



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	3
4 Entwicklung der Schlüsselemente	4
4.1 Übersicht des gesamten Modells	4
4.2 Schaltplan	4
4.3 BMS und Laden	4
4.4 Mecanum	4
4.5 Planetengetriebe	4
4.6 ESP-32 vs. ESP-8266	4
4.7 User-Interface	4
4.7.1 Java (obsolet)	4
4.7.2 HTML5 und Controller-Anbindung	4
4.7.3 Protokoll	6
4.8 Steuerung per (XBox) Controller	7
4.9 PLA vs. TPU	7

4.10 Eierhalter	7
4.11 Leichtbau	7
5 Zukünftige Entwicklungsmöglichkeiten	8
5.1 Regler	8
5.2 Verbesserungen	8
5.2.1 Hardware	8
5.2.2 Software	8
A Quellcode	10
A.1 ESP32	10
A.2 Webpage	23
B Technische Zeichnungen	35
Listings	50
Abbildungsverzeichnis	51
Tabellenverzeichnis	52

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

3.4 Verwendete Technologien

3.4.1 Git

3.4.2 Creo

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 rawSSockets, ohne ein eigenes Framework 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.

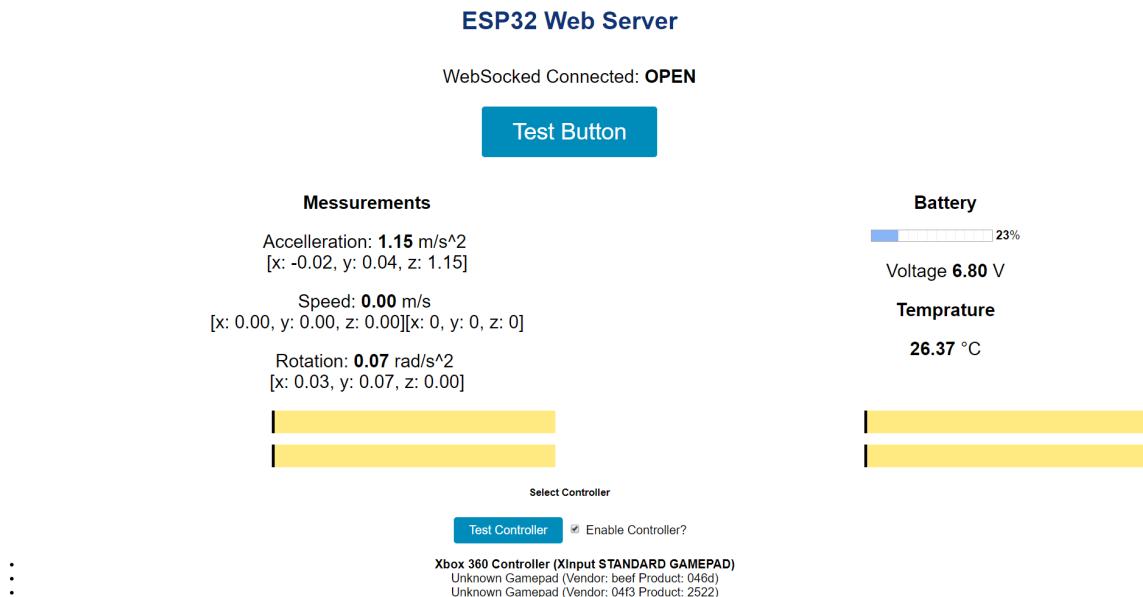


Abbildung 4.1: Bildschirmfoto Weboberfläche

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	Accel - X	Accel - Y	Accel - Z	Rot - X	Rot - Y	Rot - Z	Battery	Temp
DRIVE	0												
GYRO	1	Speed - X	Speed - Y	Speed - Z									
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

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
32 // ##### Attributes #####
33 Motor Movement::MOTOR_VL = Motor(CH_MOTOR_VL, PIN_MOTOR_VL_DIR1,
34     PIN_MOTOR_VL_DIR2);
35 Motor Movement::MOTOR_VR = Motor(CH_MOTOR_VR, PIN_MOTOR_VR_DIR1,
36     PIN_MOTOR_VR_DIR2);
37 Motor Movement::MOTOR_HL = Motor(CH_MOTOR_HL, PIN_MOTOR_HL_DIR1,
38     PIN_MOTOR_HL_DIR2);
39 Motor Movement::MOTOR_HR = Motor(CH_MOTOR_HR, PIN_MOTOR_HR_DIR1,
40     PIN_MOTOR_HR_DIR2);
41
42 // ##### Functions #####
43
44 void Movement::initPWM() {
45     Serial.println("Initing PWMs");
46
47     // Setting all pins to output
48     pinMode(PIN_LED, OUTPUT);
49     pinMode(PIN_MOTOR_VL_EN, OUTPUT);
50     pinMode(PIN_MOTOR_VL_DIR1, OUTPUT);
51     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:
121 // %6.2f°/s, gz: %6.2f°/s, temp: %6.2f°C\r", x / 16384.0, y / 16384.0, z /
122 16384.0, gx / 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 id="pathHR" d="M155,8 L205,8 105,8 155,8" stroke="#FFEA82"  
59     stroke-width="4" fill-opacity="0"  
60     style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"/></path>  
61     <path d="M 50 0 L 50 4" stroke="black" stroke-width="0.5" fill="none" />  
62     <path d="M 50 6 L 50 10" stroke="black" stroke-width="0.5" fill="none" />  
63     <path d="M 155 0 L 155 4" stroke="black" stroke-width="0.5" fill="none" />  
64     <path d="M 155 6 L 155 10" stroke="black" stroke-width="0.5" fill="none" />  
65   </svg>  
66 </div>  
67 <div>  
68   <h5>Select Controller</h5>  
69   <button class="buttonsmall" onclick="updateControllerList()">Test  
70   Controller</button>  
71   <input type="checkbox" id="controllerEnabled" checked>  
72   <label for="controllerEnabled">Enable Controller?</label>  
73   <ul id="controllerList">  
74     <li onclick="onControllerSelected(0)">Test</li>  
75   </ul>  
76 </div>  
77 </body>  
78 </html>  
79 <script type="text/javascript" src="progressbar.min.js"></script>  
80 <script type="text/javascript" src="code.js"></script>  
81 </body>  
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);
```



```

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

B Technische Zeichnungen

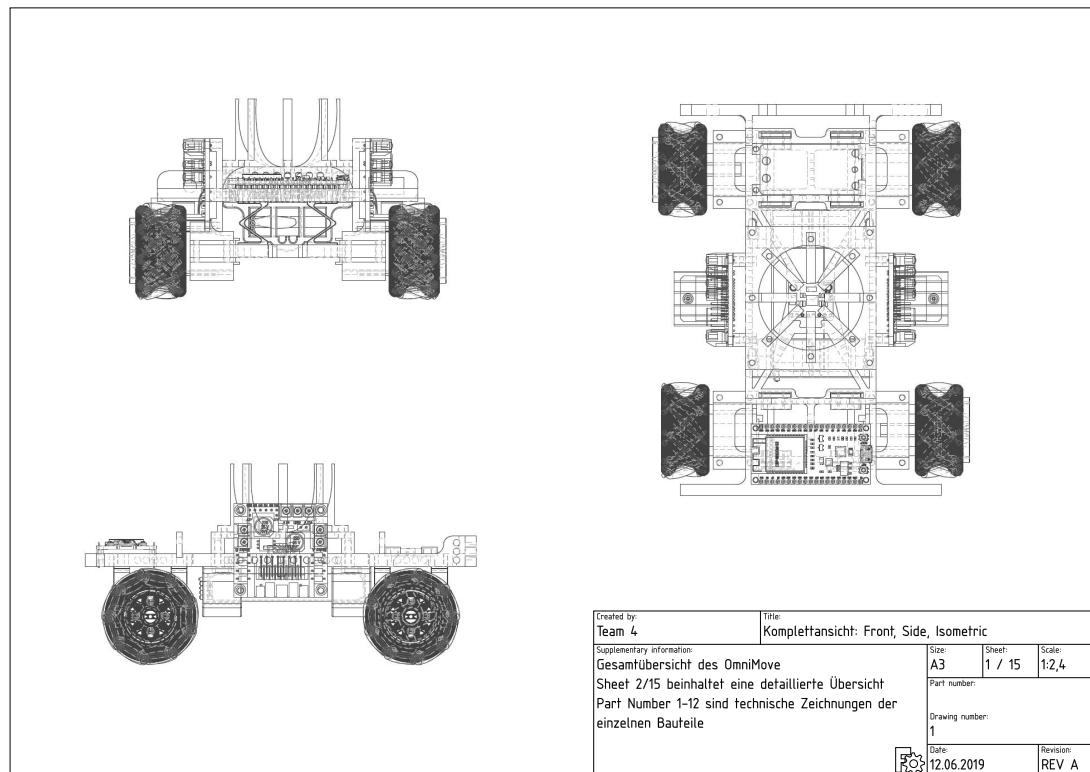


Abbildung B.1: Komplettansicht

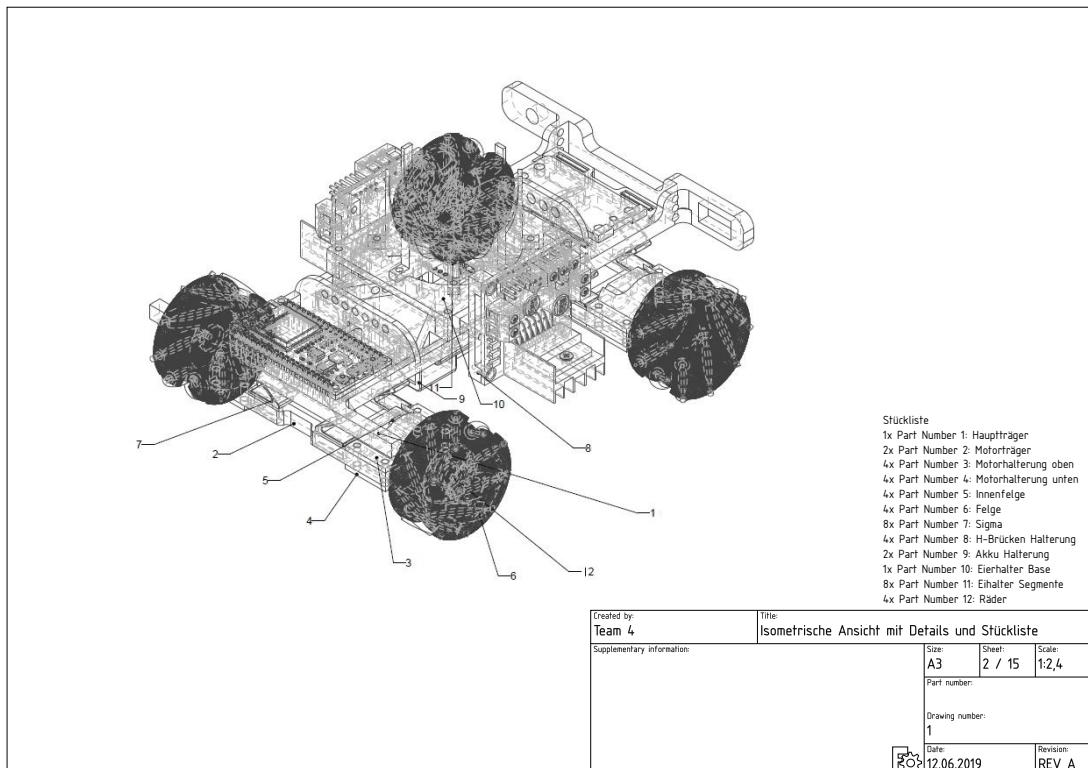


Abbildung B.2: Isometrisch und Stückliste

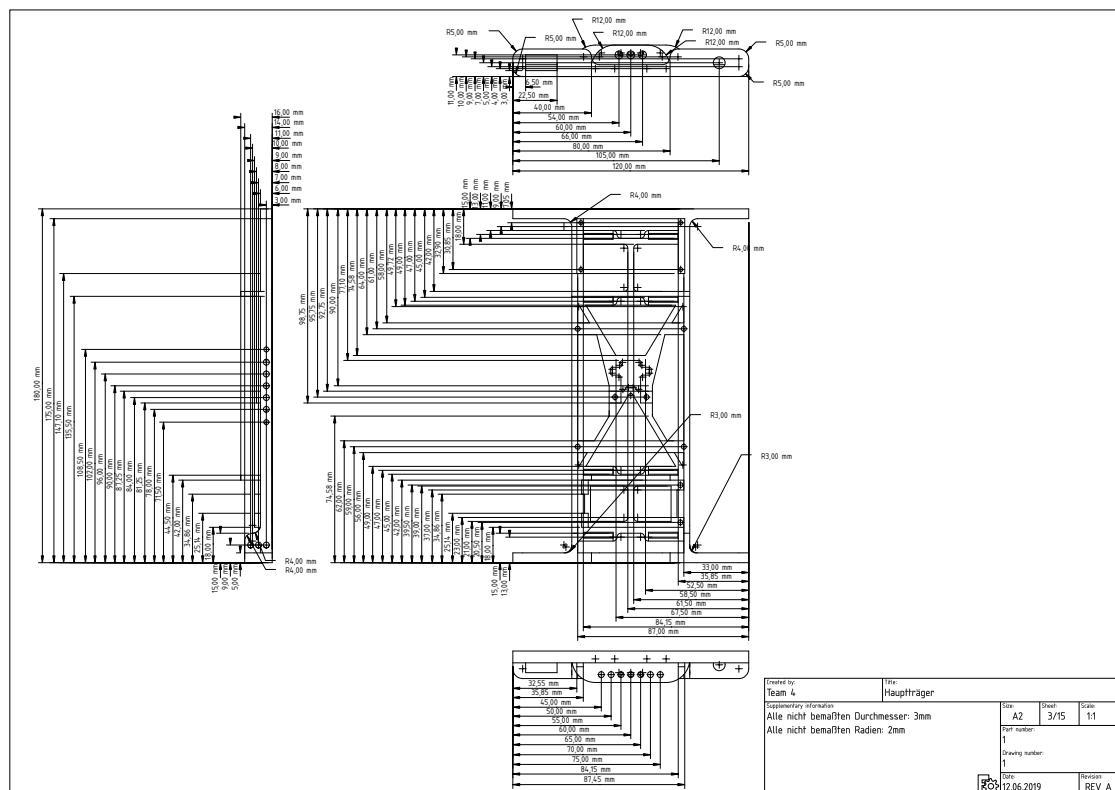


Abbildung B.3: Hauptträger

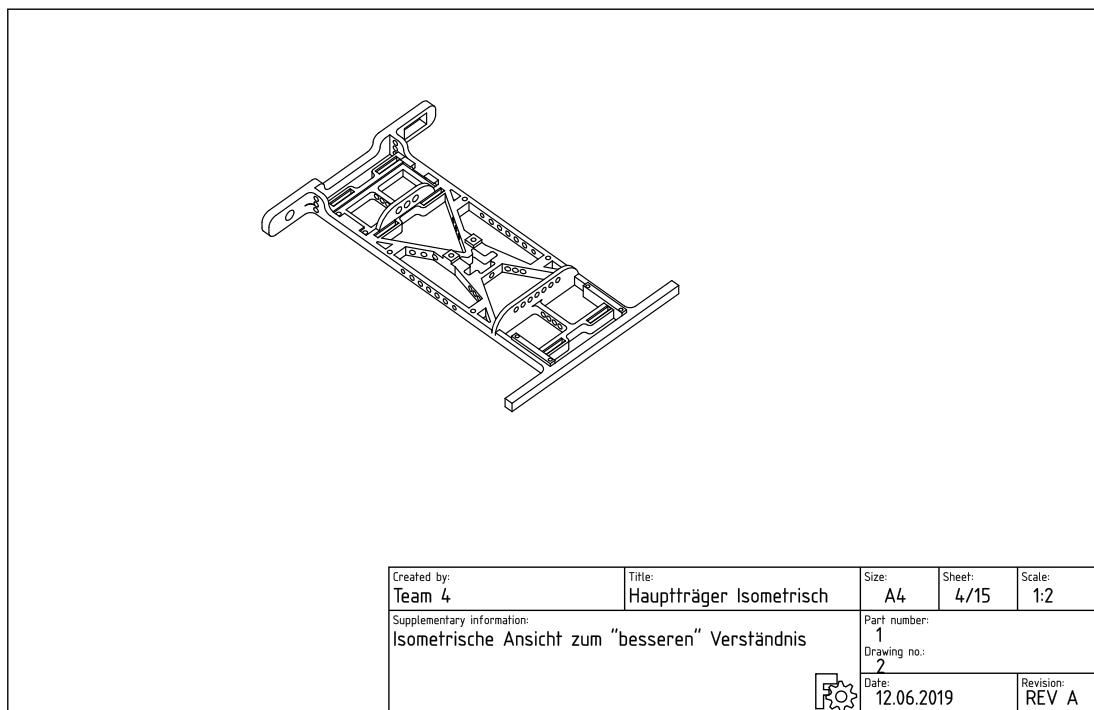


Abbildung B.4: Hauptträger Isometrisch

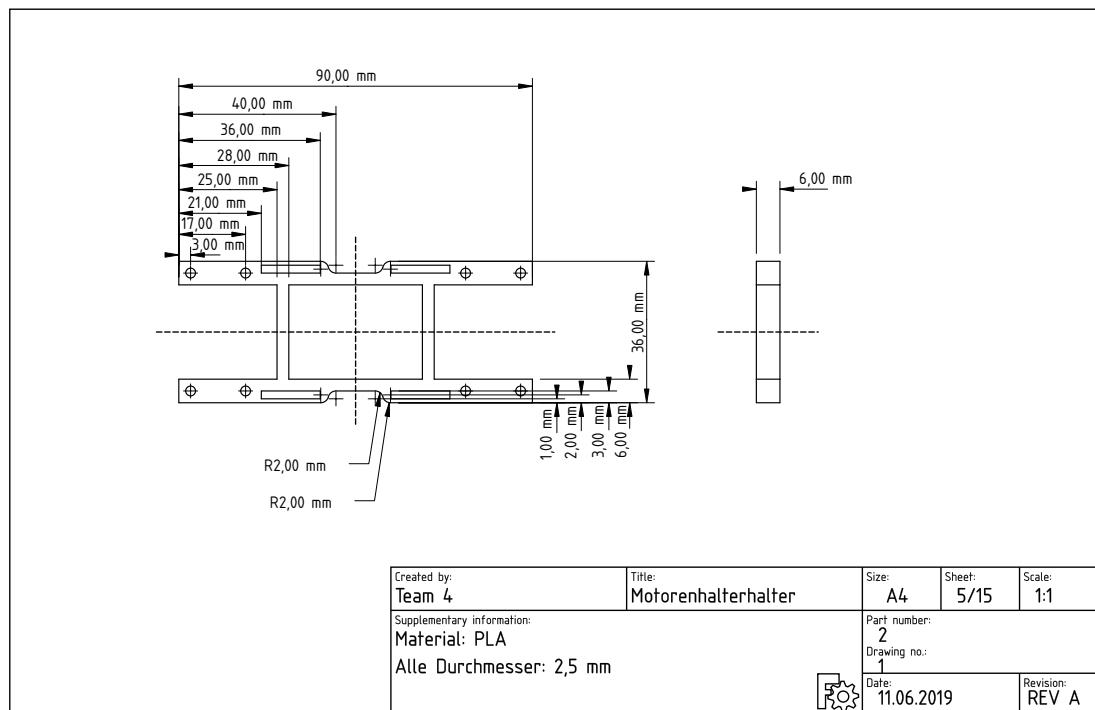


Abbildung B.5: Motorenhalterhalter

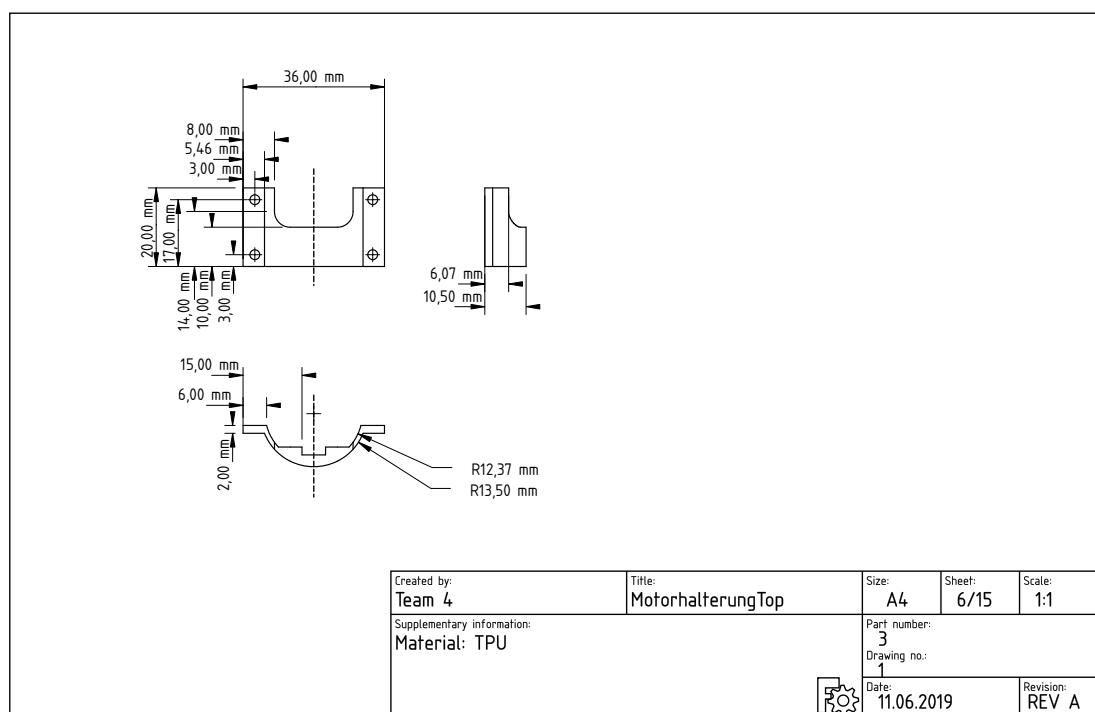


Abbildung B.6: Motorenhalterung Oben

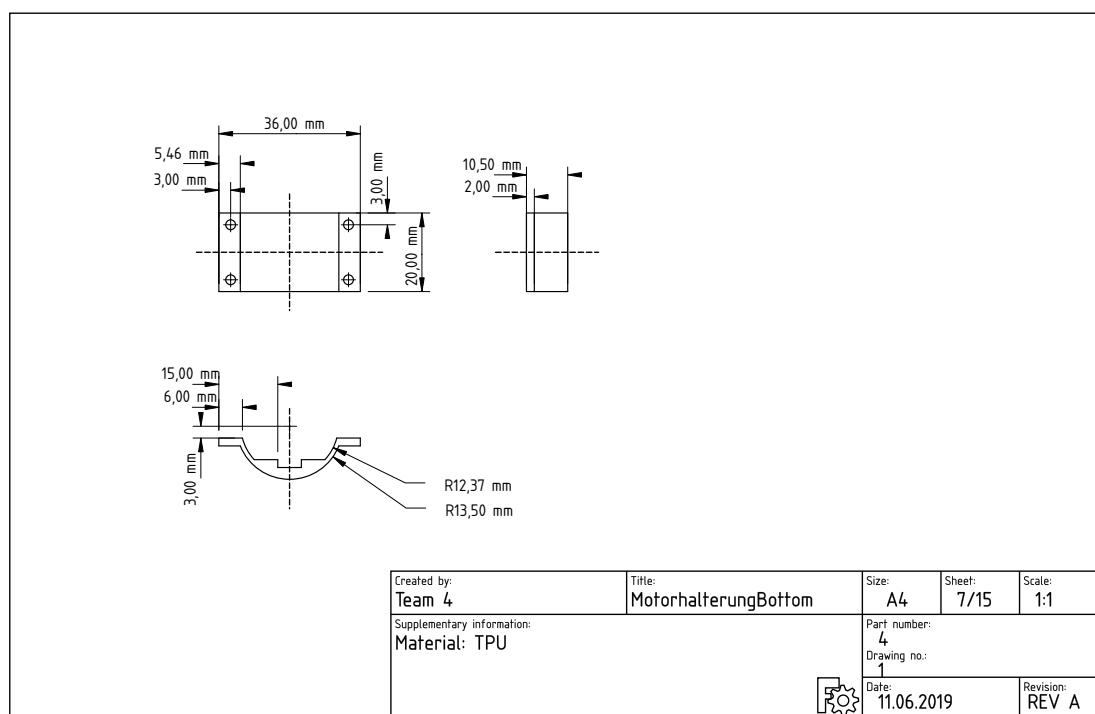


Abbildung B.7: Motorenhalterung Unten

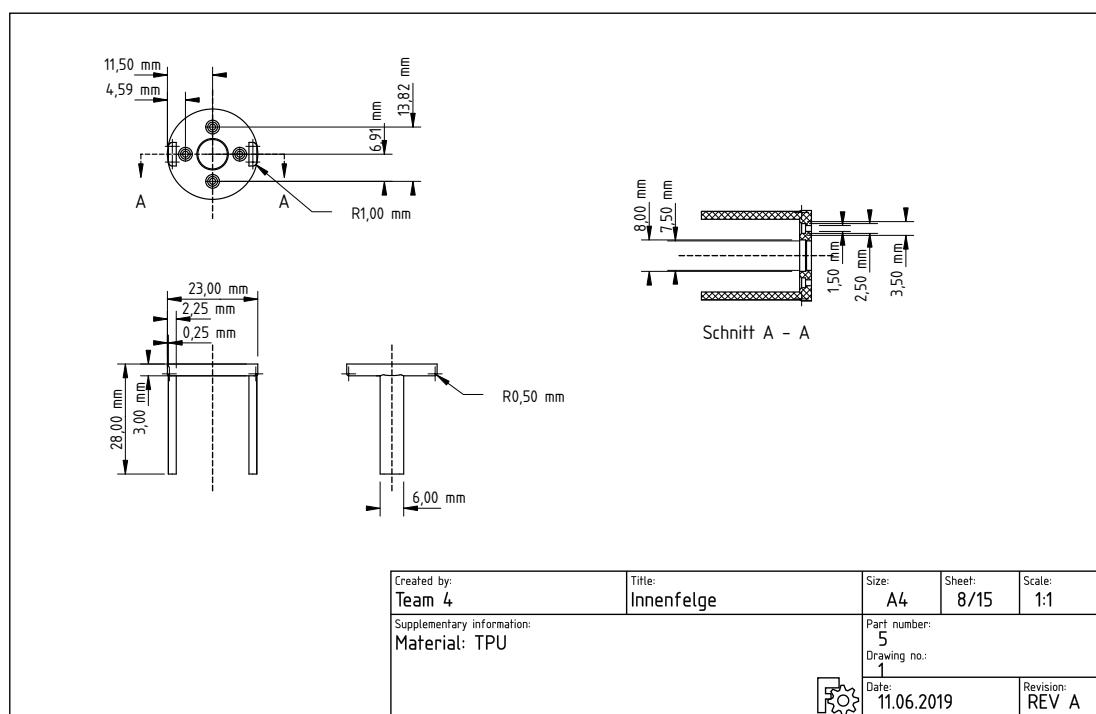


Abbildung B.8: Innenfelge

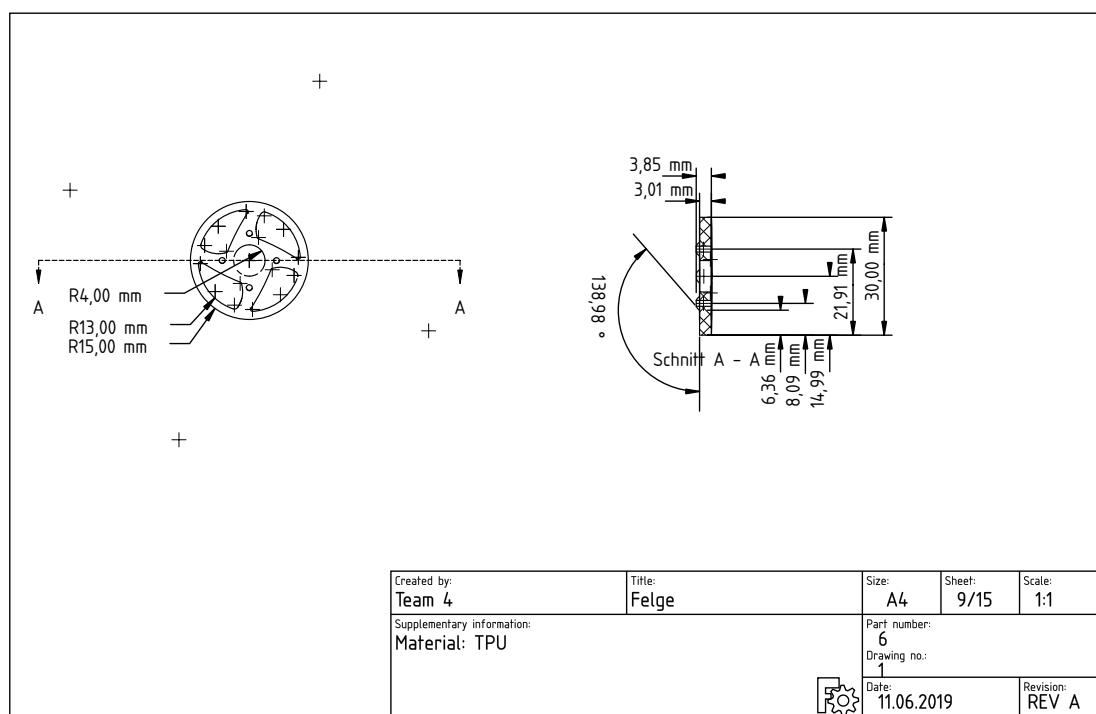


Abbildung B.9: Felge

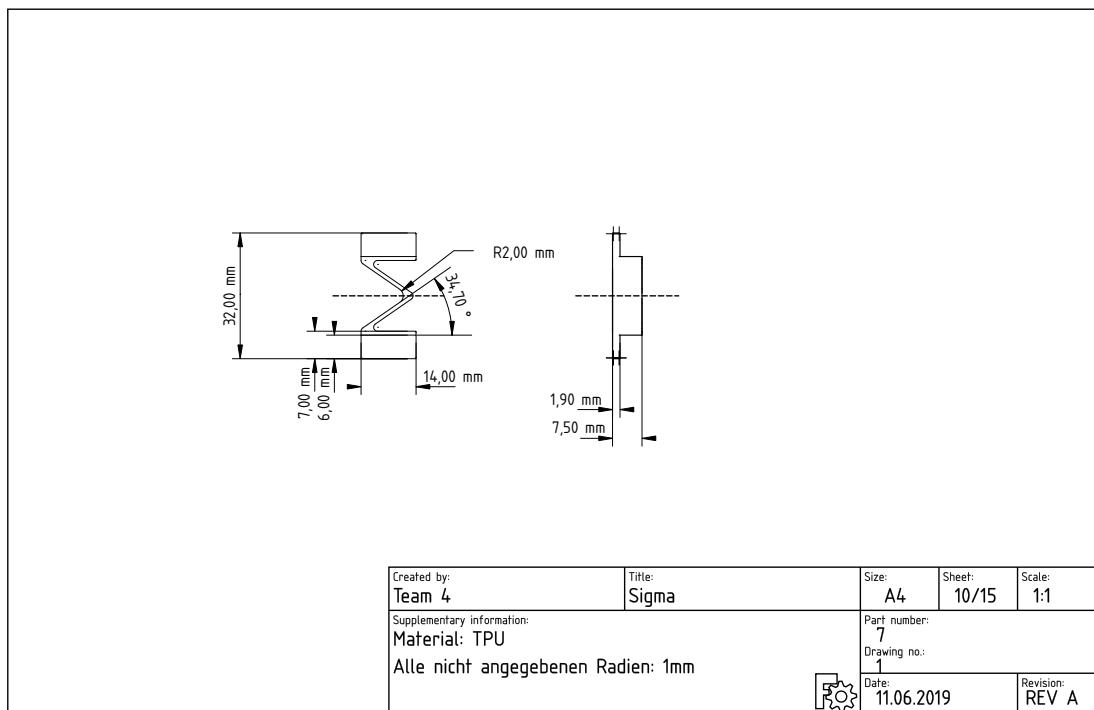


Abbildung B.10: Sigma

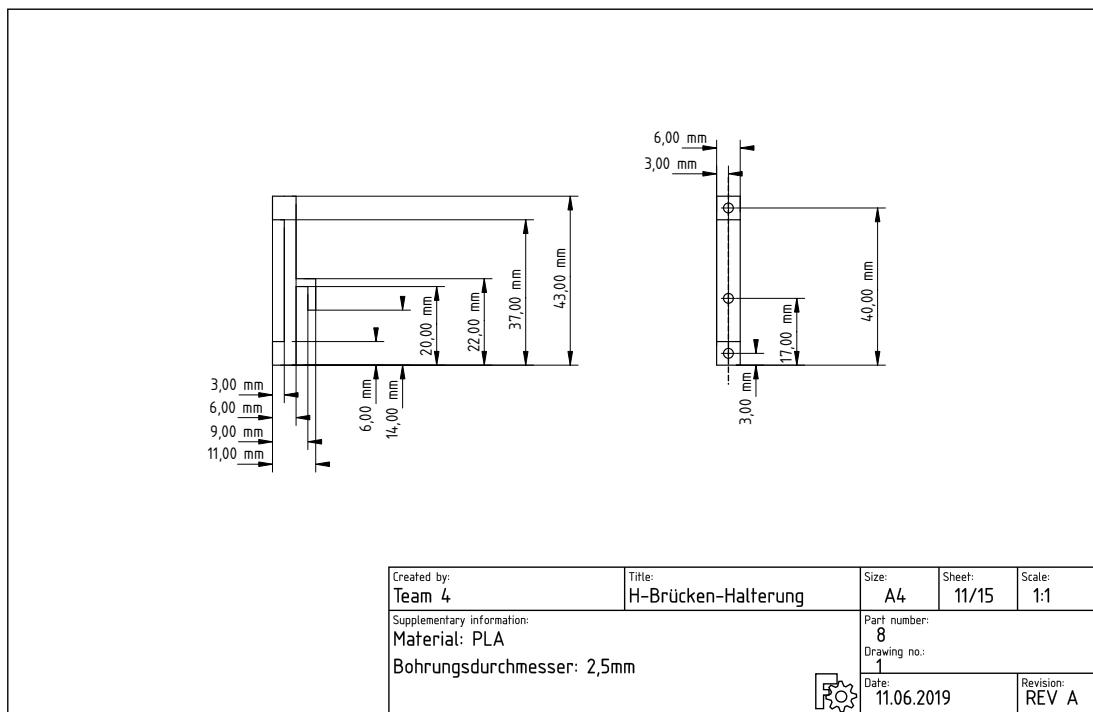


Abbildung B.11: H-Brücken-Halterung

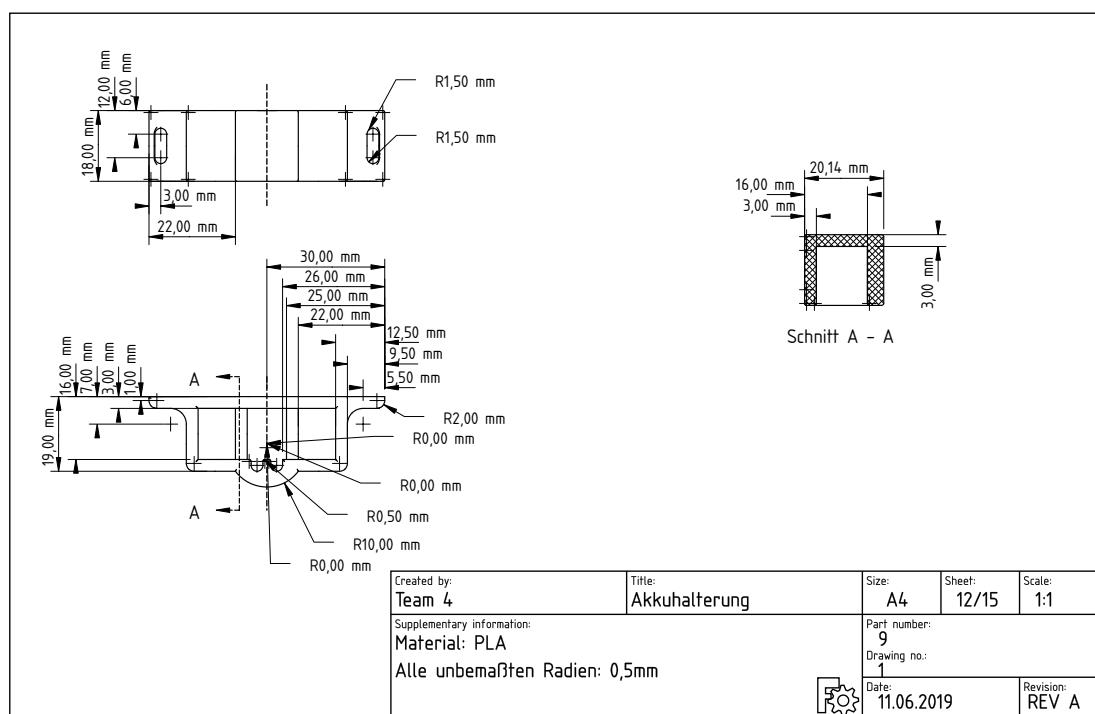


Abbildung B.12: Akkuhalterung

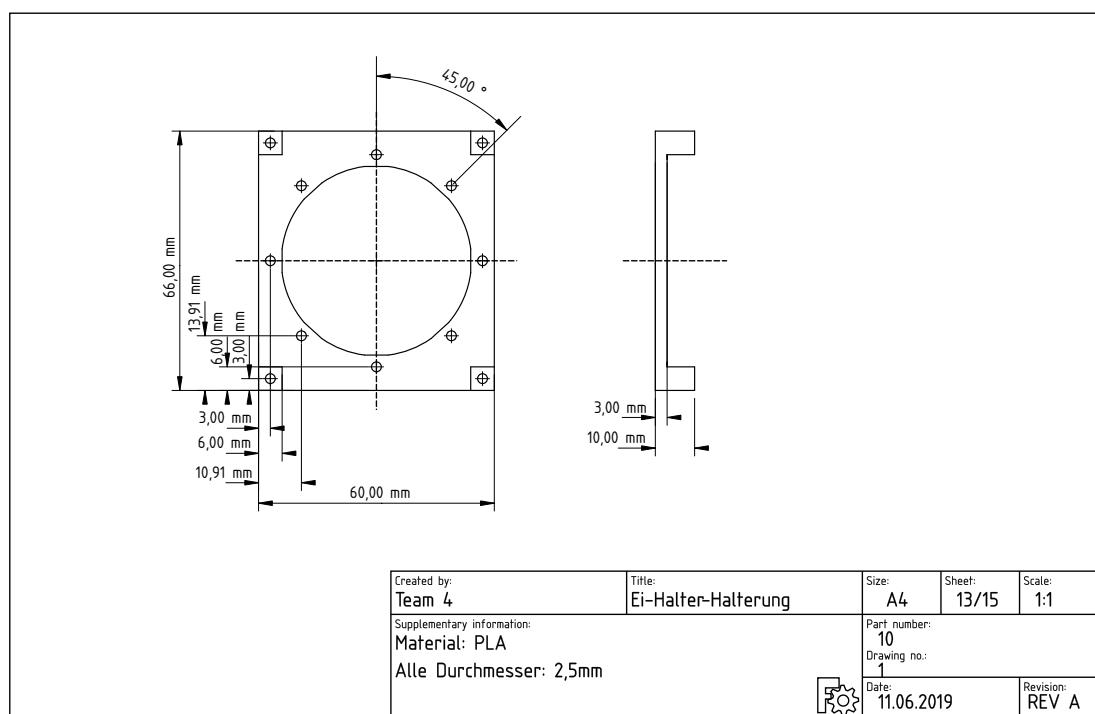


Abbildung B.13: Ei-Halterung-Halterung

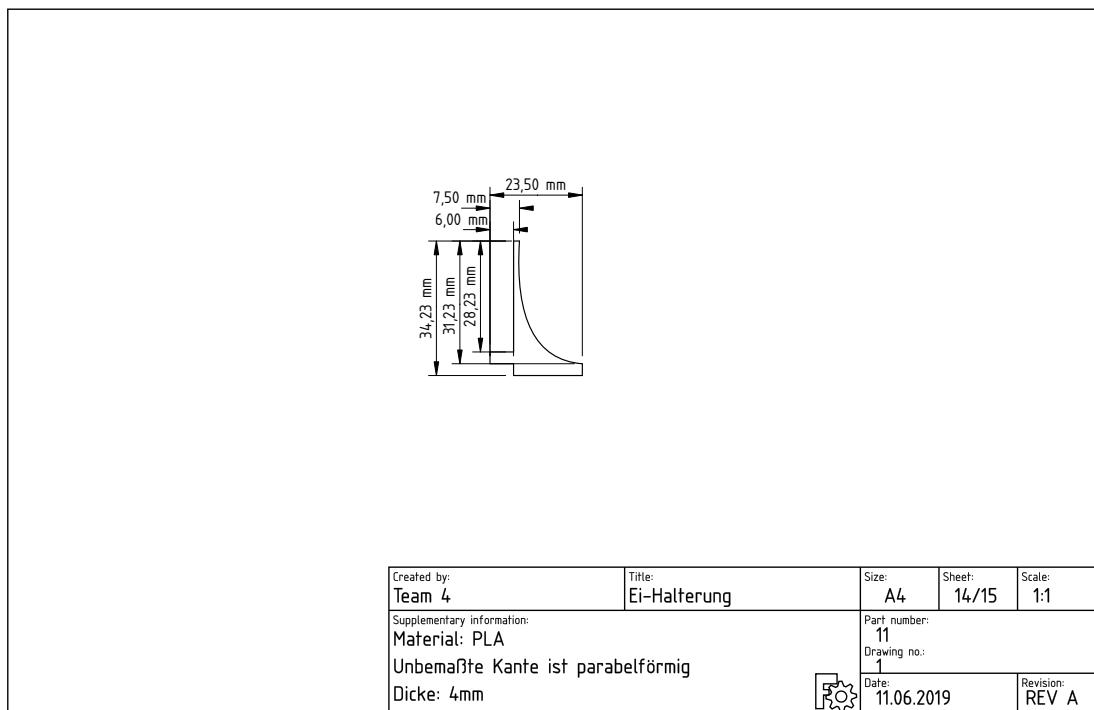


Abbildung B.14: Ei-Halterung

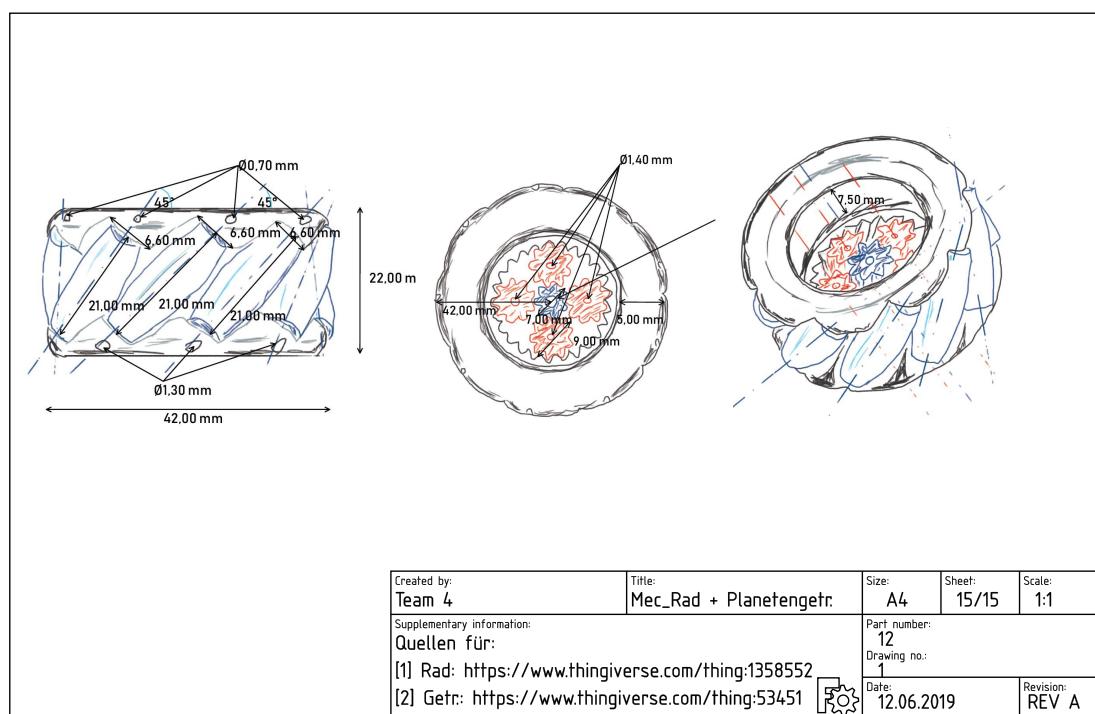


Abbildung B.15: Reifen und Planetengetriebe

Listings

A.1	communication.cpp	10
A.2	communication.h	11
A.3	movement.cpp	12
A.4	movement.h	16
A.5	util.cpp	18
A.6	util.h	20
A.7	main.cpp	20
A.8	index.html	23
A.9	code.js	26
A.10	style.css	33

Abbildungsverzeichnis

4.1	Bildschirmfoto Weboberfläche	5
B.1	Komplettansicht	35
B.2	Isometrisch und Stückliste	36
B.3	Hauptträger	37
B.4	Hauptträger Isometrisch	38
B.5	Motorenhalterhalter	39
B.6	Motorenhalterung Oben	40
B.7	Motorenhalterung Unten	41
B.8	Innenfelge	42
B.9	Felge	43
B.10	Sigma	44
B.11	H-Brücken-Halterung	45
B.12	Akkuhalterung	46
B.13	Ei-Halterung-Halterung	47
B.14	Ei-Halterung	48
B.15	Reifen und Planetengetriebe	49

Tabellenverzeichnis

4.1 Binäres Protokoll	6
---------------------------------	---