



UNIVERSITÄT AUGSBURG

Fakultät für Angewandte Informatik

Praktikum für Produktionstechnik

Entwicklungsprozess eines lenkbaren Fahrzeuges TEGGLA

vorgelegt von:	Jonas Wilfert: 1541778 Niklas Paprotta: 1543350 Johannes Evertz: 1463672 Marcel Khodabakhsh: 1333430
eingereicht am:	30.07.2019
Studiengang:	B.Sc. Ingenieurinformatik
Anfertigung am Lehrstuhl:	Produktionsinformatik
	Fakultät für Angewandte Informatik
Verantwortlicher Professor:	Prof. Dr. Ing. Johannes Schilp
Wissenschaftliche Betreuer:	M.Sc. Michael Aumüller M.Sc. Fabian Herzer M.Sc. Shuang Lu M.Sc. Paul Haase

Kurzfassung

Dieser Bericht soll als Dokumentation des Entwicklungsprozesses und der technischen Komponenten des TEGGLA dienen. Hierbei handelt es sich um ein im Rahmen des Praktikum für Produktionstechnik entwickelten Fahrzeugs, welches durch die besonderen Mecanum-Räder mehr Freiheitsgrade hat als ein konventionelles Automobil.

Inhaltsverzeichnis

1 Einleitung & Motivation	1
2 Anforderungen	2
2.1 Lastenheft	2
3 Planung	3
3.1 Entwürfe	3
3.1.1 Seggway	3
3.1.2 Weggchair	3
3.1.3 TEGGLA	3
3.2 Morphologischer Kasten	3
3.3 Online Bestellungen	3
3.4 Verwendete Technologien	3
3.4.1 Git	3
3.4.2 Creo	3
3.4.3 PlatformIO	5
4 Entwicklung der Schlüsselemente	6
4.1 Übersicht des gesamten Modells	6
4.2 Schaltplan	6
4.3 BMS und Laden	6
4.4 Mecanum	6
4.5 Planetengetriebe	6
4.6 ESP-32 vs. ESP-8266	6
4.7 User-Interface	6
4.7.1 Java (obsolet)	6
4.7.2 HTML5 und Controller-Anbindung	6
4.7.3 Protokoll	8
4.8 Steuerung per (XBox) Controller	9
4.9 PLA vs. TPU	9

4.10 Eierhalter	9
4.11 Leichtbau	9
5 Zukünftige Entwicklungsmöglichkeiten	10
5.1 Regler	10
5.2 Verbesserungen	10
5.2.1 Hardware	10
5.2.2 Software	10
6 Zusammenfassung	11
6.1 Evaluation	11
6.2 Errungene Erfahrung	11
A Quellcode	13
A.1 ESP32	13
A.2 Webpage	26
A.3 Java	37
B Technische Zeichnungen	64
Listings	79
Abbildungsverzeichnis	80
Tabellenverzeichnis	81

1 Einleitung & Motivation

2 Anforderungen

2.1 Lastenheft

3 Planung

3.1 Entwürfe

3.1.1 Seggway

3.1.2 Weggchair

3.1.3 TEGGLA

3.2 Morphologischer Kasten

3.3 Online Bestellungen

Um das Fahrzeug nach dem Praktikum behalten zu können, war eine Voraussetzung, dass nur Teile verbaut werden, die nicht Eigentum des Lehrstuhls sind. Aus diesem wurden die nötigen Bauteile bei unterschiedlichen Onlineshops herausgesucht und bestellt. Hierbei fiel die Entscheidung auf Pollin, einem deutschen Elektronik Händler und AliExpress, einem chinesischen Großhändler. In der ursprünglichen Planung wurden die Kosten pro Fahrzeug auf circa 20 € – 25 € überschlagen. In der finalen Bestellung beliefen sich die Kosten auf insgesamt etwa 32 €.

Siehe Tabelle 3.1 für eine genaue Aufteilung der Kosten.

3.4 Verwendete Technologien

3.4.1 Git

3.4.2 Creo

Bei der CAD-Software standen mehrere unterschiedliche Programme unterschiedlicher Hersteller zur Auswahl. Diese waren “Catia” von Dassault Systemes, “Sketchup” von Trimble Inc., “Blender” von Blender Foundation

Artikel	Stk	€/Stk	€	Laden
Netzteil 9V 1A	1	0,95	0,95	Pollin
DC Motor	4	0,95	3,80	Pollin
2S LiPo	1	9,95	9,95	Pollin
XT60 5er Satz	0,5	1,8	0,90	AliExpress
ESP32	1	3,77	3,77	AliExpress
2s BMS	1	0,89	0,89	AliExpress
Kabelset 20cm	0,5	3,30	1,65	AliExpress
Kabelset F - F 10cm	0,5	0,68	0,34	AliExpress
Gyroskop	1	0,93	0,93	AliExpress
H-Brücken	2	1,17	2,34	Bestand
Filament [kg]	0,05	20,00	1,00	Bestand
Schrauben + Muttern [Set]	1	1,00	1,00	Bestand
Motorkabel	1	0,50	0,50	Bestand
Versandkosten AliExpress	0,25	9,00	2,25	
Versandkosten Pollin	0,25	5,00	1,25	
			Total 31,52€	

Tabelle 3.1: Kostenübersicht

3.4.3 PlatformIO

4 Entwicklung der Schlüsselemente

4.1 Übersicht des gesamten Modells

4.2 Schaltplan

4.3 BMS und Laden

4.4 Mecanum

4.5 Planetengetriebe

4.6 ESP-32 vs. ESP-8266

4.7 User-Interface

4.7.1 Java (obsolet)

4.7.2 HTML5 und Controller-Anbindung

Das Java Programm wurde aus mehreren Gründen zugunsten einer auf HTML5, sowie JavaScript basierten Weboberfläche ersetzt:

- Native und einheitliche Unterstützung für Controller unterschiedlichster Marken in HTML5
- Unabhängig von Java Laufzeitumgebung, sowie Verfügbarkeit des Programms. (Hierbei muss nur ein Browser auf dem PC installiert sein.)
- Einfache Übertragung der Daten per WebSockets anstatt von “ra” Sockets, ohne ein eigenes “Frame” um die Daten bauen zu müssen

Dies lässt sich sehr leicht durch die ESPAsyncWebServer Library für den ESP32 lösen. Diese hostet direkt auf dem ESP32 einen WebServer der sowohl HTML5, JS, als auch CSS bereitstellen kann. Als Speicherort für diese Dateien wird das sogenannte Dateisystem SPIFFS verwendet, welches den Flashspeicher des ESP32 als Dateisystem benutzt, wie es beispielsweise aus Windows bekannt ist.

Eine Einschränkung ist die Limitierung auf eine gleichzeitige Verbindung zu dem Server. Dies wurde empirisch herausgefunden und somit konnte nicht sicher gesagt werden, ob es sich hier um eine Einschränkung aufgrund von mangelnder Rechenleistung handelt oder ob die Library nicht mehr unterstützt. Als Lösung dieses Problems, ist nun die Anzahl der Verbindungen die der WiFi Accesspoint akzeptiert, auf eins gesetzt.

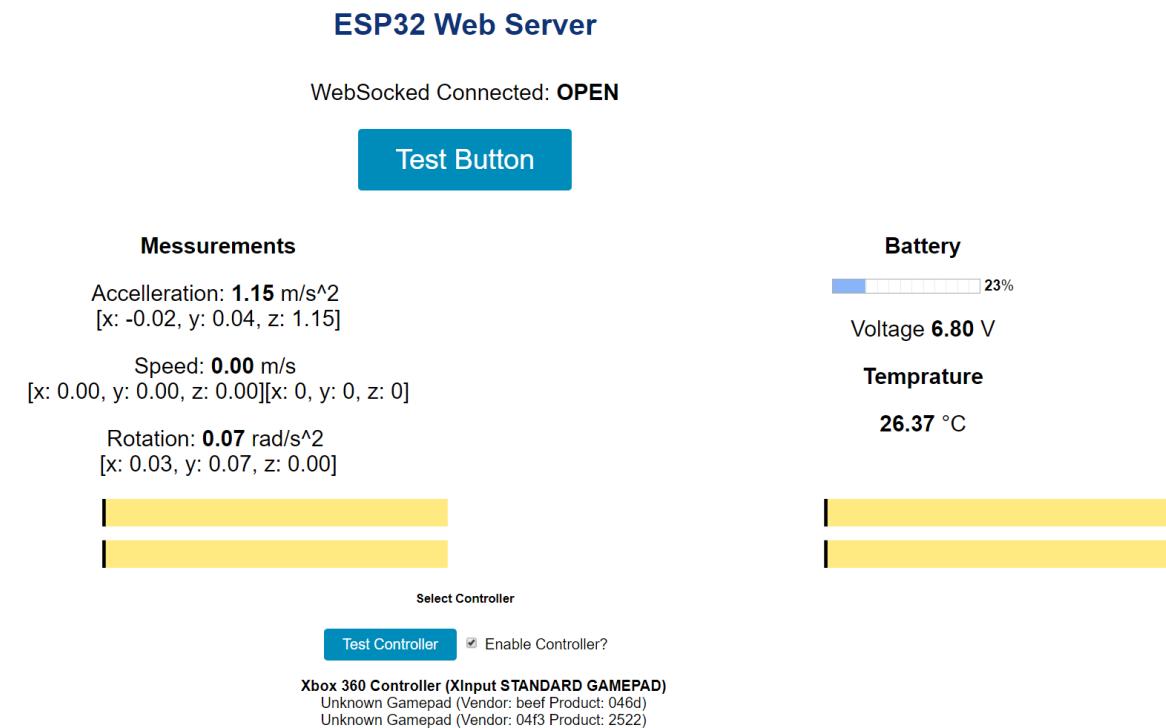


Abbildung 4.1: Bildschirmfoto Weboberfläche

In der Weboberfläche (siehe Abbildung 4.1) sind ebenfalls noch jegliche Messwerte hinterlegt, die das Fahrzeug sammelt.

Diese sind:

- Beschleunigung
- Geschwindigkeit

- Rotationsgeschwindigkeit
- Batteriespannung
- Temperatur
- Drehgeschwindigkeit der jeweiligen Motoren

Für die Wahl des Controllers ist eine auswählbare Liste aller angeschlossenen Controller am Ende der Seite vorhanden. Falls der Nutzer keinen Controller besitzt oder angeschlossen hat, kann durch das Abwählen der Checkbox auf die Steuerung per WASD, sowie die Pfeiltasten gewechselt werden.

4.7.3 Protokoll

Zum Übertragen wurde ein binäres Protokoll ausgewählt, um die Datenrate gering zu halten, sowie die Verarbeitung auf dem Microcontroller zu vereinfachen.

Da durch Websockets bereits ein Integrierter Frame geschickt wird, muss nicht bei jeder Nachricht die Länge und Anfangs- und Endbyte mitgeschickt werden, wie es sonst bei raw Sockets nötig gewesen wäre.

Hierbei wird jeder Wert als Int16 geschickt, um ein ausreichend großes Spektrum bei geringer Datenrate zu gewährleisten.

Am Anfang jeder Nachricht wird die ID ebenfalls als int16 geschickt, somit wären 65536 unterschiedliche Nachrichten erlaubt.

Um die Anzahl an Nachrichten zu verringern, wurden die Batteriespannung und Werte des Gyroskop zu einer kombinierte Nachricht zusammengefasst, um den Overhead gering zu halten.

Name	ID	Werte										
		X-Achse L	Y-Achse L	X-Achse R	Y-Achse R							
DRIVE	0	Speed-X	Speed-Y	Speed-Z	Accel-X	Accel-Y	Accel-Z	Rot-X	Rot-Y	Rot-Z		
GYRO	1											
BATTERY	2	Voltage										
MOTOR	3	Speed-M1	Speed-M2	Speed-M3	Speed-M4							
COMB	4	Speed-X	Speed-Y	Speed-Z	Accel-X	Accel-Y	Accel-Z	Rot-X	Rot-Y	Rot-Z	Battery	Temp

Tabelle 4.1: Binäres Protokoll

4.8 Steuerung per (XBox) Controller

4.9 PLA vs. TPU

4.10 Eierhalter

4.11 Leichtbau

5 Zukünftige Entwicklungsmöglichkeiten

5.1 Regler

5.2 Verbesserungen

5.2.1 Hardware

5.2.2 Software

6 Zusammenfassung

6.1 Evaluation

6.2 Errungene Erfahrung

Anhang

A Quellcode

A.1 ESP32

```
1 #include "communication.h"
2
3 AsyncWebSocketClient *ws_client = nullptr;
4
5 void Communication::onWSData(AsyncWebSocket *server, AsyncWebSocketClient *client,
6     AwsEventType type, uint8_t *data, size_t len) {
7     ws_client = client;
8
9     int16_t *d16 = (int16_t *)data;
10    int16_t id = d16[0];
11
12    switch (id) {
13        case OmniMessageType::DRIVE:
14            onDrive(d16[1], d16[2], d16[3], d16[4]);
15            break;
16
17        default:
18            Serial.printf("No such command %i\n", id);
19            break;
20    }
21
22 void Communication::onDrive(int16_t x1, int16_t y1, int16_t x2, int16_t y2) {
23     // Serial.printf("Driving with speed: %i, %i and %i, %i now\n", x1, y1, x2,
24     // y2);
25     Movement::drive(y1, x1, x2);
26 }
27
28 void Communication::sendCurrGyro(XYZ speed, XYZ accel, XYZ rot) {
29     int16_t buff[10] = {OmniMessageType::CURR_GYRO, speed.x, speed.y, speed.z,
30     accel.x, accel.y, accel.z, rot.x, rot.y, rot.z};
31     ws->binaryAll((uint8_t *)buff, 20);
```

```

30 }
31
32 void Communication::sendCurrGyBatComb(int16_t sX, int16_t sY, int16_t sZ, int16_t
33     aX, int16_t aY, int16_t aZ, int16_t rX, int16_t rY, int16_t rZ, int16_t bat,
34     int16_t temp) {
35     int16_t buff[12] = {OmniMessageType::CURR_GY_BAT_COMB, sX, sY, sZ, aX, aY, aZ,
36     rX, rY, rZ, bat, temp};
37     ws->binaryAll((uint8_t *)buff, 24);
38 }
39
40 void Communication::sendCurrMotor(int16_t vl, int16_t vr, int16_t hl, int16_t hr) {
41     int16_t buff[5] = {OmniMessageType::CURR_MOTOR, vl, vr, hl, hr};
42     // Serial.println("should send now");
43     ws->binaryAll((uint8_t *)buff, 10);
44 }
45
46 void Communication::sendCurrBattery(int16_t cell1, int16_t cell2) {
47     int16_t buff[5] = {OmniMessageType::CURR_BATTERY, cell1, cell2};
48     // Serial.println("should send now");
49     ws->binaryAll((uint8_t *)buff, 10);
50 }
51
52 AsyncWebSocket *Communication::ws;

```

Listing A.1: communication.cpp

```

1 #pragma once
2
3 #include "movement.h"
4 #include <ESPAsyncWebServer.h>
5
6 /**
7  * Limitations due to js:
8  * Buffer has to be all the same type
9  *
10 * Type is int16 as well at the start of the array
11 *
12 */
13 enum OmniMessageType {
14     DRIVE = 0,           //L4 {x1} int16 [-1023, 1023] :: {y1} int16 [-1023,
15     1023] :: {x2} int16 [-1023, 1023] :: {y2} int16 [-1023, 1023]

```

```

15     CURR_GYRO = 1,           //L18 {speed m/s} (x) int16 (y) int16 (z) int16 :: 
16     {accel m/s^2} (x) int16 (y) int16 (z) int16 :: {rot} (x) int16 (y) int16 (z)
17     int16
18     CURR_BATTERY = 2,       //L1 {cell1} int16 [0, 2^12]
19     CURR_MOTOR = 3,         //L4 {vl} int16 [-1023, 1023] :: {vr} int16 [-1023,
20     1023] :: {hl} int16 [-1023, 1023] :: {hr} int16 [-1023, 1023]
21     CURR_GY_BAT_COMB = 4,   //L20 {speed m/s} (x) int16 (y) int16 (z) int16 :: 
22     {accel m/s^2} (x) int16 (y) int16 (z) int16 :: {rot} (x) int16 (y) int16 (z)
23     int16 :: {bat} int16 [0, 2^12] :: {temp} int16
24 };
25
26
27 struct XYZ {
28     int16_t x;
29     int16_t y;
30     int16_t z;
31 };
32
33
34 class Communication {
35 public:
36     static void onWSData(AsyncWebSocket *server, AsyncWebSocketClient *client,
37     AwsEventType type, uint8_t *data, size_t len);
38     static void onDrive(int16_t x1, int16_t y1, int16_t x2, int16_t y2);
39     static void sendCurrGyro(XYZ speed, XYZ accel, XYZ rot);
40     static void sendCurrBattery(int16_t cell1, int16_t cell2);
41     static void sendCurrGyBatComb(int16_t sX, int16_t sY, int16_t sZ, int16_t aX,
42     int16_t aY, int16_t aZ, int16_t rX, int16_t rY, int16_t rZ, int16_t bat,
43     int16_t temp);
44     static void sendCurrMotor(int16_t vl, int16_t vr, int16_t hl, int16_t hr);
45
46     static AsyncWebSocket *ws;
47 };

```

Listing A.2: communication.h

```

1 #include "movement.h"
2
3 // ----- Motor Class
4 // ##### Functions #####
5 void Motor::setSpeed(int16_t speed) {
6     // disable before before changing direction
7     // (probably not needed) safety to prevent shoutthrough

```

```
8     ledcWrite(channel, 0);
9     if (speed > 0) {
10         digitalWrite(pin_dir1, LOW);
11         digitalWrite(pin_dir2, HIGH);
12     } else if (speed < 0) {
13         digitalWrite(pin_dir1, HIGH);
14         digitalWrite(pin_dir2, LOW);
15     } else {
16         digitalWrite(pin_dir1, LOW);
17         digitalWrite(pin_dir2, LOW);
18         return;
19     }
20
21     ledcWrite(channel, abs(speed));
22 }
23
24 void Motor::stop() {
25     ledcWrite(channel, 0);
26     digitalWrite(pin_dir1, LOW);
27     digitalWrite(pin_dir2, LOW);
28 }
29
30 // ----- Movement Class
31 // ##### Attributes #####
32 Motor Movement::MOTOR_VL = Motor(CH_MOTOR_VL, PIN_MOTOR_VL_DIR1,
33     PIN_MOTOR_VL_DIR2);
33 Motor Movement::MOTOR_VR = Motor(CH_MOTOR_VR, PIN_MOTOR_VR_DIR1,
34     PIN_MOTOR_VR_DIR2);
34 Motor Movement::MOTOR_HL = Motor(CH_MOTOR_HL, PIN_MOTOR_HL_DIR1,
35     PIN_MOTOR_HL_DIR2);
35 Motor Movement::MOTOR_HR = Motor(CH_MOTOR_HR, PIN_MOTOR_HR_DIR1,
36     PIN_MOTOR_HR_DIR2);
36
37 // ##### Functions #####
38 void Movement::initPWM() {
39     Serial.println("Initing PWMs");
40
41     // Setting all pins to output
42     pinMode(PIN_LED, OUTPUT);
43     pinMode(PIN_MOTOR_VL_EN, OUTPUT);
44     pinMode(PIN_MOTOR_VL_DIR1, OUTPUT);
45     pinMode(PIN_MOTOR_VL_DIR2, OUTPUT);
```

```
46
47     pinMode(PIN_MOTOR_VR_EN, OUTPUT);
48     pinMode(PIN_MOTOR_VR_DIR1, OUTPUT);
49     pinMode(PIN_MOTOR_VR_DIR2, OUTPUT);
50
51     pinMode(PIN_MOTOR_HL_EN, OUTPUT);
52     pinMode(PIN_MOTOR_HL_DIR1, OUTPUT);
53     pinMode(PIN_MOTOR_HL_DIR2, OUTPUT);
54
55     pinMode(PIN_MOTOR_HR_EN, OUTPUT);
56     pinMode(PIN_MOTOR_HR_DIR1, OUTPUT);
57     pinMode(PIN_MOTOR_HR_DIR2, OUTPUT);
58
59 // Disable all pins by default to not possibly cause shootthrough
60 digitalWrite(PIN_MOTOR_VL_EN, LOW);
61 digitalWrite(PIN_MOTOR_VL_DIR1, LOW);
62 digitalWrite(PIN_MOTOR_VL_DIR2, LOW);
63
64 digitalWrite(PIN_MOTOR_VR_EN, LOW);
65 digitalWrite(PIN_MOTOR_VR_DIR1, LOW);
66 digitalWrite(PIN_MOTOR_VR_DIR2, LOW);
67
68 digitalWrite(PIN_MOTOR_HL_EN, LOW);
69 digitalWrite(PIN_MOTOR_HL_DIR1, LOW);
70 digitalWrite(PIN_MOTOR_HL_DIR2, LOW);
71
72 digitalWrite(PIN_MOTOR_HR_EN, LOW);
73 digitalWrite(PIN_MOTOR_HR_DIR1, LOW);
74 digitalWrite(PIN_MOTOR_HR_DIR2, LOW);
75
76 // configure LED PWM functionalitites
77 ledcSetup(CH_LED, PWM_FREQ, PWM_RESOLUTION);
78 ledcSetup(CH_MOTOR_VL, PWM_FREQ, PWM_RESOLUTION);
79 ledcSetup(CH_MOTOR_VR, PWM_FREQ, PWM_RESOLUTION);
80 ledcSetup(CH_MOTOR_HL, PWM_FREQ, PWM_RESOLUTION);
81 ledcSetup(CH_MOTOR_HR, PWM_FREQ, PWM_RESOLUTION);
82
83 // attach the channel to the GPIO2 to be controlled
84 ledcAttachPin(PIN_LED, CH_LED);
85 ledcAttachPin(PIN_MOTOR_VL_EN, CH_MOTOR_VL);
86 ledcAttachPin(PIN_MOTOR_VR_EN, CH_MOTOR_VR);
87 ledcAttachPin(PIN_MOTOR_HL_EN, CH_MOTOR_HL);
88 ledcAttachPin(PIN_MOTOR_HR_EN, CH_MOTOR_HR);
```

```
89     Serial.println("Initiated PWMs");
90 }
91
92
93 void Movement::drive(int controlFront, int controlSide, int controlTurn) {
94     if (abs(controlFront) < CONTROLLER_LOWER_LIMIT && abs(controlSide) <
95         CONTROLLER_LOWER_LIMIT && abs(controlTurn) < CONTROLLER_LOWER_LIMIT) {
96         // Serial.println("Stopping Motors");
97         MOTOR_VL.stop();
98         MOTOR_VR.stop();
99         MOTOR_HL.stop();
100        MOTOR_HR.stop();
101
102        Communication::sendCurrMotor(0, 0, 0, 0);
103        return;
104    }
105
106    int speedVL = 0;
107    int speedVR = 0;
108    int speedHL = 0;
109    int speedHR = 0;
110
111    if (abs(controlFront) > 0 && controlSide == 0 && controlTurn == 0) {
112        speedVL = controlFront;
113        speedVR = controlFront;
114        speedHL = controlFront;
115        speedHR = controlFront;
116    } else if (abs(controlSide) > 0 && controlFront == 0 && controlTurn == 0) {
117        speedVL = controlSide;
118        speedVR = -controlSide;
119        speedHL = -controlSide;
120        speedHR = controlSide;
121    } else if (controlSide == 0 && controlFront == 0 && abs(controlTurn) > 0) {
122        speedVL = controlTurn;
123        speedVR = -controlTurn;
124        speedHL = controlTurn;
125        speedHR = -controlTurn;
126    } else {
127        driveAlgorithm(controlFront, controlSide, controlTurn, &speedVL, &speedVR,
128                      &speedHL, &speedHR);
129    }
130}
```

```

129     speedVL = speedVL / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
130     USEABLE_UPPER_LIMIT) * sgn(speedVL);
131     speedVR = speedVR / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
132     USEABLE_UPPER_LIMIT) * sgn(speedVR);
133     speedHL = speedHL / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
134     USEABLE_UPPER_LIMIT) * sgn(speedHL);
135     speedHR = speedHR / 1023.0 * USEABLE_UPPER_LIMIT + (1023 -
136     USEABLE_UPPER_LIMIT) * sgn(speedHR);
137
138
139     MOTOR_VL.setSpeed(speedVL);
140     MOTOR_VR.setSpeed(speedVR);
141     MOTOR_HL.setSpeed(speedHL);
142     MOTOR_HR.setSpeed(speedHR);
143
144
145     Communication::sendCurrMotor(speedVL, speedVR, speedHL, speedHR);
146 }
147
148 void Movement::driveAlgorithm(int controlFront, int controlSide, int controlTurn,
149     int *speedVL, int *speedVR, int *speedHL, int *speedHR) {
150     double phi = atan2(controlSide, controlFront);
151
152     int vd = min((int)sqrt(controlFront * controlFront + controlSide *
153     controlSide), 1023);
154     vd -= controlTurn / 2;
155     int vphi = controlTurn / 2;
156
157     double s = vd * sin(phi + PI / 4);
158     double c = vd * cos(phi + PI / 4);
159
160     *speedVL = s + vphi;
161     *speedVR = c - vphi;
162     *speedHL = c + vphi;
163     *speedHR = s - vphi;
164 }
```

Listing A.3: movement.cpp

```

1 #pragma once
2
3 #include "communication.h"
4 #include "util.h"
5 #include <Arduino.h>
```

```
6 #include <math.h>
7
8 class Motor {
9 private:
10     char channel;
11     char pin_dir1;
12     char pin_dir2;
13
14 public:
15     Motor(char channel, char pin_dir1, char pin_dir2) : channel(channel),
16         pin_dir1(pin_dir1), pin_dir2(pin_dir2){};
17     ~Motor() {}
18
19     /**
20      * speed in [-1023; 1023]
21      */
22     void setSpeed(int16_t speed);
23
24     void stop();
25 };
26
27 class Movement {
28 private:
29     /* data */
30
31     static Motor MOTOR_VL;
32     static Motor MOTOR_VR;
33     static Motor MOTOR_HL;
34     static Motor MOTOR_HR;
35
36
37     /** Amount of values down from 1023 which can be used */
38     static const int USEABLE_UPPER_LIMIT = 700;
39     /** Wont move below this limit */
40     static const int CONTROLLER_LOWER_LIMIT = 50;
41
42     static const char PIN_MOTOR_VL_EN = 23;
43     static const char PIN_MOTOR_VL_DIR1 = 18;
44     static const char PIN_MOTOR_VL_DIR2 = 19;
45
46     static const char PIN_MOTOR_VR_EN = 32;
47     static const char PIN_MOTOR_VR_DIR1 = 12;
```

```

48     static const char PIN_MOTOR_VR_DIR2 = 13;
49
50     static const char PIN_MOTOR_HL_EN = 16;
51     static const char PIN_MOTOR_HL_DIR1 = 17;
52     static const char PIN_MOTOR_HL_DIR2 = 5;
53
54     static const char PIN_MOTOR_HR_EN = 4;
55     static const char PIN_MOTOR_HR_DIR1 = 3;
56     static const char PIN_MOTOR_HR_DIR2 = 15;
57
58     static const int PWM_FREQ = 500;
59     static const char CH_LED = 0;
60     static const char CH_MOTOR_VL = 1;
61     static const char CH_MOTOR_VR = 2;
62     static const char CH_MOTOR_HL = 3;
63     static const char CH_MOTOR_HR = 4;
64     static const char PWM_RESOLUTION = 10; //Resolution 8, 10, 12, 15
65
66     static void initPWM();
67
68     static void driveMotor(char channel, char dir1, char dir2, int speed);
69
70 /**
71  * Converts the given controls into wheel speeds
72  * Algorithm source:
73  * http://eprints.utm.edu.my/16543/1/Omni%20Directional%20Control%20Algorithm%20For%20Mecanu
74  */
75     static void drive(int controlFront, int controlSide, int controlTurn);
76
77 private:
78     static void driveAlgorithm(int controlFront, int controlSide, int controlTurn,
79     int *speedVL, int *speedVR, int *speedHL, int *speedHR);
78 };

```

Listing A.4: movement.h

```

1 #include "util.h"
2
3 ****
4 * high precision sine/cosine
5 *
6 * Source: https://gist.github.com/geraldyeo/988116

```

```
7  *
8  *
9  ****
10 void cossin(float x, float *outCos, float *outSin) {
11     float sin, cos;
12
13     //always wrap input angle to -PI..PI
14     if (x < -3.14159265)
15         x += 6.28318531;
16     else if (x > 3.14159265)
17         x -= 6.28318531;
18
19     //compute sine
20     if (x < 0) {
21         sin = 1.27323954 * x + .405284735 * x * x;
22
23         if (sin < 0)
24             sin = .225 * (sin * -sin - sin) + sin;
25         else
26             sin = .225 * (sin * sin - sin) + sin;
27     } else {
28         sin = 1.27323954 * x - 0.405284735 * x * x;
29
30         if (sin < 0)
31             sin = .225 * (sin * -sin - sin) + sin;
32         else
33             sin = .225 * (sin * sin - sin) + sin;
34     }
35
36     //compute cosine: sin(x + PI/2) = cos(x)
37     x += 1.57079632;
38     if (x > 3.14159265)
39         x -= 6.28318531;
40
41     if (x < 0) {
42         cos = 1.27323954 * x + 0.405284735 * x * x;
43
44         if (cos < 0)
45             cos = .225 * (cos * -cos - cos) + cos;
46         else
47             cos = .225 * (cos * cos - cos) + cos;
48     } else {
49         cos = 1.27323954 * x - 0.405284735 * x * x;
```

```

50
51     if (cos < 0)
52         cos = .225 * (cos * -cos - cos) + cos;
53     else
54         cos = .225 * (cos * cos - cos) + cos;
55 }
56
57 *outSin = sin;
58 *outCos = cos;
59 }
60
61 /**
62 * Branchless signum function
63 */
64
65 int sgn(int val) {
66     return (0 < val) - (val < 0);
67 }

```

Listing A.5: util.cpp

```

1 #pragma once
2
3 void coassin(float x, float *outCos, float *outSin);
4
5 int sgn(int val);

```

Listing A.6: util.h

```

1 #include "MPU6050.h"
2 #include "communication.h"
3 #include "movement.h"
4 #include <Arduino.h>
5 #include <ESPAsyncWebServer.h>
6 #include <SPIFFS.h>
7 #include <WiFi.h>
8 #include <Wire.h>
9
10 // Replace with your network credentials
11 const char *ssid = "ESP32-OmniMove";

```

```
12 const char *password = "123456789";
13 // const char* ssid      = "moto g(6) 2970";
14 // const char* password = "428d1382abf1";
15
16 // const byte DNS_PORT = 53;
17 const IPAddress apIP = IPAddress(192, 168, 4, 1);
18
19 AsyncWebServer server(80);
20 AsyncWebSocket ws("/ws");
21 MPU6050 mpu;
22
23 void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *client, AwsEventType
24   type, void *arg, uint8_t *data, size_t len) {
25
26   switch (type) {
27     case WS_EVT_CONNECT:
28       Serial.printf("Client connected from %s\n",
29         client->remoteIP().toString().c_str());
30       break;
31
32     case WS_EVT_DATA:
33       Communication::onWSData(server, client, type, data, len);
34       break;
35
36     default:
37       break;
38   }
39 }
40
41 void setup() {
42   // enableCore1WDT();
43   Serial.begin(115200);
44
45   Wire.begin();
46
47   mpu.initialize();
48
49   // mpu.CalibrateAccel_MP6500(6);
50   // mpu.CalibrateGyro(6);
51
52   // mpu.PrintActiveOffsets_MP6500();
53   mpu.setXGyroOffset(96);
54   mpu.setYGyroOffset(92);
```

```
53     mpu.setZGyroOffset(-20);
54
55     // Serial.printf("\n");
56     Movement::initPWM();
57
58     // enable AP with dns
59     WiFi.mode(WIFI_AP);
60
61     // Setup websockets
62     ws.onEvent(onEvent);
63     server.addHandler(&ws);
64     Communication::ws = &ws;
65     ws.enable(true);
66
67     // Initialize SPIFFS
68     if (!SPIFFS.begin(true)) {
69         while (true) {
70             Serial.println("An Error has occurred while mounting SPIFFS");
71             delay(1000);
72         }
73         return;
74     } else {
75         Serial.println("Mounted SPIFFS successfully");
76     }
77
78     // Route for root / web page
79     server.on("/", HTTP_GET, [] (AsyncWebRequest *request) {
80         // Serial.println("request on index");
81         request->send(SPIFFS, "/index.html", String(), false, nullptr);
82     });
83
84     // Route to load style.css file
85     server.on("/style.css", HTTP_GET, [] (AsyncWebRequest *request) {
86         // Serial.println("request on style");
87         request->send(SPIFFS, "/style.css", "text/css");
88     });
89
90     // Route to load code.js file
91     server.on("/code.js", HTTP_GET, [] (AsyncWebRequest *request) {
92         // Serial.println("request on code");
93         request->send(SPIFFS, "/code.js", "text/javascript");
94     });
95
```

```

96  // Route to load code.js file
97  server.on("/progressbar.min.js", HTTP_GET, [] (AsyncWebServerRequest *request) {
98      // Serial.println("request on progressbar");
99      request->send(SPIFFS, "/progressbar.min.js", "text/javascript");
100 });
101
102 // WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
103 WiFi.setSleep(false);
104 WiFi.softAP(ssid, password, 6, 1);
105 delay(1000);
106
107 server.begin();
108 }
109
110 void loop() {
111
112 // put your main code here, to run repeatedly:
113 // dnsServer.processNextRequest();
114 uint16_t v = analogRead(39);
115 int16_t x, y, z, gx, gy, gz, temp;
116
117 temp = mpu.getTemperature();
118 mpu.getMotion6(&x, &y, &z, &gx, &gy, &gz);
119
120 // Serial.printf("x: %6.2fg, y: %6.2fg, z: %6.2fg, gx: %6.2f°/s, gy: %6.2f°/s,
121 // gz: %6.2f°/s, temp: %6.2f°C\r", x / 16384.0, y / 16384.0, z / 16384.0, gx /
122 // 250.0, gy / 250.0, gz / 250.0, temp / 340.0 + 36.53);
123 // power = (v / 4095 * 3.1 + .1) * 3;
124 Communication::sendCurrGyBatComb(0, 0, 0, x, y, z, gx, gy, gz, v, temp);
125 delay(1000);
126 }

```

Listing A.7: main.cpp

A.2 Webpage

```

1 <!DOCTYPE html>
2 <html>
3
4 <head>

```

```
5  <title>ESP32 Web Server</title>
6  <meta charset="utf-8" />
7  <meta name="viewport" content="width=device-width, initial-scale=1">
8  <link rel="icon" href="data:, ">
9  <link rel="stylesheet" type="text/css" href="style.css">
10 </head>
11
12 <body>
13  <h1>ESP32 Web Server</h1>
14  <p>WebSocked Connected: <strong id="wsState">CONNECTING</strong></p>
15  <p><button class="button" onclick="sendSpeed()">Test Button</button></p>
16
17
18  <div class="wrapper">
19    <div class="left">
20      <h2>Measurements</h2>
21      <p>Accelleration: <strong id="valAccelTotal">0</strong> m/s2
22        <br /> <span id="valAccel">[x: 0, y: 0, z: 0]</span>
23      </p>
24
25      <p>Speed: <strong id="valSpeedTotal">0</strong> m/s
26        <br /> <span id="valSpeed">[x: 0, y: 0, z: 0]</span></p>
27
28      <p>Rotation: <strong id="valRotTotal">0</strong> rad/s2
29        <br /> <span id="valRot">[x: 0, y: 0, z: 0]</span></p>
30
31    </div>
32
33    <div class="right">
34      <h2>Battery</h2>
35      <progress id="progBattery" value="60" max="100"></progress>
36
37      <strong id="valBatteryTotal">-1</strong>%
38      <p>Voltage <strong id="valVolt">-1</strong> V</p>
39
40      <h2>Temprature</h2>
41      <p><strong id="valTemp">-1</strong> °C</p>
42    </div>
43  </div>
44
45
46  <div>
```

```
47  <svg viewBox="0 0 205 10" preserveAspectRatio="none" style="width: 100%;  
48  height: 100%;>  
49  <path id="pathVL" d="M50,2 L100,2 0,2 50,2" stroke="#FFEA82"  
50  stroke-width="4" fill-opacity="0"  
51  style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>  
52  <path id="pathHL" d="M50,8 L100,8 0,8 50,8" stroke="#FFEA82"  
53  stroke-width="4" fill-opacity="0"  
54  style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>  
55  <path id="pathVR" d="M155,2 L205,2 105,2 155,2" stroke="#FFEA82"  
56  stroke-width="4" fill-opacity="0"  
57  style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>  
58  <path d="M 50 0 L 50 4" stroke="black" stroke-width="0.5" fill="none" />  
59  <path d="M 50 6 L 50 10" stroke="black" stroke-width="0.5" fill="none" />  
60  <path d="M 155 0 L 155 4" stroke="black" stroke-width="0.5" fill="none" />  
61  <path d="M 155 6 L 155 10" stroke="black" stroke-width="0.5" fill="none" />  
62  </svg>  
63 </div>  
64  
65 <div>  
66  <h5>Select Controller</h5>  
67  <button class="buttonsmall" onclick="updateControllerList()">Test  
68  Controller</button>  
69  <input type="checkbox" id="controllerEnabled" checked>  
70  <label for="controllerEnabled">Enable Controller?</label>  
71  <ul id="controllerList">  
72  <li onclick="onControllerSelected(0)">Test</li>  
73  </ul>  
74 </div>  
75  
76 </body>  
77  
78  
79 <script type="text/javascript" src="progressbar.min.js"></script>  
80 <script type="text/javascript" src="code.js"></script>  
81  
82 </html>
```

Listing A.8: index.html

```
1 let ws = new WebSocket("ws://192.168.4.1/ws");
2 // let ws = new WebSocket("ws://demos.kaazing.com/echo");
3 ws.binaryType = 'arraybuffer';
4
5 let barVL = new ProgressBar.Path('#pathVL', { easing: 'easeInOut', duration: 140,
6   });
7 let barVR = new ProgressBar.Path('#pathVR', { easing: 'easeInOut', duration: 140,
8   });
9 let barHL = new ProgressBar.Path('#pathHL', { easing: 'easeInOut', duration: 140,
10  });
11 let barHR = new ProgressBar.Path('#pathHR', { easing: 'easeInOut', duration: 140,
12  });
13
14 barVL.set(0.1)
15 barVR.set(0.1)
16 barHL.set(0.1)
17 barHR.set(0.1)
18
19
20 //##region >>> WebSocket
21 ws.onopen = (event) => {
22   document.getElementById("wsState").innerHTML = "OPEN";
23 }
24
25 ws.onerror = (event) => {
26   document.getElementById("wsState").innerHTML = "ERROR";
27 }
28
29 ws.onclose = (event) => {
30   document.getElementById("wsState").innerHTML = "CLOSED";
31 }
32
33 ws.onmessage = function (event) {
34   // console.log("WebSocket message received:", event);
```

```

35 let dv = new DataView(event.data);
36 let id = dv.getInt16(0, true);
37
38 switch (id) {
39   case 1: //gyro
40     let a = [dv.getInt16(2, true), dv.getInt16(4, true), dv.getInt16(6, true)];
41     let s = [dv.getInt16(8, true), dv.getInt16(10, true), dv.getInt16(12, true)];
42     let r = [dv.getInt16(2, true), dv.getInt16(4, true), dv.getInt16(6, true)];
43
44     document.getElementById("valAccelTotal").innerHTML = Math.sqrt(a[0] * a[0] +
45     a[1] * a[1] + a[2] * a[2]).toFixed(2);
46     document.getElementById("valSpeedTotal").innerHTML = Math.sqrt(a[0] * s[0] +
47     s[1] * s[1] + s[2] * s[2]).toFixed(2);
48     document.getElementById("valRotTotal").innerHTML = Math.sqrt(r[0] * r[0] +
49     r[1] * r[1] + r[2] * r[2]).toFixed(2);
50
51     document.getElementById("valAccel").innerHTML = "[x: " + a[0] + ", y: " +
52     a[1] + ", z: " + a[2] + "]";
53     document.getElementById("valSpeed").innerHTML = "[x: " + s[0] + ", y: " +
54     s[1] + ", z: " + s[2] + "]";
55     document.getElementById("valRot").innerHTML = "[x: " + r[0] + ", y: " + r[1] +
56     ", z: " + r[2] + "]";
57     break;
58
59   case 2: //battery
60     let c1 = (dv.getInt16(2, true) / 4095 * 3.1 + .1) * 3;
61
62     document.getElementById("valVolt").innerHTML = c1.toFixed(2);
63     break;
64
65   case 3: //motors
66     const UPPER_LIMIT = 300;
67
68     let vl = dv.getInt16(2, true) / (1023 * 4);
69     let vr = dv.getInt16(4, true) / (1023 * 4);
70     let hl = dv.getInt16(6, true) / (1023 * 4);
71     let hr = dv.getInt16(8, true) / (1023 * 4);
72
73     barVL.animate(vl);
74     barVR.animate(vr);
75     barHL.animate(hl);
76     barHR.animate(hr);
77     break;
78
79 }

```

```

72     case 4: //gyroBatComb
73         let s1 = [dv.getInt16(2, true), dv.getInt16(4, true), dv.getInt16(6, true)];
74         let a1 = [dv.getInt16(8, true) / 16384.0, dv.getInt16(10, true) / 16384.0,
75         dv.getInt16(12, true) / 16384.0];
76         let r1 = [dv.getInt16(14, true) / 250.0, dv.getInt16(16, true) / 250.0,
77         dv.getInt16(18, true) / 250.0];
78
79
80         document.getElementById("valAccelTotal").innerHTML = Math.sqrt(a1[0] * a1[0]
81         + a1[1] * a1[1] + a1[2] * a1[2]).toFixed(2);
82         document.getElementById("valSpeedTotal").innerHTML = Math.sqrt(s1[0] * s1[0]
83         + s1[1] * s1[1] + s1[2] * s1[2]).toFixed(2);
84         document.getElementById("valRotTotal").innerHTML = Math.sqrt(r1[0] * r1[0] +
85         r1[1] * r1[1] + r1[2] * r1[2]).toFixed(2);
86
87         document.getElementById("valAccel").innerHTML = "[x: " + a1[0].toFixed(2) +
88         ", y: " + a1[1].toFixed(2) + ", z: " + a1[2].toFixed(2) + "J]";
89         document.getElementById("valSpeed").innerHTML = "[x: " + s1[0].toFixed(2) +
90         ", y: " + s1[1].toFixed(2) + ", z: " + s1[2].toFixed(2) + "J]";
91         document.getElementById("valRot").innerHTML = "[x: " + r1[0].toFixed(2) + ",
92         y: " + r1[1].toFixed(2) + ", z: " + r1[2].toFixed(2) + "J]";
93
94         let bat = (dv.getInt16(20, true) / 4095 * 3.1 + .1) * 3;
95         let batPercent = (bat - 6.5) / (7.8 - 6.5) * 100;
96         let temp = dv.getInt16(22, true) / 340.0 + 36.53;
97
98         document.getElementById("valVolt").innerHTML = bat.toFixed(2);
99         document.getElementById("progBattery").value = batPercent.toFixed(2);
100        document.getElementById("valBatteryTotal").innerHTML = batPercent.toFixed(0);
101
102        document.getElementById("valTemp").innerHTML = temp.toFixed(2);
103        break;
104
105    default:
106        break;
107    }
108};
109
110//#endregion
111
112//#region >>> Controller
113
114window.addEventListener("gamepadconnected", (event) => {
115    console.log("A gamepad connected:");
116

```

```
107  console.log(event.gamepad);
108
109  updateControllerList();
110 });
111
112 window.addEventListener("gamepaddisconnected", (event) => {
113  console.log("A gamepad disconnected:");
114  console.log(event.gamepad);
115
116  updateControllerList();
117 });
118
119
120
121
122 // Lets the user select which connected controller to use
123 function updateControllerList() {
124  let inner = ""
125
126  let pads = navigator.getGamepads();
127  console.log(pads);
128
129  for (let i = 0; i < pads.length; i++) {
130    const element = pads[i];
131    if (element === null)
132      continue;
133    if (selectedControllerIndex == i) {
134      inner += "<li onclick=\"onControllerSelected(" + i + ")\"><strong>" +
135      element.id + "</strong></li>";
136    } else {
137      inner += "<li onclick=\"onControllerSelected(" + i + ")\">" + element.id +
138      "</li>";
139    }
140
141  document.getElementById("controllerList").innerHTML = inner;
142
143 // gets called when a Controller gets selected in the List
144 function onControllerSelected(index) {
145  selectedControllerIndex = index;
146  updateControllerList();
147 }
```

```
148
149
150 let lastX1 = 0, lastX2 = 0, lastY1 = 0, lastY2 = 0;
151 function controllerFunc() {
152     if (!document.getElementById("controllerEnabled").checked)
153         return;
154
155     let pads = navigator.getGamepads();
156     if (pads === null || pads === undefined || pads[selectedControllerIndex] === null
157         || pads[selectedControllerIndex] === undefined)
158         return;
159     x1 = Math.floor(pads[selectedControllerIndex].axes[0] * 1023);
160     y1 = -Math.floor(pads[selectedControllerIndex].axes[1] * 1023);
161     x2 = Math.floor(pads[selectedControllerIndex].axes[2] * 1023);
162     y2 = -Math.floor(pads[selectedControllerIndex].axes[3] * 1023);
163
164     if (Math.abs(x1) < 200)
165         x1 -= x1;
166     if (Math.abs(y1) < 200)
167         y1 -= y1;
168     if (Math.abs(x2) < 200)
169         x2 -= x2;
170     if (Math.abs(y2) < 200)
171         y2 -= y2;
172
173     if (lastX1 != x1 || lastY1 != y1 || lastX2 != x2 || lastY2 != y2) {
174         console.log(x1 + ", " + y1 + " : " + x2 + ", " + y2);
175         if (ws.readyState == WebSocket.OPEN) {
176             let buf = new Int16Array([0, x1, y1, x2, y2])
177             ws.send(buf);
178         }
179         lastX1 = x1;
180         lastX2 = x2;
181         lastY1 = y1;
182         lastY2 = y2;
183     }
184
185 }
186 //endregion
187
188 //#region >>> Keyboard Control
189 let dirX = 0;
```

```
190 let dirY = 0;
191 let rotX = 0;
192 //a: 65, s: 83, d:68, w:87
193 document.addEventListener('keydown', function (event) {
194     let c = event.keyCode;
195     switch (event.keyCode) {
196         case 65:
197             dirX = -1;
198             break;
199         case 68:
200             dirX = 1;
201             break;
202         case 87:
203             dirY = 1;
204             break;
205         case 83:
206             dirY = -1;
207             break;
208         case 39:
209             rotX = 1;
210             break;
211         case 37:
212             rotX = -1;
213             break;
214         case 27:
215             dirY = 0;
216             dirX = 0;
217             break;
218     default:
219         break;
220     }
221
222     if (lastX1 != dirX * 1023 || lastY1 != dirY * 1023 || lastX2 != rotX * 1023 ||
223         lastY2 != 0) {
224         console.log(dirX * 1023 + ", " + dirY * 1023 + " : " + rotX * 1023 + ", " + 0);
225         if (ws.readyState == WebSocket.OPEN) {
226             let buf = new Int16Array([0, dirX * 1023, dirY * 1023, rotX * 1023, 0])
227             ws.send(buf);
228         }
229         lastX1 = dirX * 1023;
230         lastY1 = dirY * 1023;
231         lastX2 = rotX * 1023;
232         lastY2 = 0;
233     }
234 }
```

```
232     }
233
234     // console.log("x: " + dirX + " y: " + dirY);
235 });
236
237 document.addEventListener('keyup', function (event) {
238     let c = event.keyCode;
239     switch (event.keyCode) {
240         case 65:
241         case 68:
242             dirX = 0;
243             break;
244         case 87:
245         case 83:
246             dirY = 0;
247             break;
248         case 39:
249         case 37:
250             rotX = 0;
251             break;
252         default:
253             break;
254     }
255     console.log("x: " + dirX + " y: " + dirY);
256
257     if (lastX1 != dirX * 1023 || lastY1 != dirY * 1023 || lastX2 != rotX * 1023 ||
258         lastY2 != 0) {
259         console.log(dirX * 1023 + ", " + dirY * 1023 + " : " + rotX * 1023 + ", " + 0);
260         if (ws.readyState == WebSocket.OPEN) {
261             let buf = new Int16Array([0, dirX * 1023, dirY * 1023, rotX * 1023, 0])
262             ws.send(buf);
263         }
264         lastX1 = dirX * 1023;
265         lastY1 = dirY * 1023;
266         lastX2 = rotX * 1023;
267         lastY2 = 0;
268     }
269 });
270
271
272
273 function sendSpeed() {
```

```
274 console.log("speed");
275 let arr = new Int16Array([2, Math.random() * 4048, Math.random() * 4048]);
276 let arr2 = new Int16Array([1,
277   Math.random() * 1023, Math.random() * 1023, Math.random() * 1023,
278   Math.random() * 1023, Math.random() * 1023, Math.random() * 1023,
279   Math.random() * 1023, Math.random() * 1023, Math.random() * 1023]);
280 ws.send(arr);
281 ws.send(arr2);
282 }
```

Listing A.9: code.js

```
1 html {
2   font-family: Helvetica;
3   display: inline-block;
4   margin: 0px auto;
5   text-align: center;
6 }
7 h1{
8   color: #0F3376;
9   padding: 2vh;
10 }
11 p{
12   font-size: 1.5rem;
13 }
14 .button {
15   display: inline-block;
16   background-color: #008CBA;
17   border: none;
18   border-radius: 4px;
19   color: white;
20   padding: 16px 40px;
21   text-decoration: none;
22   font-size: 30px;
23   margin: 2px;
24   cursor: pointer;
25 }
26
27 .buttonsmall {
28   display: inline-block;
29   background-color: #008CBA;
30   border: none;
```

```
31 border-radius: 4px;  
32 color: white;  
33 padding: 8px 20px;  
34 text-decoration: none;  
35 font-size: 16px;  
36 margin: 2px;  
37 cursor: pointer;  
38 }  
39  
40 .button2 {  
41 background-color: #f44336;  
42 }  
43  
44 .wrapper {  
45 display: flex;  
46 }  
47  
48 .left {  
49 flex: 0 0 65%;  
50 }  
51  
52 .right {  
53 flex: 1;  
54 }
```

Listing A.10: style.css

A.3 Java

```
1 package main;  
2 import Toaster.Toaster;  
3 import Utils.*;  
4  
5 import java.awt.*;  
6 import java.awt.event.*;  
7 import javax.swing.*;  
8  
9 public class ConnectUI extends JFrame {  
10  
11     /**
```

```
12     *
13     */
14     private static final long serialVersionUID = 1L;
15
16     private final Toaster toaster;
17
18     public static void main(String[] args) {
19         new ConnectUI();
20     }
21
22     private ConnectUI() {
23         JPanel mainJPanel = getMainJPanel();
24
25         addLogo(mainJPanel);
26
27         addSeparator(mainJPanel);
28
29         addIPTextField(mainJPanel);
30
31         addPortTextField(mainJPanel);
32
33         addConnectButton(mainJPanel);
34
35         getContentPane().add(mainJPanel);
36
37         pack();
38         setVisible(true);
39         toFront();
40
41         Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
42         setLocation(screenSize.width / 2 - getWidth() / 2, screenSize.height / 2 -
43         getHeight() / 2);
44
45         toaster = new Toaster(mainJPanel);
46     }
47
48     private JPanel getMainJPanel() {
49         setUndecorated(true);
50
51         Dimension size = new Dimension(800, 400);
52
53         JPanel panel1 = new JPanel();
54         panel1.setSize(size);
```

```
54     panel1.setPreferredSize(size);
55     panel1.setBackground(UIUtils.COLOR_BACKGROUND);
56     panel1.setLayout(null);
57
58     TextFieldPort titel = new TextFieldPort();
59     titel.setEnabled(false);
60     titel.setEditable(false);
61     titel.setHorizontalAlignment(SwingConstants.CENTER);
62     titel.setBorderColor(UIUtils.COLOR_INTERACTIVE);
63     titel.setFont(new Font("Segoe UI", Font.PLAIN, 16));
64     titel.setText("OmniMove Wlan Verbindung");
65     titel.setBounds(68, 57, 250, 44);
66     panel1.add(titel);
67
68     panel1.setFocusable(true);
69     panel1.requestFocus();
70
71     MouseAdapter ma = new MouseAdapter() {
72         int lastX, lastY;
73
74         @Override
75         public void mousePressed(MouseEvent e) {
76             lastX = e.getXOnScreen();
77             lastY = e.getYOnScreen();
78         }
79
80         @Override
81         public void mouseDragged(MouseEvent e) {
82             int x = e.getXOnScreen();
83             int y = e.getYOnScreen();
84             setLocation(getLocationOnScreen().x + x - lastX,
85             getLocationOnScreen().y + y - lastY);
86             lastX = x;
87             lastY = y;
88         }
89     };
90
91     panel1.addMouseListener(ma);
92     panel1.addMouseMotionListener(ma);
93
94     addWindowListener(new WindowAdapter() {
95         @Override
96         public void windowClosing(WindowEvent e) {
```

```
96         System.exit(0);
97     }
98 );
99
100    return panel1;
101 }
102
103 private void addSeparator(JPanel panel1) {
104     JSeparator separator1 = new JSeparator();
105     separator1.setOrientation(SwingConstants.VERTICAL);
106     separator1.setForeground(UIUtils.COLOR_OUTLINE);
107     panel1.add(separator1);
108     separator1.setBounds(387, 41, 2, 316);
109 }
110
111 private void addLogo(JPanel panel1) {
112     JLabel label1 = new JLabel();
113     //label1.setSize(300,300);
114     label1.setFocusable(false);
115     label1.setBounds(57, 123, 273, 197);
116     //ImageIcon robot = new ImageIcon(new
117     ImageIcon("/robot.png").getImage().getScaledInstance(20, 20,
118     Image.SCALE_DEFAULT));
119     //label1.setIcon(robot);
120     Image robot = new
121     ImageIcon(this.getClass().getResource("/robot.png")).getImage();
122     label1.setIcon(new ImageIcon(robot));
123     panel1.add(label1);
124 }
125
126 private void addIPTextField(JPanel panel1) {
127     TextFieldPort usernameField = new TextFieldPort();
128     //usernameField.setBorderColor(new Color(102, 205, 170));
129
130     usernameField.setText(UIUtils.PLACEHOLDER_TEXT_IP);
131     usernameField.setForeground(UIUtils.COLOR_OUTLINE);
132     usernameField.setBorderColor(UIUtils.COLOR_OUTLINE);
133
134     usernameField.setBounds(473, 109, 250, 44);
135     usernameField.addFocusListener(new FocusListener() {
136         @Override
137         public void focusGained(FocusEvent e) {
138             if (usernameField.getText().equals(UIUtils.PLACEHOLDER_TEXT_IP)) {
```

```
136         usernameField.setText("“);
137     }
138     usernameField.setForeground(Color.white);
139     usernameField.setBorderColor(UIUtils.COLOR_INTERACTIVE);
140 }
141
142     @Override
143     public void focusLost(FocusEvent e) {
144         if (usernameField.getText().isEmpty()) {
145             usernameField.setText(UIUtils.PLACEHOLDER_TEXT_IP);
146         }
147         usernameField.setForeground(UIUtils.COLOR_OUTLINE);
148         usernameField.setBorderColor(UIUtils.COLOR_OUTLINE);
149     }
150 });
151
152     panel1.add(usernameField);
153 }
154
155     private void addPortTextField(JPanel panel1) {
156         TextFieldIP passwordField = new TextFieldIP();
157         //passwordField.setBorderColor(new Color(102, 205, 170));
158
159         passwordField.setText(UIUtils.PLACEHOLDER_TEXT_PORT);
160         passwordField.setForeground(UIUtils.COLOR_OUTLINE);
161         passwordField.setBorderColor(UIUtils.COLOR_OUTLINE);
162
163         passwordField.setBounds(473, 168, 250, 44);
164         passwordField.addFocusListener(new FocusListener() {
165             @Override
166             public void focusGained(FocusEvent e) {
167                 if (passwordField.getText().equals(UIUtils.PLACEHOLDER_TEXT_PORT))
168                 {
169                     passwordField.setText("“);
170                 }
171                 passwordField.setForeground(Color.white);
172                 passwordField.setBorderColor(UIUtils.COLOR_INTERACTIVE);
173             }
174
175             @Override
176             public void focusLost(FocusEvent e) {
177                 if (passwordField.getText().isEmpty()) {
178                     passwordField.setText(UIUtils.PLACEHOLDER_TEXT_PORT);
```

```
178     }
179     passwordField.setForeground(UIUtils.COLOR_OUTLINE);
180     passwordField.setBorderColor(UIUtils.COLOR_OUTLINE);
181   }
182 );
183
184 passwordField.addKeyListener(new KeyAdapter() {
185   @Override
186   public void keyTyped(KeyEvent e) {
187     if (e.getKeyChar() == KeyEvent.VK_ENTER)
188       loginEventHandler();
189   }
190 );
191
192 panel1.add(passwordField);
193 }
194
195 private void addConnectButton(JPanel panel1) {
196   final Color[] connectColors = {UIUtils.COLOR_INTERACTIVE, Color.white};
197
198   JLabel connectButton = new JLabel() {
199     /**
200      *
201     */
202     private static final long serialVersionUID = 1L;
203
204     @Override
205     protected void paintComponent(Graphics g) {
206       Graphics2D g2 = UIUtils.get2dGraphics(g);
207       super.paintComponent(g2);
208
209       Insets insets = getInsets();
210       int w = getWidth() - insets.left - insets.right;
211       int h = getHeight() - insets.top - insets.bottom;
212       g2.setColor(connectColors[0]);
213       g2.fillRoundRect(insets.left, insets.top, w, h, UIUtils.ROUNDNESS,
214                         UIUtils.ROUNDNESS);
215
216       FontMetrics metrics = g2.getFontMetrics(UIUtils.FONT_GENERAL_UI);
217       int x2 = (getWidth() -
218 metrics.stringWidth(UIUtils.BUTTON_TEXT_CONNECT)) / 2;
219       int y2 = ((getHeight() - metrics.getHeight()) / 2) +
220 metrics.getAscent();
```

```
218         g2.setFont(UIUtils.FONT_GENERAL_UI);
219         g2.setColor(connectColors[1]);
220         g2.drawString(UIUtils.BUTTON_TEXT_CONNECT, x2, y2);
221     }
222 };
223
224 connectButton.addMouseListener(new MouseAdapter() {
225
226     @Override
227     public void mousePressed(MouseEvent e) {
228         loginEventHandler();
229     }
230
231     @Override
232     public void mouseEntered(MouseEvent e) {
233         connectColors[0] = UIUtils.COLOR_INTERACTIVE_DARKER;
234         connectColors[1] = UIUtils.OFFWHITE;
235         connectButton.repaint();
236     }
237
238     @Override
239     public void mouseExited(MouseEvent e) {
240         connectColors[0] = UIUtils.COLOR_INTERACTIVE;
241         connectColors[1] = Color.white;
242         connectButton.repaint();
243     }
244 });
245
246 connectButton.setBackground(UIUtils.COLOR_BACKGROUND);
247 connectButton.setBounds(473, 247, 250, 44);
248 connectButton.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
249 panel1.add(connectButton);
250 }
251
252 private void loginEventHandler() {
253     // hier Connect Code reinschreiben
254     toaster.warn("Login event");
255 }
256 }
```

Listing A.11: main/ConnectUI.java

```
1 package components;
2
3 import javax.swing.SwingUtilities;
4 import javax.swing.JFrame;
5 import javax.swing.JPanel;
6 import javax.swing.JComponent;
7
8 import java.awt.BorderLayout;
9 import java.awt.Graphics;
10 import java.awt.Color;
11 import java.awt.Dimension;
12 import javax.swing.border.EmptyBorder;
13
14 public class SpeedMeter extends JComponent {
15     private int meterWidth = 10;
16
17     private float speed = 0f;
18     private boolean breaks = false;
19     private boolean colorfade = false;
20
21     private Color background = Color.LIGHT_GRAY;
22     private Color border = Color.BLACK;
23     private Color fill = Color.GREEN;
24     private Color maxSpeedFill = Color.ORANGE;
25     private Color breakfill = Color.RED;
26
27
28     public void setSpeed(float speed) {
29         this.speed = speed;
30         repaint();
31     }
32
33     public void setBreaks(boolean breaks) {
34         this.breaks = breaks;
35         repaint();
36     }
37
38     public void setColorfade(boolean colorfade) {
39         this.colorfade = colorfade;
40     }
41
42     public void setMaxSpeedFill(Color maxSpeedFill) {
43         this.maxSpeedFill = maxSpeedFill;
```

```
44     }
45
46     public boolean isColorfade() {
47         return colorfade;
48     }
49
50     public Color getMaxSpeedFill() {
51         return maxSpeedFill;
52     }
53
54     public void toggleBreaks() {
55         setBreaks(!breaks);
56     }
57
58     public void setMeterWidth(int meterWidth) {
59         this.meterWidth = meterWidth;
60     }
61
62     public void setBackgroundFill(Color background) {
63         this.background = background;
64         repaint();
65     }
66
67     public void setBorderFill(Color border) {
68         this.border = border;
69         repaint();
70     }
71
72     public void setFill(Color fill) {
73         this.fill = fill;
74         repaint();
75     }
76
77     public void setBreakfill(Color breakfill) {
78         this.breakfill = breakfill;
79         repaint();
80     }
81
82     public Color getBackgroundFill() {
83         return background;
84     }
85
86     public Color getBorderFill() {
```

```
87         return border;
88     }
89
90     public Color getFill() {
91         return fill;
92     }
93
94     public Color getBreakfill() {
95         return breakfill;
96     }
97
98     @Override
99     protected void paintComponent(Graphics g) {
100         int w = Math.min(meterWidth, getWidth());
101         int h = getHeight();
102         int x = getWidth() / 2 - w / 2;
103         int y = 0;
104
105         g.setColor(background);
106         g.fillRect(x, y, w, h);
107
108         g.setColor(border);
109         g.drawRect(x, y, w - 1, h - 1);
110
111         int a = Math.round(speed * (h - 2));
112         if (colorfade)
113             g.setColor(new Color(Math.min(fill.getRed() +
114             Math.round((maxSpeedFill.getRed() - fill.getRed()) * (Math.abs(speed))), 255),
115             Math.min(fill.getGreen() + Math.round((maxSpeedFill.getGreen() -
116             fill.getGreen()) * (Math.abs(speed))), 255),
117             Math.min(fill.getBlue() + Math.round((maxSpeedFill.getBlue() -
118             fill.getBlue()) * (Math.abs(speed))), 255),
119             Math.min(fill.getAlpha() + Math.round((maxSpeedFill.getAlpha() -
120             fill.getAlpha()) * (Math.abs(speed))), 255)));
121         else
122             g.setColor(fill);
123         if (a > 0)
124             g.fillRect(x + 1, y + h / 2 - a / 2, w - 2, a / 2);
125         else
126             g.fillRect(x + 1, y + h / 2 - 1, w - 2, -(a / 2) + 1);
127
128         int p = Math.round(h / 2);
129         g.setColor(breakfill);
```

```
126     g.drawLine(x + 1, y + h - 1 - p, x + w - 1, y + h - 1 - p);  
127  
128     if (breaks) {  
129         g.setColor(breakfill);  
130         g.fillRect(x + 1, y + 1, w - 2, h / 6);  
131         g.fillRect(x + 1, y + h - 1 - h / 6, w - 2, h / 6);  
132     }  
133 }  
134  
135 @Override  
136 public Dimension getMinimumSize() {  
137     Dimension min = super.getMinimumSize();  
138     if (min.width < meterWidth)  
139         min.width = meterWidth;  
140     if (min.height < meterWidth)  
141         min.height = meterWidth;  
142     return min;  
143 }  
144  
145 @Override  
146 public Dimension getPreferredSize() {  
147     Dimension pref = super.getPreferredSize();  
148     pref.width = meterWidth;  
149     return pref;  
150 }  
151  
152 @Override  
153 public void setPreferredSize(Dimension pref) {  
154     super.setPreferredSize(pref);  
155     setMeterWidth(pref.width);  
156 }  
157  
158 public static void main(String[] args) {  
159     SwingUtilities.invokeLater(() -> {  
160         JFrame frame = new JFrame("Meter");  
161         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
162  
163         JPanel content = new JPanel(new BorderLayout());  
164         content.setBorder(new EmptyBorder(25, 50, 25, 50));  
165  
166         SpeedMeter meter = new SpeedMeter();  
167         meter.setPreferredSize(new Dimension(9, 100));  
168         content.add(meter, BorderLayout.CENTER);
```

```
169         frame.setContentPane(content);
170         frame.pack();
171         frame.setLocationRelativeTo(null);
172         frame.setVisible(true);
173
174         new Thread(new SpeedTester(meter)).start();
175     });
176 }
177
178 static class SpeedTester implements Runnable {
179     final SpeedMeter meter;
180
181     SpeedTester(final SpeedMeter meter) {
182         this.meter = meter;
183     }
184
185     @Override
186     public void run() {
187         meter.setColorfade(true);
188         float speed = 0;
189         while (true) {
190             if (speed > 1) {
191                 speed = -1;
192                 meter.toggleBreaks();
193                 try {
194                     Thread.sleep(1000);
195                 } catch (InterruptedException e) {
196                     e.printStackTrace();
197                 }
198             } else
199                 speed += 0.01;
200             meter.setSpeed(speed);
201             meter.setBreaks(false);
202             try {
203                 Thread.sleep(20);
204             } catch (InterruptedException e) {
205                 e.printStackTrace();
206             }
207         }
208     }
209 }
210 }
211 }
```

Listing A.12: components/SpeedMeter.java

```
1 package Toaster;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.MouseAdapter;
6 import java.awt.event.MouseEvent;
7 import java.util.ArrayList;
8 import java.util.concurrent.atomic.AtomicInteger;
9
10 public class Toaster {
11     private static final int STARTING_Y_POS = 15;
12     private static final int SPACER_DISTANCE = 15;
13     private static final ArrayList<ToasterBody> toasterBodies = new ArrayList<>();
14     private final static AtomicInteger CURRENT_Y_OFFSET = new AtomicInteger();
15     private final JPanel panelToToastOn;
16
17     public Toaster(JPanel panelToToastOn) {
18         this.panelToToastOn = panelToToastOn;
19     }
20
21     public void error(String... messages) {
22         for (String s : messages) {
23             toast(s, new Color(181, 59, 86));
24         }
25     }
26
27     public void success(String... messages) {
28         for (String s : messages) {
29             toast(s, new Color(33, 181, 83));
30         }
31     }
32
33     public void info(String... messages) {
34         for (String s : messages) {
35             toast(s, new Color(13, 116, 181));
36         }
37     }
38 }
```

```
39     public void warn(String... messages) {
40         for (String s : messages) {
41             toast(s, new Color(181, 147, 10));
42         }
43     }
44
45     private void toast(String message, Color bgColor) {
46         ToasterBody toasterBody;
47
48         if (toasterBodies.isEmpty()) {
49             toasterBody = new ToasterBody(panelToToastOn, message, bgColor,
50 STARTING_Y_POS);
51             CURRENT_Y_OFFSET.set(STARTING_Y_POS + toasterBody.getHeightOfToast());
52         } else {
53             toasterBody = new ToasterBody(panelToToastOn, message, bgColor,
54 CURRENT_Y_OFFSET.get() + SPACER_DISTANCE);
55             CURRENT_Y_OFFSET.addAndGet(SPACER_DISTANCE +
56 toasterBody.getHeightOfToast());
57         }
58
59         toasterBodies.add(toasterBody);
60
61         new Thread(() -> {
62             toasterBody.addMouseListener(new MouseAdapter() {
63                 @Override
64                 public void mousePressed(MouseEvent e) {
65                     removeToast(toasterBody);
66                 }
67             });
68
69             try {
70                 Thread.sleep(6000);
71                 removeToast(toasterBody);
72             } catch (InterruptedException e) {
73                 e.printStackTrace();
74             }
75         }).start();
76     }
77
78     private synchronized void removeToast(ToasterBody toasterBody) {
```

```
79     if (!toasterBody.getStopDisplaying()) {
80         toasterBody.setStopDisplaying(true);
81
82         toasterBodies.forEach(toasterBody1 -> {
83             if (toasterBodies.indexOf(toasterBody1) >=
84                 toasterBodies.indexOf(toasterBody)) {
85                 toasterBody1.setyPos(toasterBody1.getyPos() -
86                 toasterBody.getHeightOfToast() - SPACER_DISTANCE);
87             }
88         });
89
90         CURRENT_Y_OFFSET.set(CURRENT_Y_OFFSET.get() - SPACER_DISTANCE -
91             toasterBody.getHeightOfToast());
92
93         panelToToastOn.remove(toasterBody);
94         panelToToastOn.repaint();
95     }
96 }
```

Listing A.13: Toaster/Toaster.java

```
1 package Toaster;
2
3 import Utils.UIUtils;
4
5 import javax.swing.*;
6 import java.awt.*;
7
8 class ToasterBody extends JPanel {
9     /**
10      *
11      */
12     private static final long serialVersionUID = 1L;
13     private static final int TOAST_PADDING = 15;
14     private final int toastWidth;
15     private final String message;
16     private final Color c;
17     private volatile boolean stopDisplaying;
18     private int heightOfToast, stringPosX, stringPosY, yPos;
```

```
19     private JPanel panelToToastOn;
20
21     public ToasterBody(JPanel panelToToastOn, String message, Color bgColor, int
22         yPos) {
23
24         this.panelToToastOn = panelToToastOn;
25         this.message = message;
26         this.yPos = yPos;
27         this.c = bgColor;
28
29
30         FontMetrics metrics = getFontMetrics(UIUtils.FONT_GENERAL_UI);
31         int stringWidth = metrics.stringWidth(this.message);
32
33
34         toastWidth = stringWidth + (TOAST_PADDING * 2);
35         heightOfToast = metrics.getHeight() + TOAST_PADDING;
36         setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
37         setOpaque(false);
38         setBounds((panelToToastOn.getWidth() - toastWidth) / 2, (int)
39             -(Math.round(heightOfToast / 10.0) * 10), toastWidth, heightOfToast);
40
41
42         stringPosX = (getWidth() - stringWidth) / 2;
43         stringPosY = ((getHeight() - metrics.getHeight()) / 2) +
44             metrics.getAscent();
45
46
47         new Thread(() -> {
48             while (getBounds().y < yPos) {
49                 int i1 = (yPos - getBounds().y) / 10;
50                 i1 = i1 <= 0 ? 1 : i1;
51                 setBounds((panelToToastOn.getWidth() - toastWidth) / 2,
52                     getBounds().y + i1, toastWidth, heightOfToast);
53                 repaint();
54                 try {
55                     Thread.sleep(5);
56                 } catch (Exception ignored) {
57                 }
58             }
59         }).start();
60     }
61
62
63     @Override
64     protected void paintComponent(Graphics g) {
65         Graphics2D g2 = UIUtils.get2dGraphics(g);
66         super.paintComponent(g2);
67     }
68 }
```

```
58     //Background
59     g2.setColor(c);
60     g2.fillRoundRect(0, 0, getWidth(), getHeight(), UIUtils.ROUNDNESS,
61     UIUtils.ROUNDNESS);
62
63     // Font
64     g2.setFont(UIUtils.FONT_GENERAL_UI);
65     g2.setColor(Color.white);
66     g2.drawString(message, stringPosX, stringPosY);
67 }
68
69 public int getHeightOfToast() {
70     return heightOfToast;
71 }
72
73 public synchronized boolean getStopDisplaying() {
74     return stopDisplaying;
75 }
76
77 public synchronized void setStopDisplaying(boolean hasStoppedDisplaying) {
78     this.stopDisplaying = hasStoppedDisplaying;
79 }
80
81 public void setyPos(int yPos) {
82     this.yPos = yPos;
83     // setBounds(panelToToastOn.getWidth() - toastWidth) / 2, yPos,
84     // toastWidth, heightOfToast);
85
86     new Thread(() -> {
87         while (getBounds().y > yPos) {
88             int i1 = Math.abs((yPos - getBounds().y) / 10);
89             i1 = i1 <= 0 ? 1 : i1;
90             setBounds((panelToToastOn.getWidth() - toastWidth) / 2,
91             getBounds().y - i1, toastWidth, heightOfToast);
92             repaint();
93             try {
94                 Thread.sleep(5);
95             } catch (Exception ignored) {
96             }
97         }
98     }).start();
99 }
```

```
98     public int getyPos() {
99         return yPos;
100    }
101 }
```

Listing A.14: Toaster/ToasterBody.java

```
1 package Utils;
2
3 import java.awt.*;
4 import java.util.HashMap;
5
6 public class UIUtils {
7     public static final Font FONT_GENERAL_UI = new Font("Segoe UI", Font.PLAIN,
8         20);
9
10    public static final Color COLOR_OUTLINE = new Color(103, 112, 120);
11    public static final Color COLOR_BACKGROUND = new Color(37, 51, 61);
12    public static final Color COLOR_INTERACTIVE = new Color(108, 216, 158);
13    public static final Color COLOR_INTERACTIVE_DARKER = new Color(87, 171, 127);
14    public static final Color OFFWHITE = new Color(229, 229, 229);
15
16    public static final String BUTTON_TEXT_CONNECT = "Connect";
17
18    public static final String Bild = "robot.png";
19
20    public static final String PLACEHOLDER_TEXT_IP = "IP Adresse";
21    public static final String PLACEHOLDER_TEXT_PORT = "PORT";
22
23    public static final int ROUNDNESS = 8;
24
25    public static Graphics2D get2dGraphics(Graphics g) {
26        Graphics2D g2 = (Graphics2D) g;
27        g2.addRenderingHints(new HashMap<RenderingHints.Key, Object>() {/*
28            *
29            */
30            private static final long serialVersionUID = 1L;
31
32            {
33                put(RenderingHints.KEY_ANTIALIASING,
34                    RenderingHints.VALUE_ANTIALIAS_ON);
35            }
36        });
37    }
38}
```

```
33         put(RenderingHints.KEY_TEXT_ANTIALIASING,
34             RenderingHints.VALUE_TEXT_ANTIALIAS_LCD_HRGB);
35     });
36     return g2;
37 }
```

Listing A.15: Utils/UIUtils.java

```
1 package Utils;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.RoundRectangle2D;
6
7
8 public class TextFieldIP extends JTextField {
9     /**
10      *
11     */
12     private static final long serialVersionUID = 1L;
13
14     private Shape shape;
15     private Color borderColor = UIUtils.COLOR_OUTLINE;
16
17     public TextFieldIP() {
18         setOpaque(false);
19         setBackground(UIUtils.COLOR_BACKGROUND);
20         setForeground(Color.white);
21         setCaretColor(Color.white);
22         setCursor(Cursor.getPredefinedCursor(Cursor.TEXT_CURSOR));
23         setMargin(new Insets(2, 10, 2, 2));
24         setHorizontalAlignment(SwingConstants.LEFT);
25        setFont(UIUtils.FONT_GENERAL_UI);
26     }
27
28     protected void paintComponent(Graphics g) {
29         Graphics2D g2 = UIUtils.get2dGraphics(g);
30         g2.setColor(getBackground());
31         g2.fillRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
32             UIUtils.ROUNDNESS);
33         super.paintComponent(g2);
```

```

33 }
34
35     protected void paintBorder(Graphics g) {
36         Graphics2D g2 = UIUtils.get2dGraphics(g);
37         g2.setColor(borderColor);
38         g2.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
39             UIUtils.ROUNDNESS);
40     }
41
42     public boolean contains(int x, int y) {
43         if (shape == null || !shape.getBounds().equals(getBounds())) {
44             shape = new RoundRectangle2D.Float(0, 0, getWidth() - 1, getHeight() -
45             1, UIUtils.ROUNDNESS, UIUtils.ROUNDNESS);
46         }
47         return shape.contains(x, y);
48     }
49
50     public void setBorderColor(Color color) {
51         borderColor = color;
52         repaint();
53     }
54 }
```

Listing A.16: Utils/TextFieldIP.java

```

1 package Utils;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.geom.RoundRectangle2D;
6
7
8 public class TextFieldPort extends JTextField {
9     /**
10      *
11     */
12     private static final long serialVersionUID = 1L;
13
14     private Shape shape;
15     private Color borderColor = UIUtils.COLOR_INTERACTIVE;
16
17     public TextFieldPort() {
```

```

18     setOpaque(false);
19     setBackground(UIUtils.COLOR_BACKGROUND);
20     setForeground(Color.white);
21     setCaretColor(Color.white);
22     setCursor(Cursor.getPredefinedCursor(Cursor.TEXT_CURSOR));
23     setMargin(new Insets(2, 10, 2, 2));
24     setHorizontalAlignment(SwingConstants.LEFT);
25     setFont(UIUtils.FONT_GENERAL_UI);
26 }
27
28 protected void paintComponent(Graphics g) {
29     Graphics2D g2 = UIUtils.get2dGraphics(g);
30     g2.setColor(getBackground());
31     g2.fillRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
32                     UIUtils.ROUNDNESS);
33     super.paintComponent(g2);
34 }
35
36 protected void paintBorder(Graphics g) {
37     Graphics2D g2 = UIUtils.get2dGraphics(g);
38     g2.setColor(borderColor);
39     g2.drawRoundRect(0, 0, getWidth() - 1, getHeight() - 1, UIUtils.ROUNDNESS,
40                      UIUtils.ROUNDNESS);
41 }
42
43 public boolean contains(int x, int y) {
44     if (shape == null || !shape.getBounds().equals(getBounds())) {
45         shape = new RoundRectangle2D.Float(0, 0, getWidth() - 1, getHeight() -
46         1, UIUtils.ROUNDNESS, UIUtils.ROUNDNESS);
47     }
48     return shape.contains(x, y);
49 }
50
51 public void setBorderColor(Color color) {
52     borderColor = color;
53     repaint();
54 }
55
56 }

```

Listing A.17: Utils/TextFieldPort.java

```
1 package zweitesFenster;
```

```
2
3 public class Main {
4
5     public static void main(String[] args) {
6         new ZweitesFenster();
7
8     }
9
10 }
```

Listing A.18: main/Main.java

```
1 package zweitesFenster;
2
3 import java.awt.BorderLayout;
4 import java.awt.Dimension;
5 import java.awt.Font;
6 import java.awt.Image;
7 import java.awt.Toolkit;
8 import java.awt.event.MouseAdapter;
9 import java.awt.event.MouseEvent;
10 import java.awt.event.WindowAdapter;
11 import java.awt.event.WindowEvent;
12 import javax.swing.ImageIcon;
13 import javax.swing.JFrame;
14 import javax.swing.JLabel;
15 import javax.swing.JPanel;
16 import javax.swing.JProgressBar;
17 import javax.swing.JSlider;
18 import javax.swing.JTextField;
19 import javax.swing.SwingConstants;
20 import Utils.TextFieldPort;
21 import Utils.UIUtils;
22 import java.awt.Color;
23
24 public class ZweitesFenster extends JFrame{
25
26     /**
27      *
28      */
29
30     private static final long serialVersionUID = 1L;
```

```
31
32
33     ZweitesFenster(){
34
35
36         JPanel mainPanel = getMainJPanel();
37
38         addLogo(mainPanel);
39         battery(mainPanel);
40         speed(mainPanel);
41
42         getContentPane().add(mainPanel, BorderLayout.SOUTH);
43
44         pack();
45         setVisible(true);
46         toFront();
47
48         Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
49         setLocation(screenSize.width / 2 - getWidth() / 2, screenSize.height / 2 -
50         getHeight() / 2);
51
52
53
54     private JPanel getMainJPanel() {
55         setUndecorated(true);
56
57         Dimension size = new Dimension(700, 600);
58
59         JPanel panel1 = new JPanel();
60         panel1.setSize(size);
61         panel1.setPreferredSize(size);
62         panel1.setBackground(UIUtils.COLOR_BACKGROUND);
63         panel1.setLayout(null);
64
65         TextFieldPort titel = new TextFieldPort();
66         titel.setEnabled(false);
67         titel.setEditable(false);
68         titel.setHorizontalAlignment(SwingConstants.CENTER);
69         //titel.setBorderColor(UIUtils.COLOR_INTERACTIVE);
70         titel.setFont(new Font("Segoe UI", Font.PLAIN, 14));
71         titel.setText("OmniMove Driving Experience");
72         titel.setBounds(20, 25, 250, 45);
```

```
73     panel1.add(titel);
74     panel1.setFocusable(true);
75     panel1.requestFocus();
76
77     MouseAdapter ma = new MouseAdapter() {
78         int lastX, lastY;
79
80         @Override
81         public void mousePressed(MouseEvent e) {
82             lastX = e.getXOnScreen();
83             lastY = e.getYOnScreen();
84         }
85
86         @Override
87         public void mouseDragged(MouseEvent e) {
88             int x = e.getXOnScreen();
89             int y = e.getYOnScreen();
90             setLocation(getLocationOnScreen().x + x - lastX,
91             getLocationOnScreen().y + y - lastY);
92             lastX = x;
93             lastY = y;
94         }
95     };
96
97     panel1.addMouseListener(ma);
98     panel1.addMouseMotionListener(ma);
99
100    addWindowListener(new WindowAdapter() {
101        @Override
102        public void windowClosing(WindowEvent e) {
103            System.exit(0);
104        }
105    });
106
107    return panel1;
108
109
110
111    private void addLogo(JPanel panel1) {
112        JLabel label1 = new JLabel();
113        label1.setFocusable(false);
114        label1.setVisible(true);
```

```
115         label1.setBounds(201,94,320,369);
116         Image robot = new
117             ImageIcon(this.getClass().getResource("/robot2.png")).getImage();
118         label1.setIcon(new ImageIcon(robot));
119         panel1.add(label1);
120
121
122     private void battery(JPanel panel2) {
123         int akkustand;
124         akkustand = 50;
125
126         JProgressBar battery_number = new JProgressBar();
127         battery_number.setBackground(UIUtils.COLOR_BACKGROUND);
128         battery_number.setForeground(UIUtils.COLOR_INTERACTIVE);
129         battery_number.setEnabled(false);
130         battery_number.setLocation(531, 47);
131         battery_number.setMinimum(0);
132         battery_number.setMaximum(100);
133         battery_number.setFocusable(false);
134         battery_number.setSize(133,45);
135         battery_number.setString("Akkustand: "+akkustand+" %");
136         battery_number.setStringPainted(true);
137         battery_number.setFont(new Font("Segoe UI", Font.PLAIN, 12));
138
139         //default values
140         battery_number.setValue(50);
141
142         panel2.add(battery_number);
143
144     }
145
146
147     private void speed(JPanel panel3) {
148
149         int kmh = 10;
150
151         JPanel combination = new JPanel();
152         combination.setSize(222, 101);
153         combination.setVisible(true);
154         combination.setBackground(UIUtils.COLOR_BACKGROUND);
155
156         JLabel geschwindigkeit = new JLabel();
```

```
157     geschwindigkeit.setHorizontalAlignment(SwingConstants.CENTER);
158     geschwindigkeit.setText("Geschwindigkeit");
159     geschwindigkeit.setFont(new Font("Segoe UI", Font.PLAIN, 12));
160     geschwindigkeit.setFocusable(false);
161     geschwindigkeit.setEnabled(false);
162
163     JSlider speed = new JSlider(JSlider.HORIZONTAL, 0, 30, 0);
164     speed.setSize(123, 35);
165     speed.setBackground(UIUtils.COLOR_BACKGROUND);
166     speed.setEnabled(false);
167     speed.setFocusable(false);
168     speed.setLocation(292, 495);
169     speed.setMajorTickSpacing(5);
170     speed.setMinorTickSpacing(1);
171     speed.setPaintTicks(true);
172     speed.setPaintLabels(true);
173
174     speed.setMaximum(30);
175     speed.setMinimum(0);
176     speed.setValue(kmh);
177
178     JLabel geschwindigkeit2 = new JLabel();
179     geschwindigkeit2.setHorizontalAlignment(SwingConstants.CENTER);
180     geschwindigkeit2.setText(" " + kmh + " km/h");
181     geschwindigkeit2.setVisible(true);
182     geschwindigkeit2.setFont(new Font("Segoe UI", Font.PLAIN, 12));
183     geschwindigkeit2.setFocusable(false);
184     geschwindigkeit2.setEnabled(false);
185
186     combination.setLayout(new BorderLayout());
187     combination.add(geschwindigkeit, BorderLayout.NORTH);
188     combination.add(speed, BorderLayout.CENTER);
189     combination.add(geschwindigkeit2, BorderLayout.SOUTH);
190     combination.setLocation(264, 488);
191
192     panel3.add(combination);
193
194 }
195
196
197 }
```

Listing A.19: main/ZweitesFenster.java

B Technische Zeichnungen

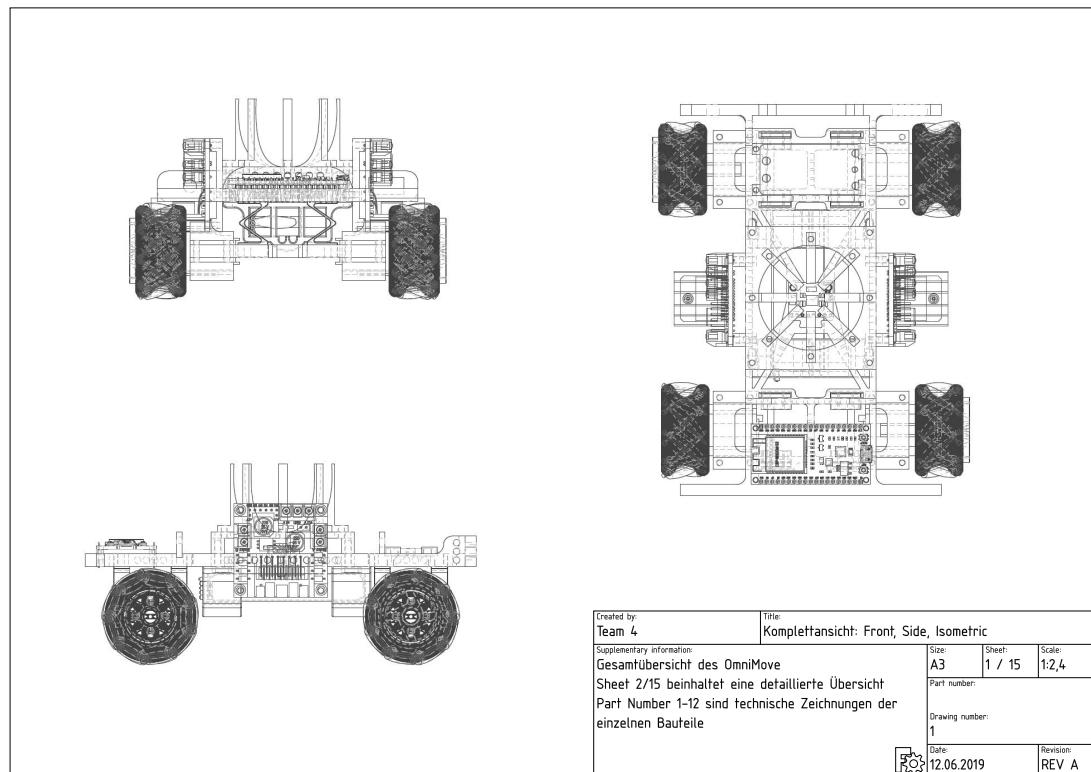


Abbildung B.1: Komplettansicht

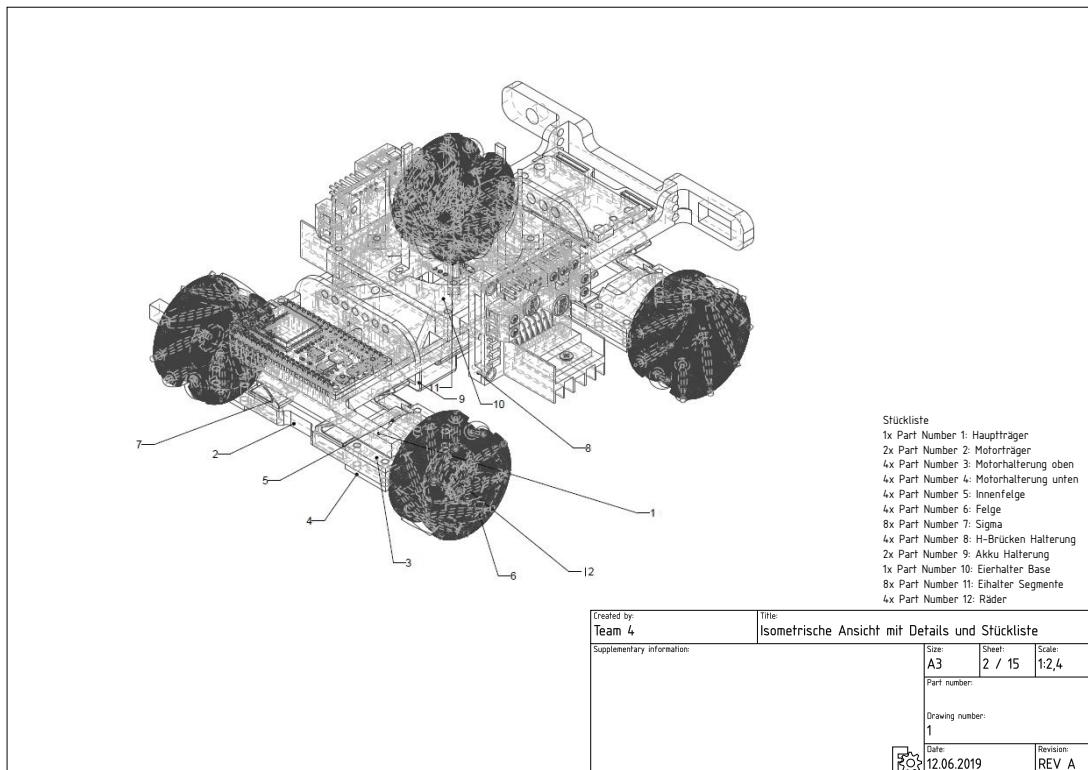


Abbildung B.2: Isometrisch und Stückliste

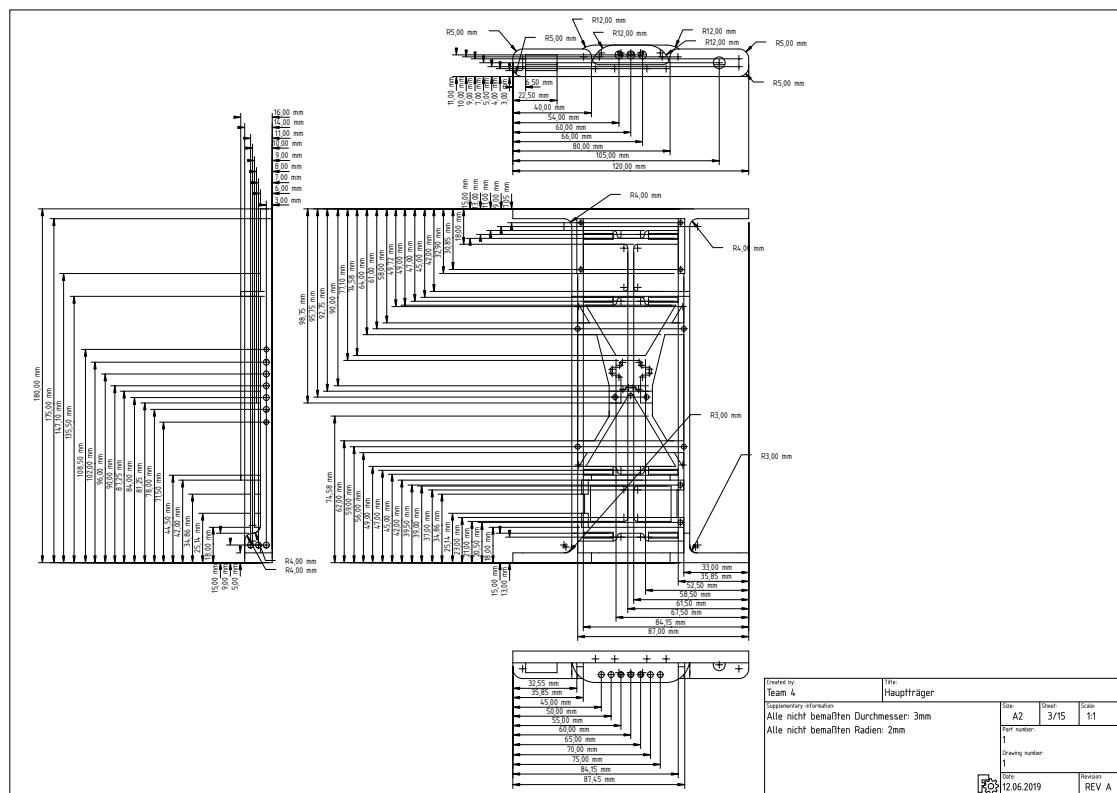


Abbildung B.3: Hauptträger

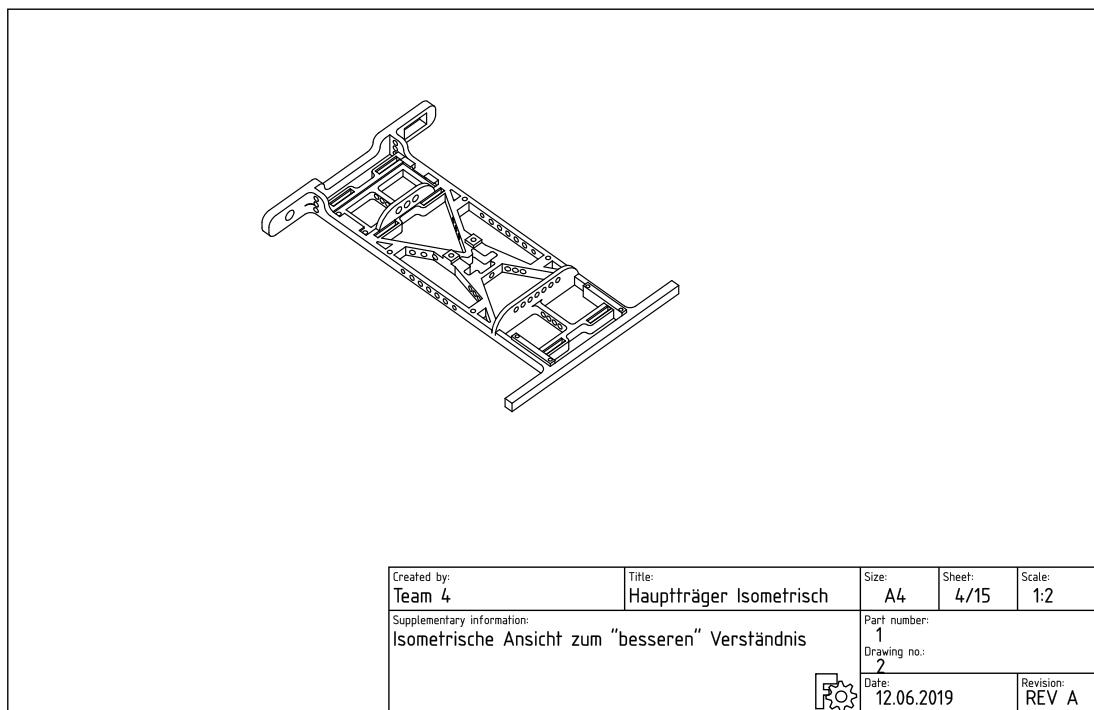


Abbildung B.4: Hauptträger Isometrisch

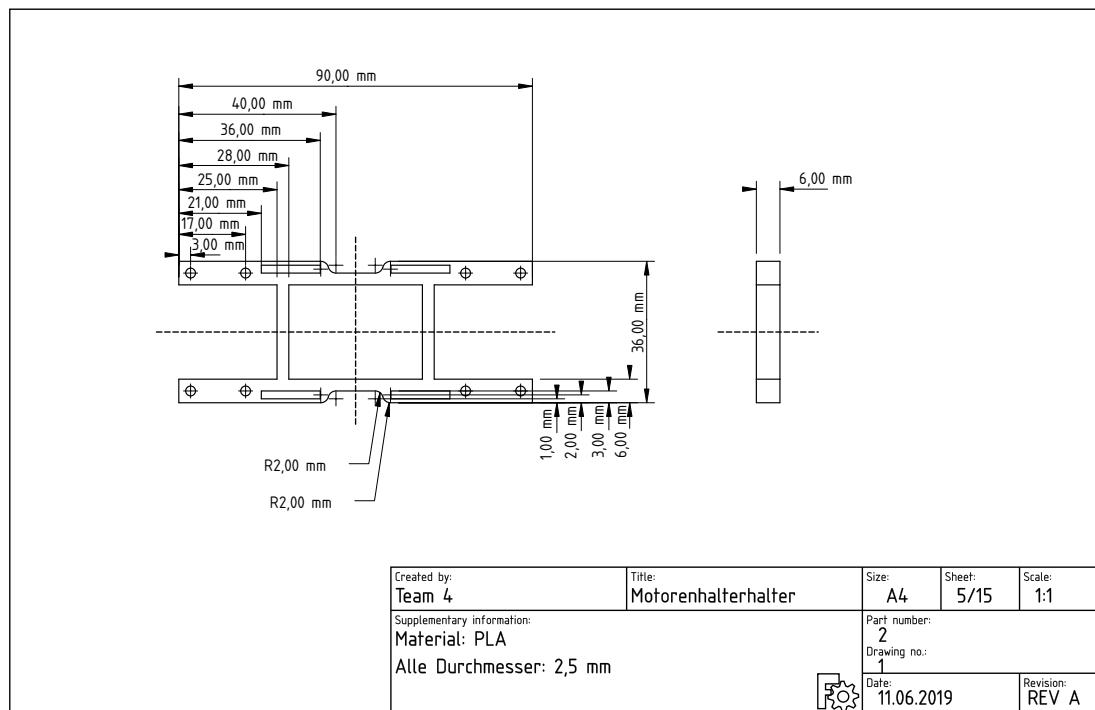


Abbildung B.5: Motorenhalterhalter

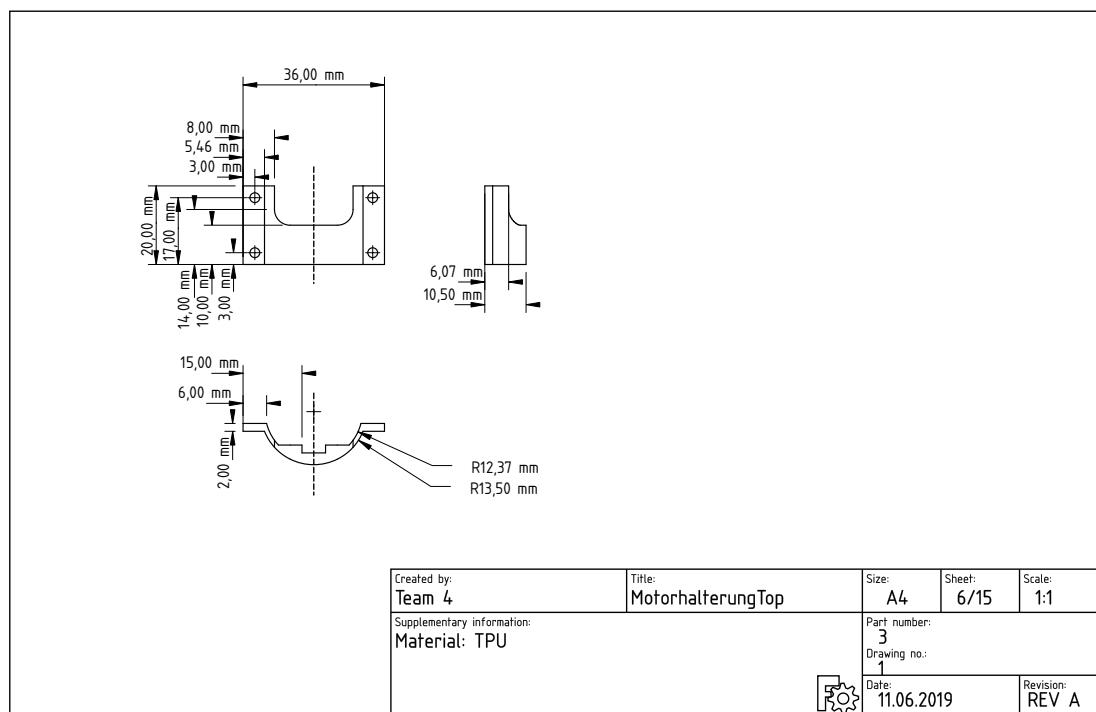


Abbildung B.6: Motorenhalterung Oben

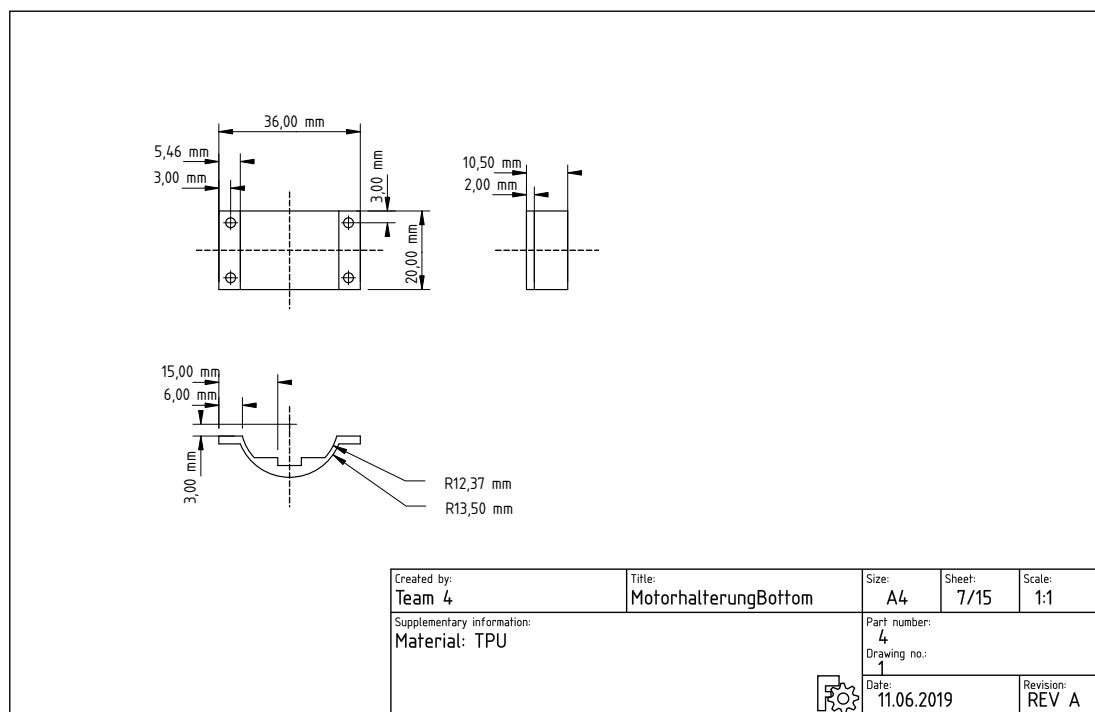


Abbildung B.7: Motorenhalterung Unten

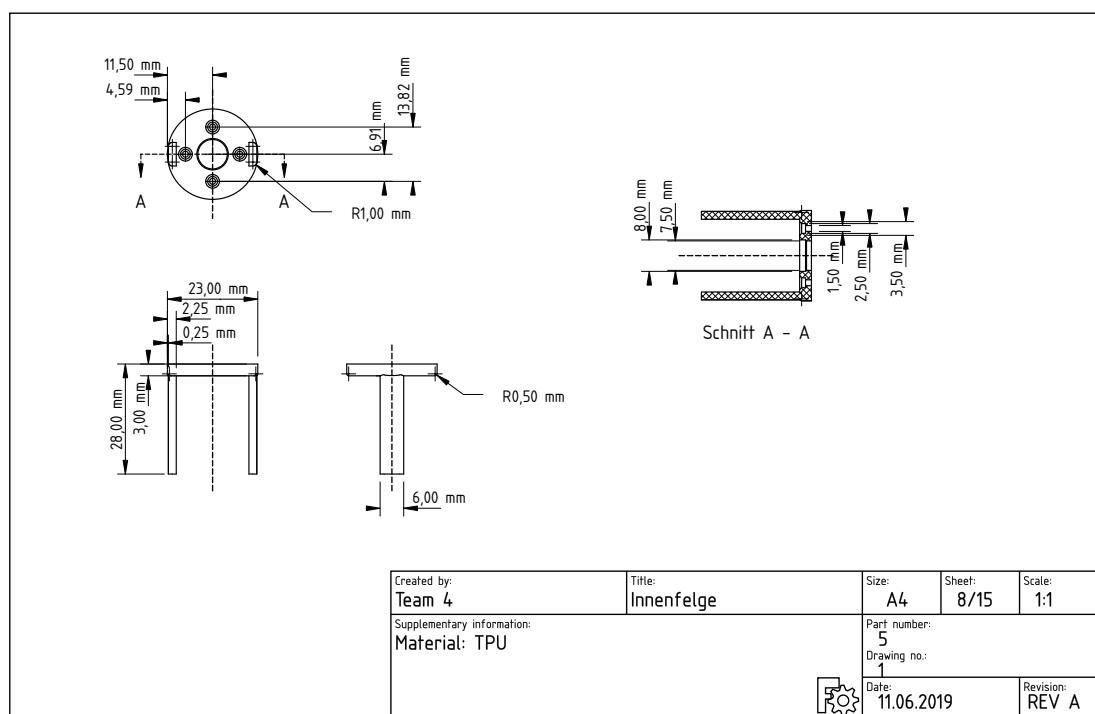


Abbildung B.8: Innenfelge

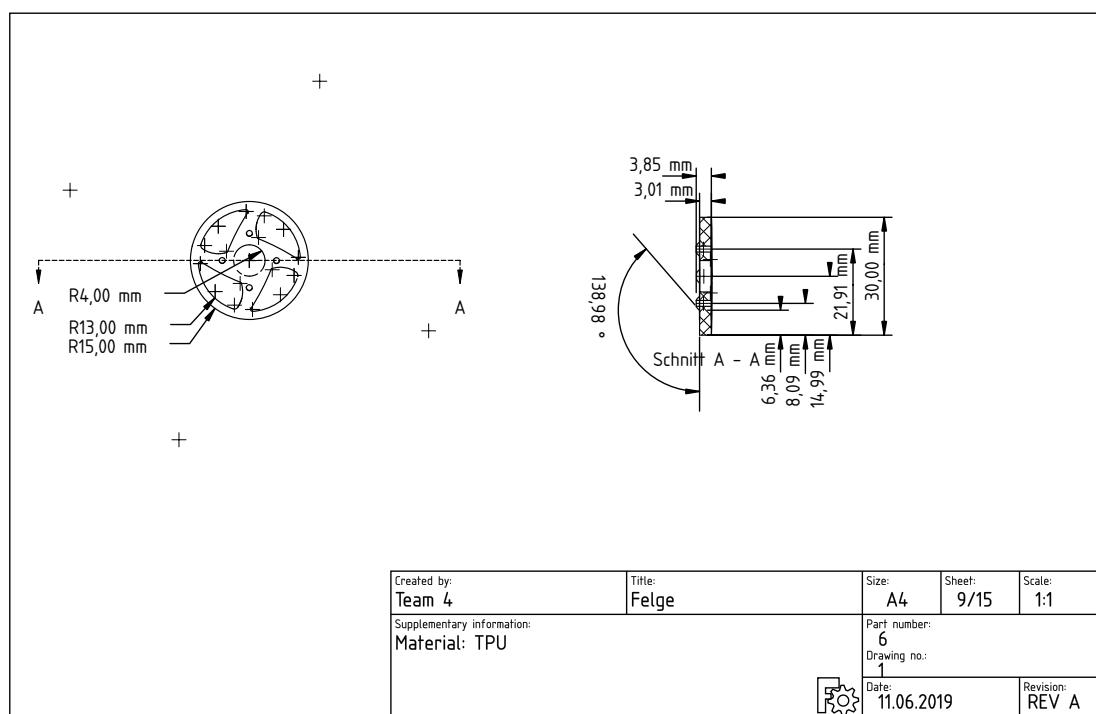


Abbildung B.9: Felge

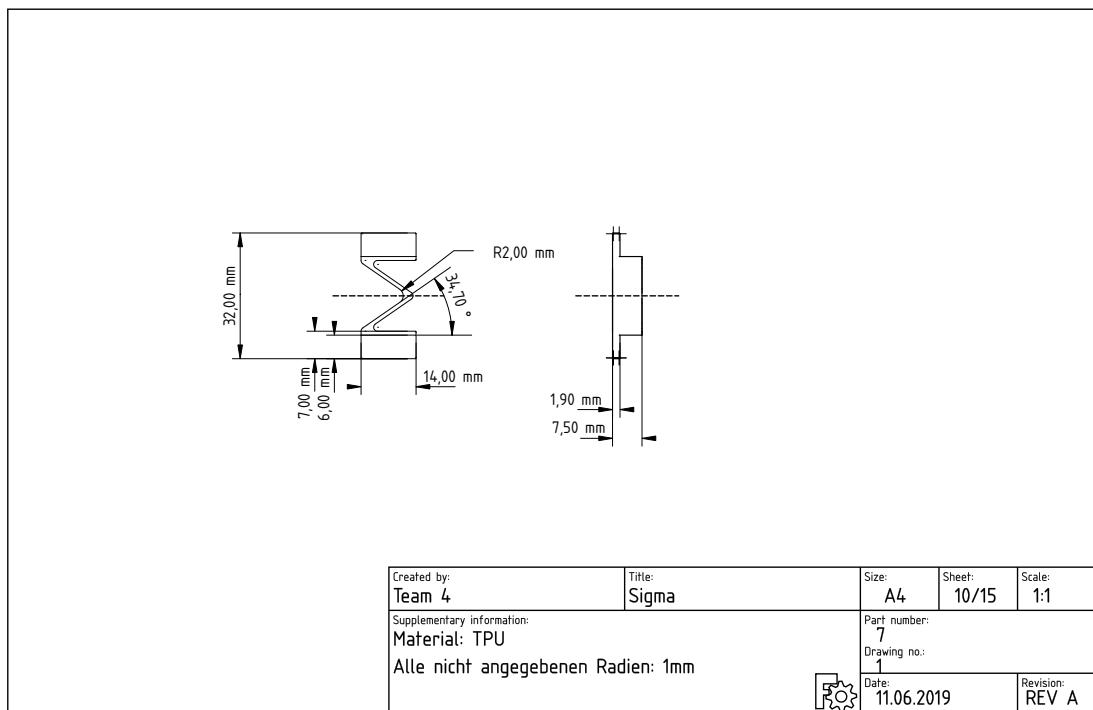


Abbildung B.10: Sigma

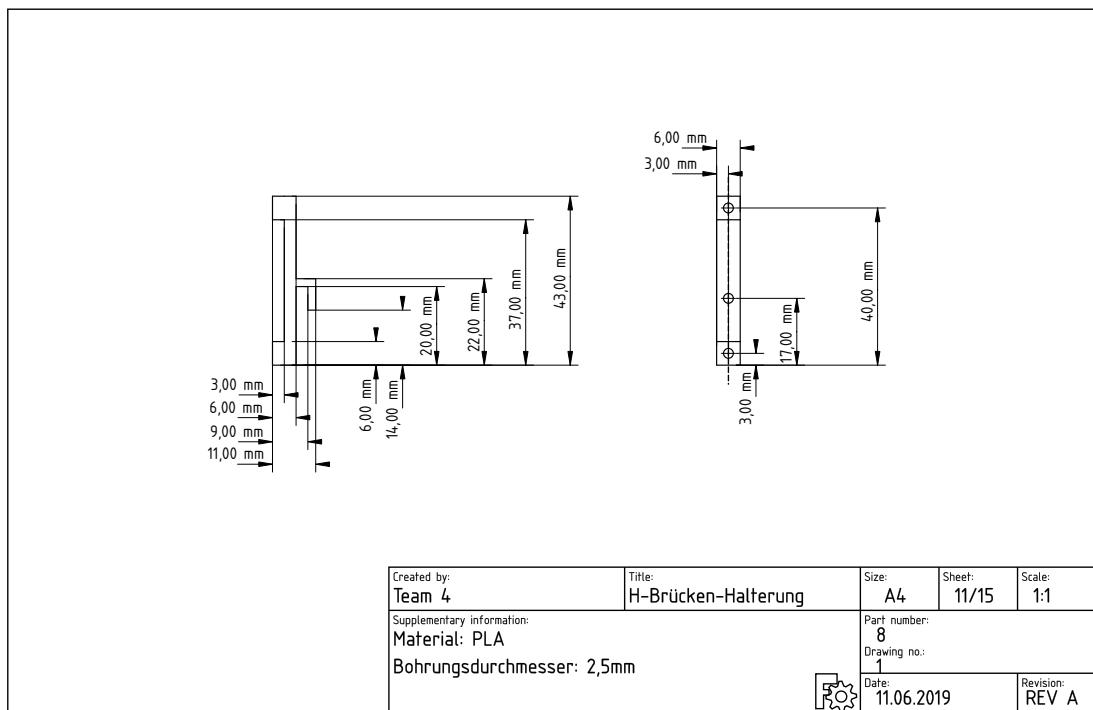


Abbildung B.11: H-Brücken-Halterung

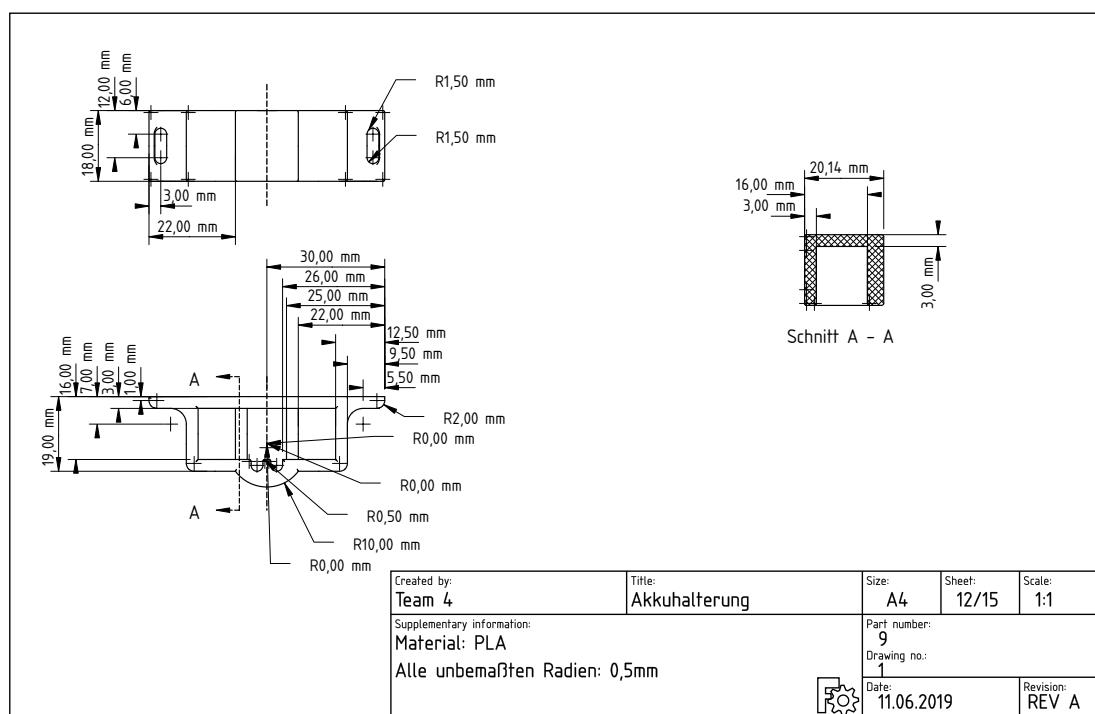


Abbildung B.12: Akkuhalterung

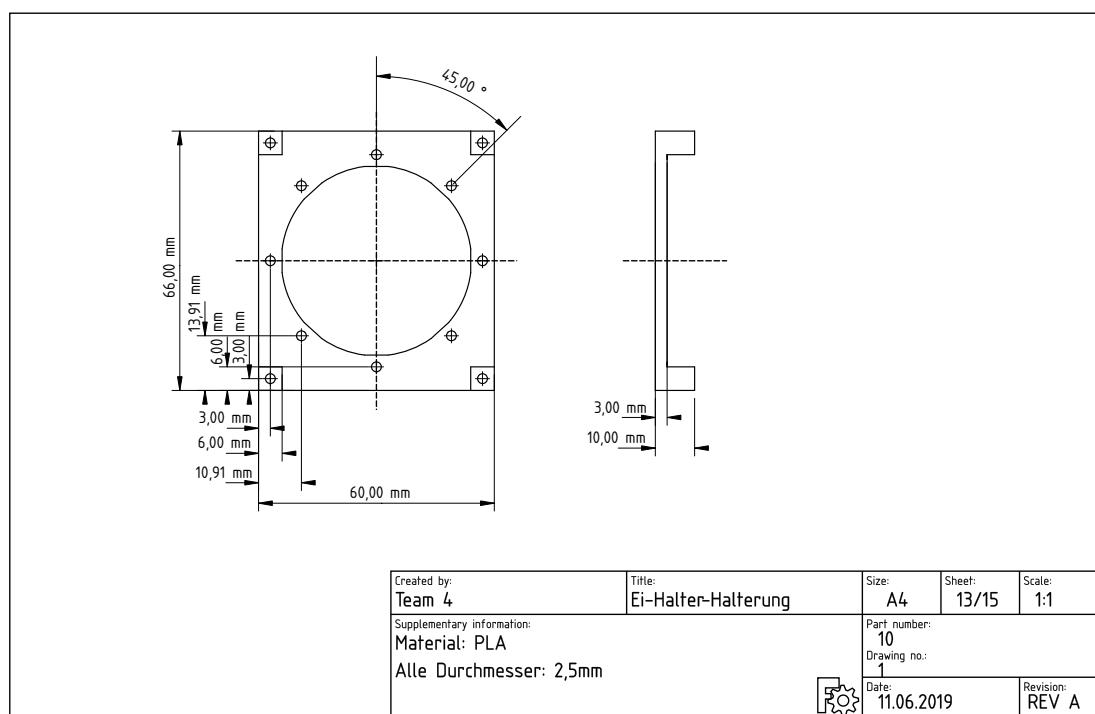


Abbildung B.13: Ei-Halterung-Halterung

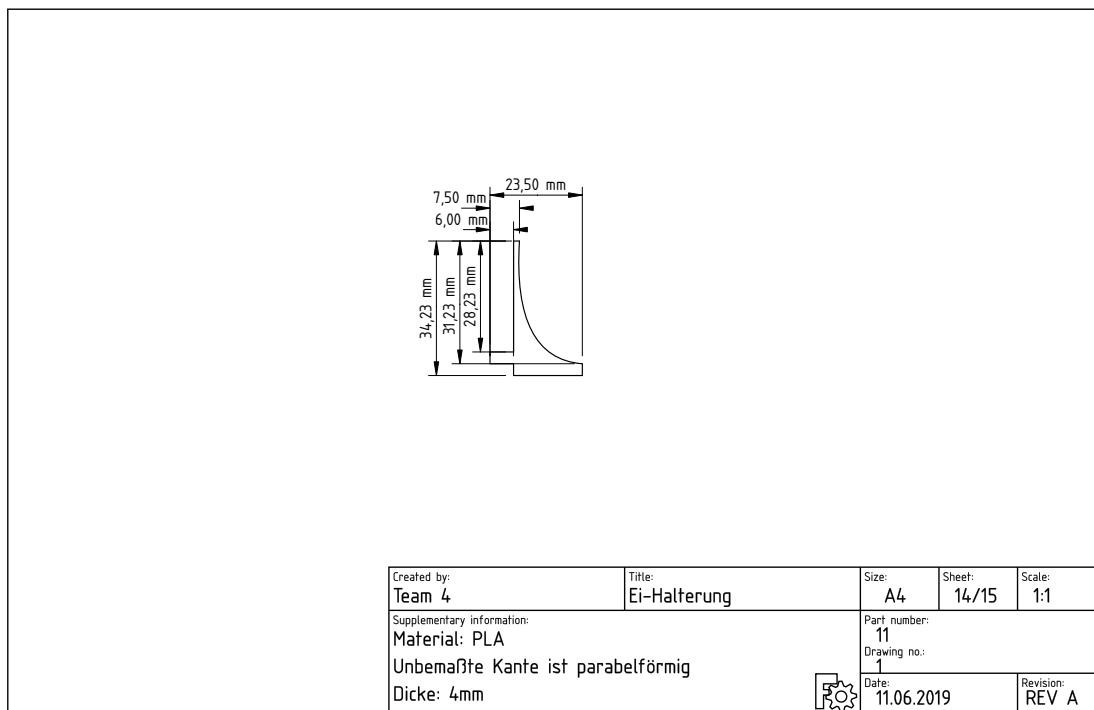


Abbildung B.14: Ei-Halterung

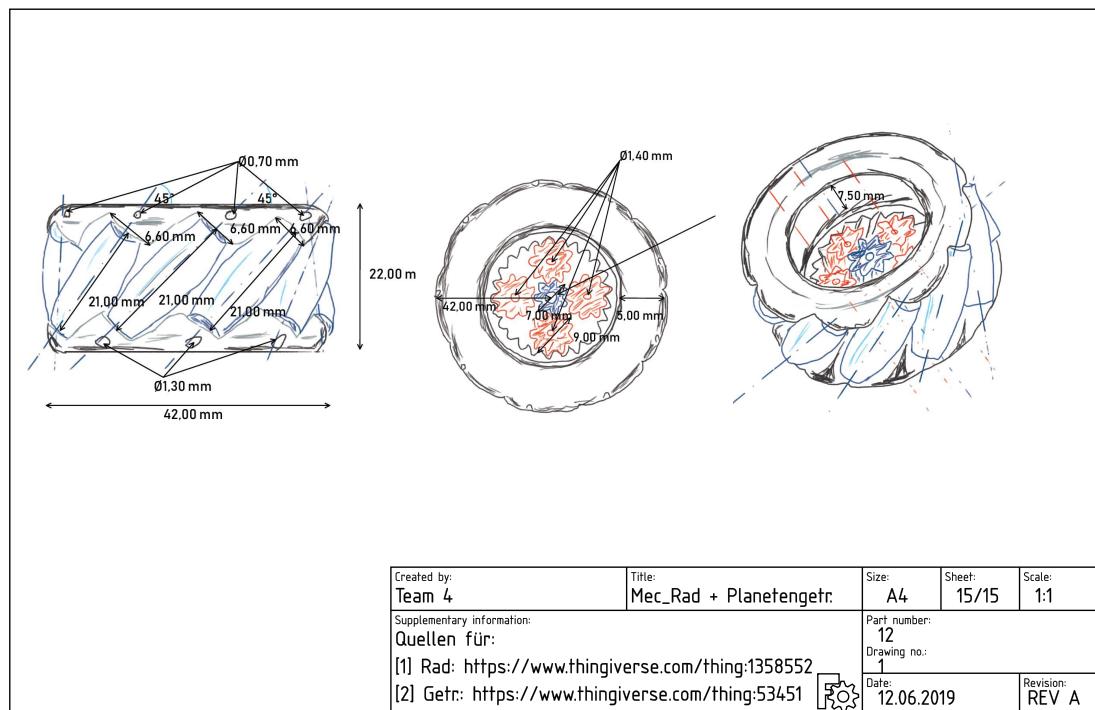


Abbildung B.15: Reifen und Planetengetriebe

Listings

A.1	communication.cpp	13
A.2	communication.h	14
A.3	movement.cpp	15
A.4	movement.h	19
A.5	util.cpp	21
A.6	util.h	23
A.7	main.cpp	23
A.8	index.html	26
A.9	code.js	29
A.10	style.css	36
A.11	main/ConnectUI.java	37
A.12	components/SpeedMeter.java	44
A.13	Toaster/Toaster.java	49
A.14	Toaster/ToasterBody.java	51
A.15	Utils/UIUtils.java	54
A.16	Utils/TextFieldIP.java	55
A.17	Utils/TextFieldPort.java	56
A.18	main/Main.java	57
A.19	main/ZweitesFenster.java	58

Abbildungsverzeichnis

4.1	Bildschirmfoto Weboberfläche	7
B.1	Komplettansicht	64
B.2	Isometrisch und Stückliste	65
B.3	Hauptträger	66
B.4	Hauptträger Isometrisch	67
B.5	Motorenhalterhalter	68
B.6	Motorenhalterung Oben	69
B.7	Motorenhalterung Unten	70
B.8	Innenfelge	71
B.9	Felge	72
B.10	Sigma	73
B.11	H-Brücken-Halterung	74
B.12	Akkuhalterung	75
B.13	Ei-Halterung-Halterung	76
B.14	Ei-Halterung	77
B.15	Reifen und Planetengetriebe	78

Tabellenverzeichnis

3.1 Kostenübersicht	4
4.1 Binäres Protokoll	8