Serial.print  ("Bearing:␣␣␣");
  Serial.print(mag.getBearing(mag.getAzimuth()));
  Serial.println("␣");
  mag.getDirection(myArray,mag.getAzimuth());
  Serial.print  ("Direction:␣␣␣"); Serial.print(myArray[0]);
  Serial.print(myArray[1]); Serial.print(myArray[2]);
  Serial.println("--------------------------------------");
  Serial.println("");

  delay(500);

}
```

```
#ifndef DEGTORAD
#define DEGTORAD 0.0174532925199432957f
#define RADTODEG 57.295779513082320876f
#endif
//coolerloc, loc
float geoBearing(struct GeoLoc &a, struct GeoLoc &b) {
  float y = sin(b.lon-a.lon) * cos(b.lat);
  float x = cos(a.lat)*sin(b.lat) -
    sin(a.lat)*cos(b.lat)*cos(b.lon-a.lon);
  return atan2(y, x) * RADTODEG;
}
//a = cooler, b = phone
float geoDistance(struct GeoLoc &a, struct GeoLoc &b) {
  const float R = 6371000; // km
  float p1 = a.lat * DEGTORAD;
  float p2 = b.lat * DEGTORAD;
  float dp = (b.lat-a.lat) * DEGTORAD;
  float dl = (b.lon-a.lon) * DEGTORAD;

  float x = sin(dp/2) * sin(dp/2) +
    cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
  float y = 2 * atan2(sqrt(x), sqrt(1-x));

  return R * y;
}

float geoHeading() {
  mag.read();//Max

  float heading = mag.getAzimuth();


  // Offset
  heading -= -COMPASS_OFFSET;

  // Correct for when signs are reversed.
  if(heading < 0){
    heading += 360;
  }

  // Check for wrap due to addition of declination.
  if(heading > 360){
    heading -= 360;
  }

  // Map to -180 - 180
  while (heading < -180) {
      heading += 360;

  }
```

```
  while (heading >  180){
      heading -= 360;
  }

  //return headingDegrees;
  return heading;
}

void setSpeedMotorA(int speed) {
  digitalWrite(MOTOR_A_IN_1_PIN, HIGH); //Max
  digitalWrite(MOTOR_A_IN_2_PIN, HIGH); //Max

  // set speed to 200 out of possible range 0~255
  analogWrite(MOTOR_A_EN_PIN, speed + MOTOR_A_OFFSET);
}

void setSpeedMotorB(int speed) {
  digitalWrite(MOTOR_B_IN_1_PIN, HIGH); //Max
  digitalWrite(MOTOR_B_IN_2_PIN, HIGH); //Max

  // set speed to 200 out of possible range 0~255
  analogWrite(MOTOR_B_EN_PIN, speed + MOTOR_B_OFFSET);
}

void setSpeed(int speed){
  // this function will run the motors in both directions
  // at a fixed speed
  // turn on motor A
  setSpeedMotorA(speed);

  // turn on motor B
  setSpeedMotorB(speed);
}

void stop() {
  // now turn off motors
  digitalWrite(MOTOR_A_IN_1_PIN, LOW); //Max
  digitalWrite(MOTOR_A_IN_2_PIN, LOW); //Max
  digitalWrite(MOTOR_B_IN_1_PIN, LOW); //Max
  digitalWrite(MOTOR_B_IN_2_PIN, LOW); //Max
  analogWrite(MOTOR_A_EN_PIN, 0);
  analogWrite(MOTOR_B_EN_PIN, 0);
  Serial.println("Motors stopped!");//Max
}

void drive(int distance, float turn) {

  int fullSpeed = 50;
  int stopSpeed = 0;
```

```
  // drive to location
  int s = fullSpeed;
  if ( distance < 15 ) {
    int wouldBeSpeed = s - stopSpeed;
    wouldBeSpeed *= distance / 15.0f;
    s = stopSpeed + wouldBeSpeed;
  }

  int autoThrottle = constrain(s, stopSpeed, fullSpeed);
  autoThrottle = 50;

  float t = turn;
  while (t < -180) t += 360;
  while (t >  180) t -= 360;

  Serial.print("turn: ");
  Serial.println(t);
  Serial.print("original: ");
  Serial.println(turn);

  float t_modifier = (180.0 - abs(t)) / 180.0;
  float autoSteerA = 1;
  float autoSteerB = 1;

  if (t > 0) {
    autoSteerB = t_modifier;
  } else if (t < 0){
    autoSteerA = t_modifier;
  }

  Serial.print("steerA: "); Serial.println(autoSteerA);
  Serial.print("steerB: "); Serial.println(autoSteerB);

  int speedA = (int) (((float) autoThrottle) * autoSteerA);
  int speedB = (int) (((float) autoThrottle) * autoSteerB);

  setSpeedMotorA(speedA);
  setSpeedMotorB(speedB);
}

void driveTo(struct GeoLoc &loc, int timeout) {
  nss.listen();
  GeoLoc coolerLoc = checkGPS();
  bluetoothSerial.listen();


  if (coolerLoc.lat != 0 && coolerLoc.lon != 0 &&
    enabled==true && geoDistance(coolerLoc,loc) > 10.0) {
    float d = 0;
    //Start move loop here
```

```
    do{
      nss.listen();
      coolerLoc = checkGPS();
      bluetoothSerial.listen();

      d = geoDistance(coolerLoc, loc);
      float t = geoBearing(coolerLoc, loc) - geoHeading();

      Serial.print("Remote gps: "); Serial.print(loc.lat, 7);
      Serial.print(", "); Serial.println(loc.lon, 7);
      Serial.print("Onboard gps: ");
      Serial.print(coolerLoc.lat, 7);
      Serial.print(", "); Serial.println(coolerLoc.lon, 7);

      Serial.print("Distance: ");
      Serial.println(geoDistance(coolerLoc, loc));

      Serial.print("Bearing: ");
      Serial.println(geoBearing(coolerLoc, loc));

      Serial.print("heading: ");
      Serial.println(geoHeading());

      drive(d, t);
      timeout -= 1;
    } while (d > 10.0 && enabled == true && timeout>0);

    stop();
    return;
  }
  return;
}

void setupCompass() {
  mag.init();
  mag.setCalibration(-1401, 803, -1243, 1028, -543, 561);
  displayCompassDetails();
}

void setup()
{
  // Motor pins
  pinMode(MOTOR_A_EN_PIN, OUTPUT);
  pinMode(MOTOR_B_EN_PIN, OUTPUT);
  pinMode(MOTOR_A_IN_1_PIN, OUTPUT); //Max
  pinMode(MOTOR_A_IN_2_PIN, OUTPUT); //Max
  pinMode(MOTOR_B_IN_1_PIN, OUTPUT); //Max
  pinMode(MOTOR_B_IN_2_PIN, OUTPUT); //Max

  pinMode(LED_BUILTIN, OUTPUT);
```

```
  //Debugging via serial
  Serial.begin(4800);

  Serial.println("This code is doing something in setup");

  //GPS
  nss.begin(9600);
  //Bluetooth
  bluetoothSerial.begin(9600);
  Blynk.begin(bluetoothSerial, auth);

  setupCompass();

  Serial.println("This code has gone through setup");
}

// Testing
void testDriveNorth() {
  float heading = geoHeading();
  int testDist = 10;
  Serial.println(heading);

  while(!(heading < 5 && heading > -5)) {
    drive(testDist, heading);
    heading = geoHeading();
    Serial.println(heading);
    delay(500);
  }

  stop();
}

void loop()
{
  Blynk.run();
  if (enabled == true){
    digitalWrite(LED_BUILTIN, HIGH);
  }
  else digitalWrite(LED_BUILTIN, LOW);
}
```

Listing 2: Definitionen für das Programm

```
// Blynk Auth
char auth[] = "iD13fj_UbiBH4_lj0XCe7pKj9BhGik2y";

// Pin variables
#define SERVO_PIN 3

#define GPS_TX_PIN 6
```

```
#define BLUETOOTH_TX_PIN 10
#define BLUETOOTH_RX_PIN 11

//Speed control Pins
#define MOTOR_A_EN_PIN 5
#define MOTOR_B_EN_PIN 9
#define MOTOR_A_IN_1_PIN 7
#define MOTOR_A_IN_2_PIN 8 //the 2's are forward
#define MOTOR_B_IN_1_PIN 12
#define MOTOR_B_IN_2_PIN 4

// If one motor tends to spin faster than the other, add offset
#define MOTOR_A_OFFSET 0
#define MOTOR_B_OFFSET 0

// You must then add your 'Declination Angle' to the compass,
// which is the 'Error' of the magnetic field in your location.
// Find yours here: http://www.magnetic-declination.com

#define DECLINATION_ANGLE 0.053f

// The offset of the mounting position to true north
// It would be best to run the /examples/magsensor
// sketch and compare to the compass on your smartphone
#define COMPASS_OFFSET 90.0f //degrees

// How often the GPS should update in MS
// Keep this above 1000
#define GPS_UPDATE_INTERVAL 1000 //maybe put this to like 5000?

// Number of changes in movement to timeout for GPS streaming
// Keeps the cooler from driving away if there is a problem
#define GPS_STREAM_TIMEOUT 18 //was 18

// Number of changes in movement to timeout for GPS waypoints
// Keeps the cooler from driving away if there is a problem
#define GPS_WAYPOINT_TIMEOUT 45

// Definitions (don't edit these)
struct GeoLoc {
  float lat;
  float lon;
};
```