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

2 Anforderungen

2.1 Lastenheft

3 Planung

3.1 EntwÃ¼rfe

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 kÃ¶nnen, war eine Voraussetzung, dass nur Teile verbaut werden, die nicht Eigentum des Lehrstuhls sind. Aus diesem wurden die nÃ¶tigen Bauteile bei unterschiedlichen Onlineshops herausgesucht und bestellt. Hierbei fiel die Entscheidung auf Pollin, einem deutschen Elektronik HÃndler und AliExpress, einem chinesischen GroÃhÃndler. In der ursprÃ¼nglichen Planung

wurden die Kosten pro Fahrzeug auf circa 20? - 25? Ã¼berschlagen. In der finalen Bestellung beliefen sich die Kosten auf insgesamt etwa 32?.

Artikel	StÃ¼cke	?/StÃ¼ck	Quelle

3.4 Verwendete Technologien

3.4.1 Git

3.4.2 Creo

3.4.3 PlatformIO

4 Entwicklung der Schlüsselelemente

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 rrawSSockets, ohne ein eigenes FFrameüm 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.

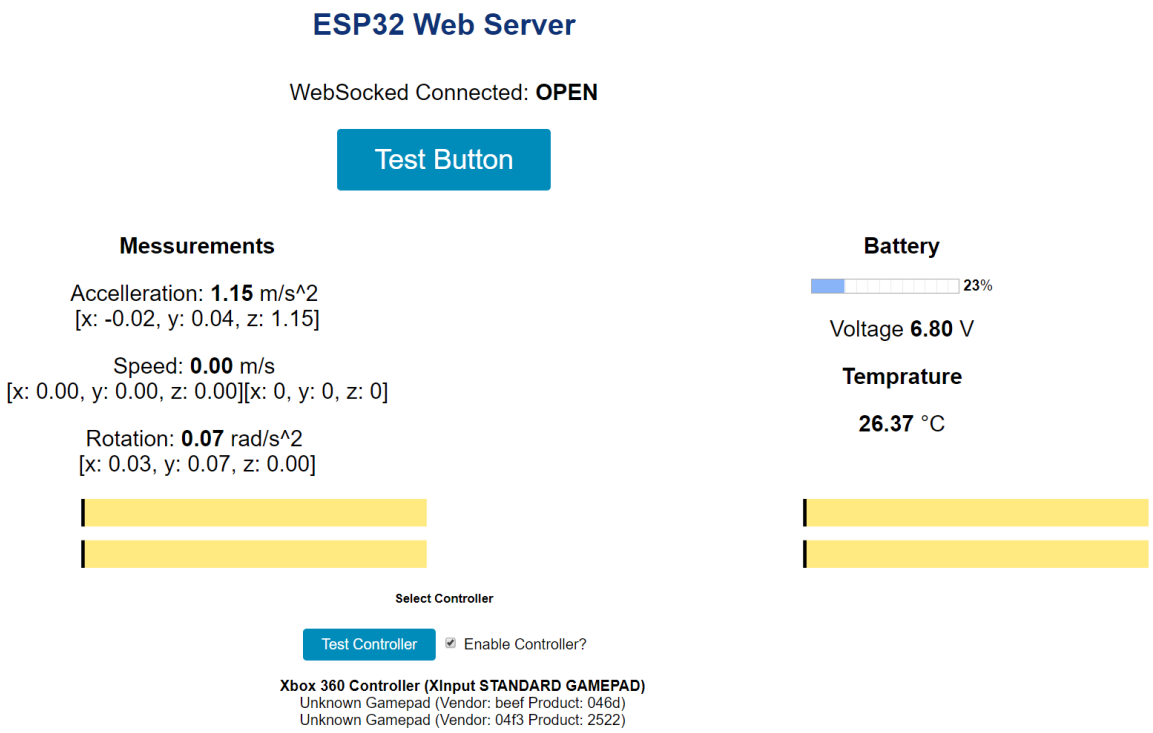

Abbildung 4.1: Bildschirmfoto Weboberfläche

In der Weboberfläche (siehe Abbildung **??**) 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**										
DRIVE	0	X-Achse L	Y-Achse L	X-Achse R	Y-Achse R							
GYRO	1	Speed-X	Speed-Y	Speed-Z	Accel-X	Accel-Y	Accel-Z	Rot-X	Rot-Y	Rot-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

6 Zusammenfassung

6.1 Evaluation

6.2 Errungene Erfahrung

Anhang

A Quellcode

A.1 ESP32

```
#include "communication.h"

AsyncWebSocketClient *ws_client = nullptr;

void Communication::onWSData(AsyncWebSocket *server, AsyncWebSocketClient *client,
    AwsEventType type, uint8_t *data, size_t len) {
    ws_client = client;

    int16_t *d16 = (int16_t *)data;
    int16_t id = d16[0];

    switch (id) {
    case OmniMessageType::DRIVE:
        onDrive(d16[1], d16[2], d16[3], d16[4]);
        break;

    default:
        Serial.printf("No such command %i\n", id);
        break;
    }
}

void Communication::onDrive(int16_t x1, int16_t y1, int16_t x2, int16_t y2) {
    // Serial.printf("Driving with speed: %i, %i and %i, %i now\n", x1, y1, x2,
    y2);
    Movement::drive(y1, x1, x2);
}

void Communication::sendCurrGyro(XYZ speed, XYZ accel, XYZ rot) {
    int16_t buff[10] = {OmniMessageType::CURR_GYRO, speed.x, speed.y, speed.z,
    accel.x, accel.y, accel.z, rot.x, rot.y, rot.z};
    ws->binaryAll((uint8_t *)buff, 20);
```

```
30 }
31
32 void Communication::sendCurrGyBatComb(int16_t sX, int16_t sY, int16_t sZ, int16_t
       aX, int16_t aY, int16_t aZ, int16_t rX, int16_t rY, int16_t rZ, int16_t bat,
       int16_t temp) {
33     int16_t buff[12] = {OmniMessageType::CURR_GY_BAT_COMB, sX, sY, sZ, aX, aY, aZ,
       rX, rY, rZ, bat, temp};
34     ws->binaryAll((uint8_t *)buff, 24);
35 }
36
37 void Communication::sendCurrMotor(int16_t vl, int16_t vr, int16_t hl, int16_t hr) {
38     int16_t buff[5] = {OmniMessageType::CURR_MOTOR, vl, vr, hl, hr};
39     // Serial.println("should send now");
40     ws->binaryAll((uint8_t *)buff, 10);
41 }
42
43 void Communication::sendCurrBattery(int16_t cell1, int16_t cell2) {
44     int16_t buff[5] = {OmniMessageType::CURR_BATTERY, cell1, cell2};
45     // Serial.println("should send now");
46     ws->binaryAll((uint8_t *)buff, 10);
47 }
48
49 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,
       1023] :: {x2} int16 [-1023, 1023] :: {y2} int16 [-1023, 1023]
```

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

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,
        PIN_MOTOR_VL_DIR2);
33  Motor Movement::MOTOR_VR = Motor(CH_MOTOR_VR, PIN_MOTOR_VR_DIR1,
        PIN_MOTOR_VR_DIR2);
34  Motor Movement::MOTOR_HL = Motor(CH_MOTOR_HL, PIN_MOTOR_HL_DIR1,
        PIN_MOTOR_HL_DIR2);
35  Motor Movement::MOTOR_HR = Motor(CH_MOTOR_HR, PIN_MOTOR_HR_DIR1,
        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
90      Serial.println("Inited PWMs");
91  }
92
93  void Movement::drive(int controlFront, int controlSide, int controlTurn) {
94      if (abs(controlFront) < CONTROLLER_LOWER_LIMIT && abs(controlSide) <
        CONTROLLER_LOWER_LIMIT && abs(controlTurn) < CONTROLLER_LOWER_LIMIT) {
95          // Serial.println("Stopping Motors");
96          MOTOR_VL.stop();
97          MOTOR_VR.stop();
98          MOTOR_HL.stop();
99          MOTOR_HR.stop();
100
101         Communication::sendCurrMotor(0, 0, 0, 0);
102         return;
103     }
104
105     int speedVL = 0;
106     int speedVR = 0;
107     int speedHL = 0;
108     int speedHR = 0;
109
110     if (abs(controlFront) > 0 && controlSide == 0 && controlTurn == 0) {
111         speedVL = controlFront;
112         speedVR = controlFront;
113         speedHL = controlFront;
114         speedHR = controlFront;
115     } else if (abs(controlSide) > 0 && controlFront == 0 && controlTurn == 0) {
116         speedVL = controlSide;
117         speedVR = -controlSide;
118         speedHL = -controlSide;
119         speedHR = controlSide;
120     } else if (controlSide == 0 && controlFront == 0 && abs(controlTurn) > 0) {
121         speedVL = controlTurn;
122         speedVR = -controlTurn;
123         speedHL = controlTurn;
124         speedHR = -controlTurn;
125     } else {
126         driveAlgorithm(controlFront, controlSide, controlTurn, &speedVL, &speedVR,
        &speedHL, &speedHR);
127     }
128
```

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

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),
       pin_dir1(pin_dir1), pin_dir2(pin_dir2){};
16     ~Motor() {}
17
18     /**
19      * speed in [-1023; 1023]
20      */
21     void setSpeed(int16_t speed);
22
23     void stop();
24 };
25
26 class Movement {
27 private:
28     /* data */
29 public:
30     static const char PIN_LED = 2;
31
32     static Motor MOTOR_VL;
33     static Motor MOTOR_VR;
34     static Motor MOTOR_HL;
35     static Motor MOTOR_HR;
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        * Alogrithm source:
        http://eprints.utem.edu.my/16543/1/Omni%20Directional%20Control%20Algorithm%20For%20Mecanum
73        */
74      static void drive(int controlFront, int controlSide, int controlTurn);
75
76  private:
77      static void driveAlgorithm(int controlFront, int controlSide, int controlTurn,
        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
```

```
 *
 *
 ************************************************************/
void cossin(float x, float *outCos, float *outSin) {
    float sin, cos;

    //always wrap input angle to -PI..PI
    if (x < -3.14159265)
        x += 6.28318531;
    else if (x > 3.14159265)
        x -= 6.28318531;

    //compute sine
    if (x < 0) {
        sin = 1.27323954 * x + .405284735 * x * x;

        if (sin < 0)
            sin = .225 * (sin * -sin - sin) + sin;
        else
            sin = .225 * (sin * sin - sin) + sin;
    } else {
        sin = 1.27323954 * x - 0.405284735 * x * x;

        if (sin < 0)
            sin = .225 * (sin * -sin - sin) + sin;
        else
            sin = .225 * (sin * sin - sin) + sin;
    }

    //compute cosine: sin(x + PI/2) = cos(x)
    x += 1.57079632;
    if (x > 3.14159265)
        x -= 6.28318531;

    if (x < 0) {
        cos = 1.27323954 * x + 0.405284735 * x * x;

        if (cos < 0)
            cos = .225 * (cos * -cos - cos) + cos;
        else
            cos = .225 * (cos * cos - cos) + cos;
    } else {
        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  /**
63   * Branchless signum function
64   */
65  int sgn(int val) {
66      return (0 < val) - (val < 0);
67  }
```

Listing A.5: util.cpp

```
1  #pragma once
2
3  void cossin(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
       type, void *arg, uint8_t *data, size_t len) {
24
25     switch (type) {
26     case WS_EVT_CONNECT:
27         Serial.printf("Client connected from %s\n",
       client->remoteIP().toString().c_str());
28         break;
29
30     case WS_EVT_DATA:
31         Communication::onWSData(server, client, type, data, len);
32         break;
33
34     default:
35         break;
36     }
37 }
38
39 void setup() {
40     // enableCore1WDT();
41     Serial.begin(115200);
42
43     Wire.begin();
44
45     mpu.initialize();
46
47     // mpu.CalibrateAccel_MPU6500(6);
48     // mpu.CalibrateGyro(6);
49
50     // mpu.PrintActiveOffsets_MPU6500();
51     mpu.setXGyroOffset(96);
52     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 succesfully");
76      }
77
78      // Route for root / web page
79      server.on("/", HTTP_GET, [](AsyncWebServerRequest *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, [](AsyncWebServerRequest *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, [](AsyncWebServerRequest *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: %6.2fÂ°/s, gy:
        %6.2fÂ°/s, gz: %6.2fÂ°/s, temp: %6.2fÂ°C\r", x / 16384.0, y / 16384.0, z /
        16384.0, gx / 250.0, gy / 250.0, gz / 250.0, temp / 340.0 + 36.53);
121     // power = (v / 4095 * 3.1 + .1) * 3;
122     Communication::sendCurrGyBatComb(0, 0, 0, x, y, z, gx, gy, gz, v, temp);
123     delay(1000);
124 }
```

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>Messurements</h2>
21        <p>Accelleration: <strong id="valAccelTotal">0</strong> m/s^2
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"></span>[x: 0, y: 0, z: 0]</p>
27
28        <p>Rotation: <strong id="valRotTotal">0</strong> rad/s^2
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> Â°C</p>
42      </div>
43    </div>
44
45
46    <div>
```

```
47      <svg viewBox="0 0 205 10" preserveAspectRatio="none" style="width: 100%;
        height: 100%;">
48        <path id="pathVL" d="M50,2 L100,2 0,2 50,2" stroke="#FFEA82"
        stroke-width="4" fill-opacity="0"
49          style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"></path>
50        <path id="pathHL" d="M50,8 L100,8 0,8 50,8" stroke="#FFEA82"
        stroke-width="4" fill-opacity="0"
51          style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"></path>
52
53        <path id="pathVR" d="M155,2 L205,2 105,2 155,2" stroke="#FFEA82"
        stroke-width="4" fill-opacity="0"
54          style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"></path>
55        <path id="pathHR" d="M155,8 L205,8 105,8 155,8" stroke="#FFEA82"
        stroke-width="4" fill-opacity="0"
56          style="stroke-dasharray: 100, 100; stroke-dashoffset: 0;"></path>
57
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
        Controller</button>
68      <input type="checkbox" id="controllerEnabled" checked>
69      <label for="controllerEnabled">Enable Controller?</label>
70      <ul id="controllerList">
71        <li onclick="onControllerSelected(0)">Test</li>
72
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

```
let ws = new WebSocket("ws://192.168.4.1/ws");
// let ws = new WebSocket("ws://demos.kaazing.com/echo");
ws.binaryType = 'arraybuffer';

let barVL = new ProgressBar.Path('#pathVL', { easing: 'easeInOut', duration: 140,
    });
let barVR = new ProgressBar.Path('#pathVR', { easing: 'easeInOut', duration: 140,
    });
let barHL = new ProgressBar.Path('#pathHL', { easing: 'easeInOut', duration: 140,
    });
let barHR = new ProgressBar.Path('#pathHR', { easing: 'easeInOut', duration: 140,
    });
barVL.set(0.1)
barVR.set(0.1)
barHL.set(0.1)
barHR.set(0.1)

let selectedControllerIndex = -1;
window.onload = function () {
  setInterval(controllerFunc, 100);
  updateControllerList();
};

//#region >>>> WebSocket
ws.onopen = (event) => {
  document.getElementById("wsState").innerHTML = "OPEN";
};

ws.onerror = (event) => {
  document.getElementById("wsState").innerHTML = "ERROR";
};

ws.onclose = (event) => {
  document.getElementById("wsState").innerHTML = "CLOSED";
};

ws.onmessage = function (event) {
  // 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] +
    a[1] * a[1] + a[2] * a[2]).toFixed(2);
45      document.getElementById("valSpeedTotal").innerHTML = Math.sqrt(a[0] * s[0] +
    s[1] * s[1] + s[2] * s[2]).toFixed(2);
46      document.getElementById("valRotTotal").innerHTML = Math.sqrt(r[0] * r[0] +
    r[1] * r[1] + r[2] * r[2]).toFixed(2);
47
48      document.getElementById("valAccel").innerHTML = "[x: " + a[0] + ", y: " +
    a[1] + ", z: " + a[2] + "]";
49      document.getElementById("valSpeed").innerHTML = "[x: " + s[0] + ", y: " +
    s[1] + ", z: " + s[2] + "]";
50      document.getElementById("valRot").innerHTML = "[x: " + r[0] + ", y: " + r[1]
    + ", z: " + r[2] + "]";
51      break;
52
53    case 2: //battery
54      let c1 = (dv.getInt16(2, true) / 4095 * 3.1 + .1) * 3;
55
56      document.getElementById("valVolt").innerHTML = c1.toFixed(2);
57      break;
58    case 3: //motors
59      const UPPER_LIMIT = 300;
60
61      let vl = dv.getInt16(2, true) / (1023 * 4);
62      let vr = dv.getInt16(4, true) / (1023 * 4);
63      let hl = dv.getInt16(6, true) / (1023 * 4);
64      let hr = dv.getInt16(8, true) / (1023 * 4);
65
66      barVL.animate(vl);
67      barVR.animate(vr);
68      barHL.animate(hl);
69      barHR.animate(hr);
70      break;
71
```

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

```
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
117   updateControllerList();
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>" +
      element.id + "</strong></li>";
135     } else {
136       inner += "<li onclick=\"onControllerSelected(" + i + ")\">" + element.id +
      "</li>";
137     }
138   }
139
140   document.getElementById("controllerList").innerHTML = inner;
141 }
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
      || pads[selectedControllerIndex] === undefined)
157     return;
158   x1 = Math.floor(pads[selectedControllerIndex].axes[0] * 1023);
159   y1 = -Math.floor(pads[selectedControllerIndex].axes[1] * 1023);
160   x2 = Math.floor(pads[selectedControllerIndex].axes[2] * 1023);
161   y2 = -Math.floor(pads[selectedControllerIndex].axes[3] * 1023);
162
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 ||
      lastY2 != 0) {
223     console.log(dirX * 1023 + ", " + dirY * 1023 + " : " + rotX * 1023 + ", " + 0);
224     if (ws.readyState == WebSocket.OPEN) {
225       let buf = new Int16Array([0, dirX * 1023, dirY * 1023, rotX * 1023, 0])
226       ws.send(buf);
227     }
228     lastX1 = dirX * 1023;
229     lastY1 = dirY * 1023;
230     lastX2 = rotX * 1023;
231     lastY2 = 0;
```

```
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 ||
        lastY2 != 0) {
258     console.log(dirX * 1023 + ", " + dirY * 1023 + " : " + rotX * 1023 + ", " + 0);
259     if (ws.readyState == WebSocket.OPEN) {
260       let buf = new Int16Array([0, dirX * 1023, dirY * 1023, rotX * 1023, 0])
261       ws.send(buf);
262     }
263     lastX1 = dirX * 1023;
264     lastY1 = dirY * 1023;
265     lastX2 = rotX * 1023;
266     lastY2 = 0;
267   }
268 });
269 //#endregion
270
271
272
273 function sendSpeed() {
```

```
  console.log("speed");
  let arr = new Int16Array([2, Math.random() * 4048, Math.random() * 4048]);
  let arr2 = new Int16Array([1,
    Math.random() * 1023, Math.random() * 1023, Math.random() * 1023,
    Math.random() * 1023, Math.random() * 1023, Math.random() * 1023,
    Math.random() * 1023, Math.random() * 1023, Math.random() * 1023]);
  ws.send(arr);
  ws.send(arr2);
}
```

Listing A.9: code.js

```
html {
  font-family: Helvetica;
  display: inline-block;
  margin: 0px auto;
  text-align: center;
}
h1{
  color: #0F3376;
  padding: 2vh;
}
p{
  font-size: 1.5rem;
}
.button {
  display: inline-block;
  background-color: #008CBA;
  border: none;
  border-radius: 4px;
  color: white;
  padding: 16px 40px;
  text-decoration: none;
  font-size: 30px;
  margin: 2px;
  cursor: pointer;
}

.buttonsmall {
  display: inline-block;
  background-color: #008CBA;
  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

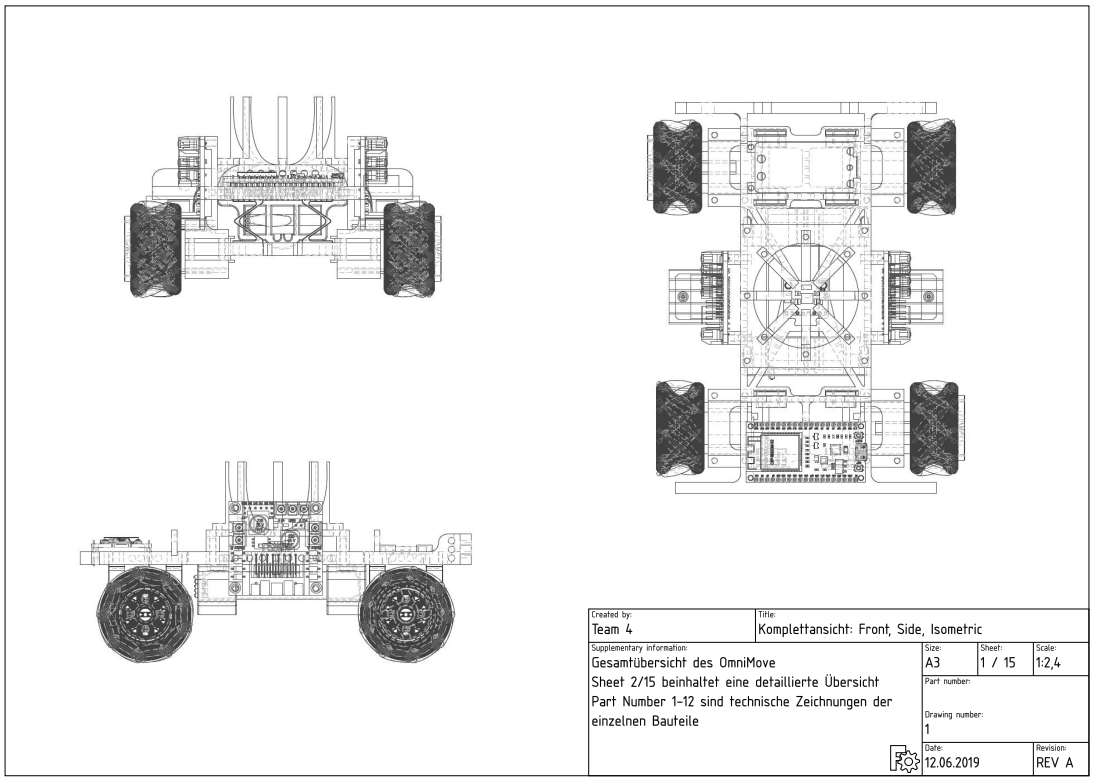

Abbildung B.1: Komplettansicht

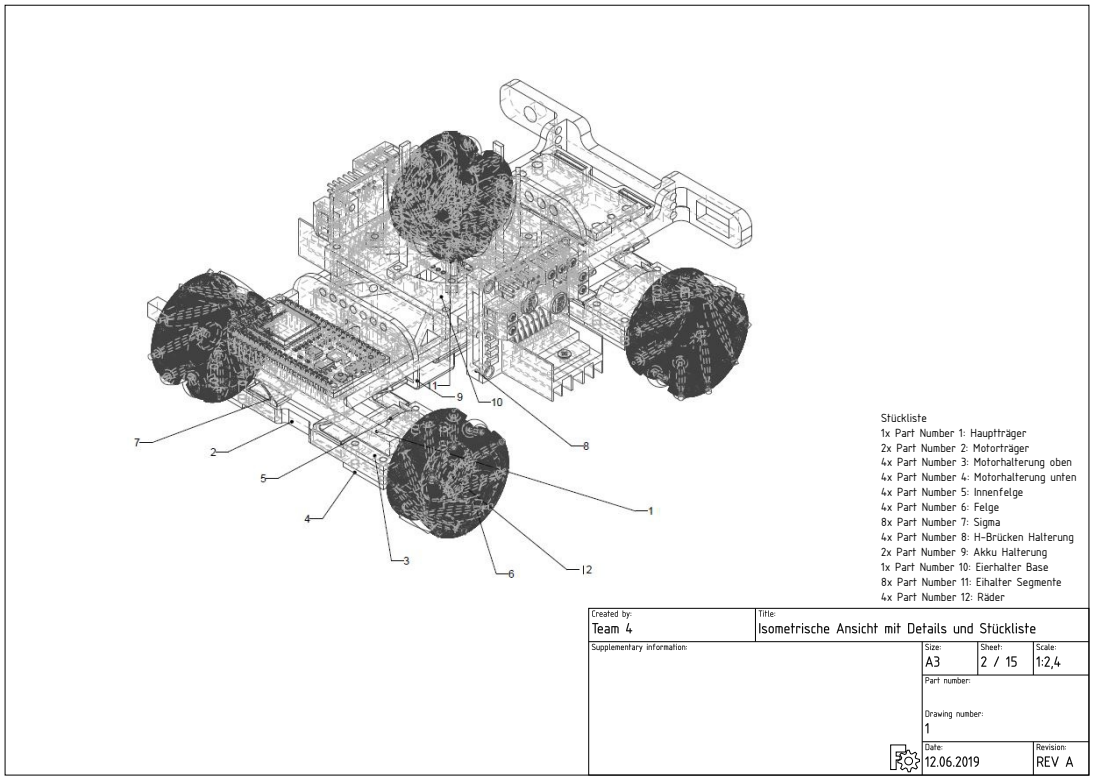

Abbildung B.2: Isometrisch und Stückliste

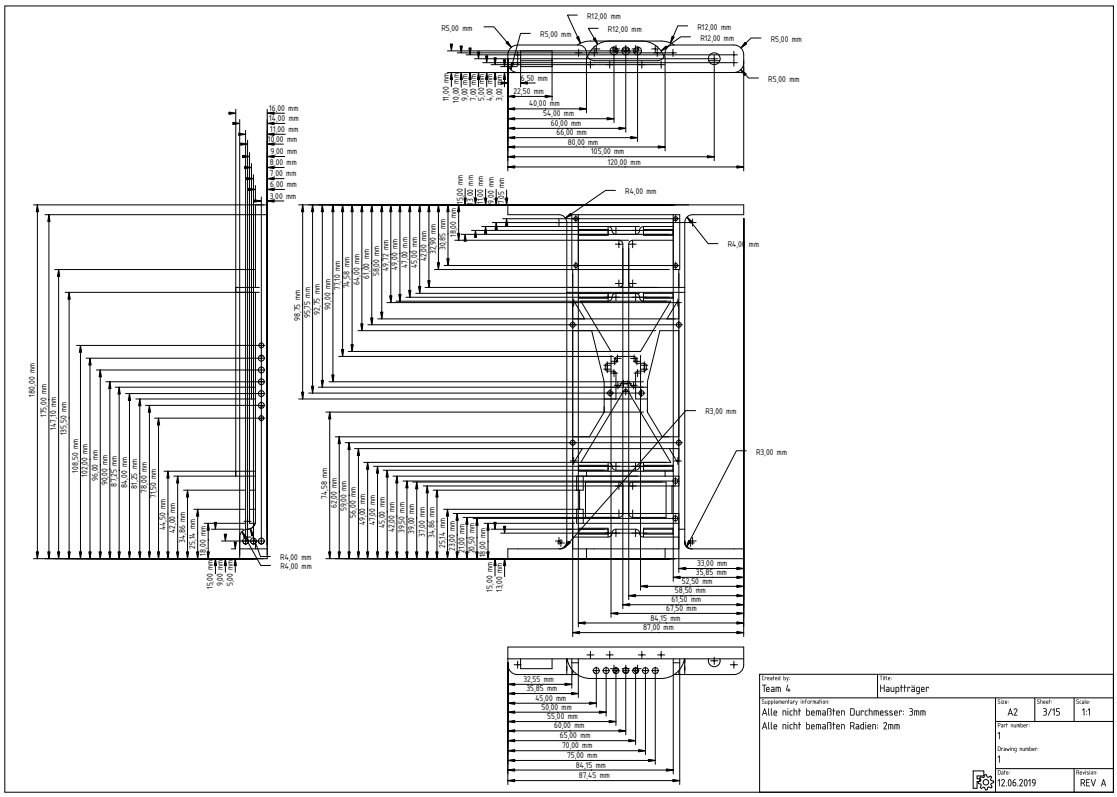

Abbildung B.3: Hauptträger

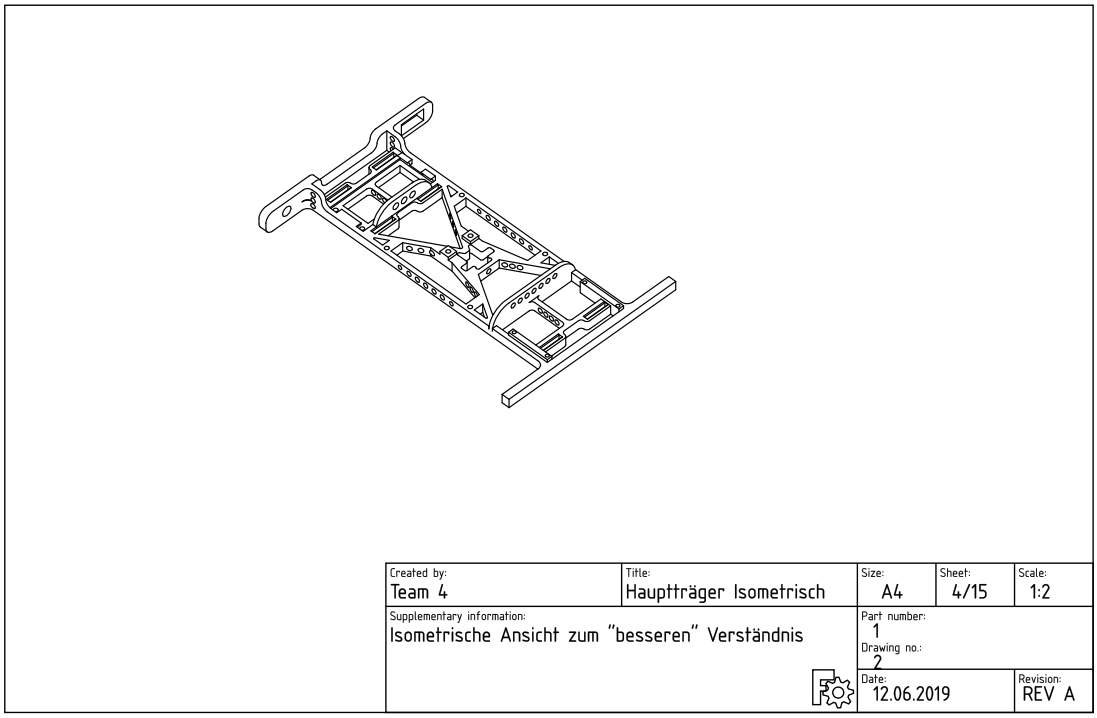

Abbildung B.4: Hauptträger Isometrisch

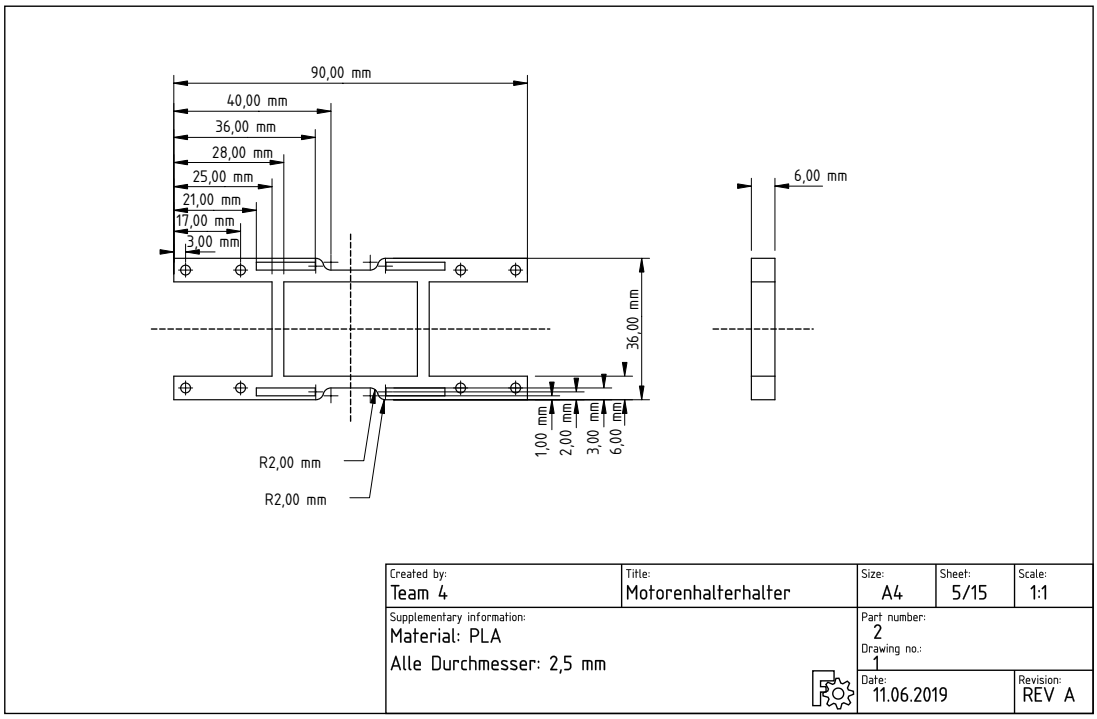

Abbildung B.5: Motorenhalterhalter

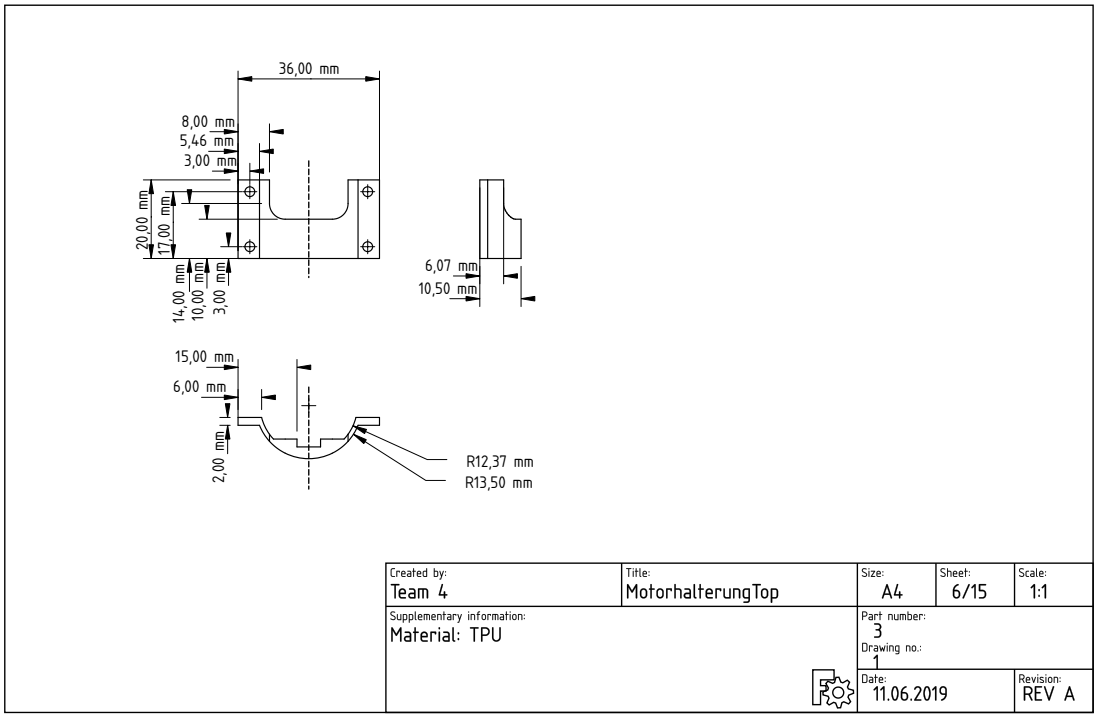

Abbildung B.6: Motorenhalterung Oben

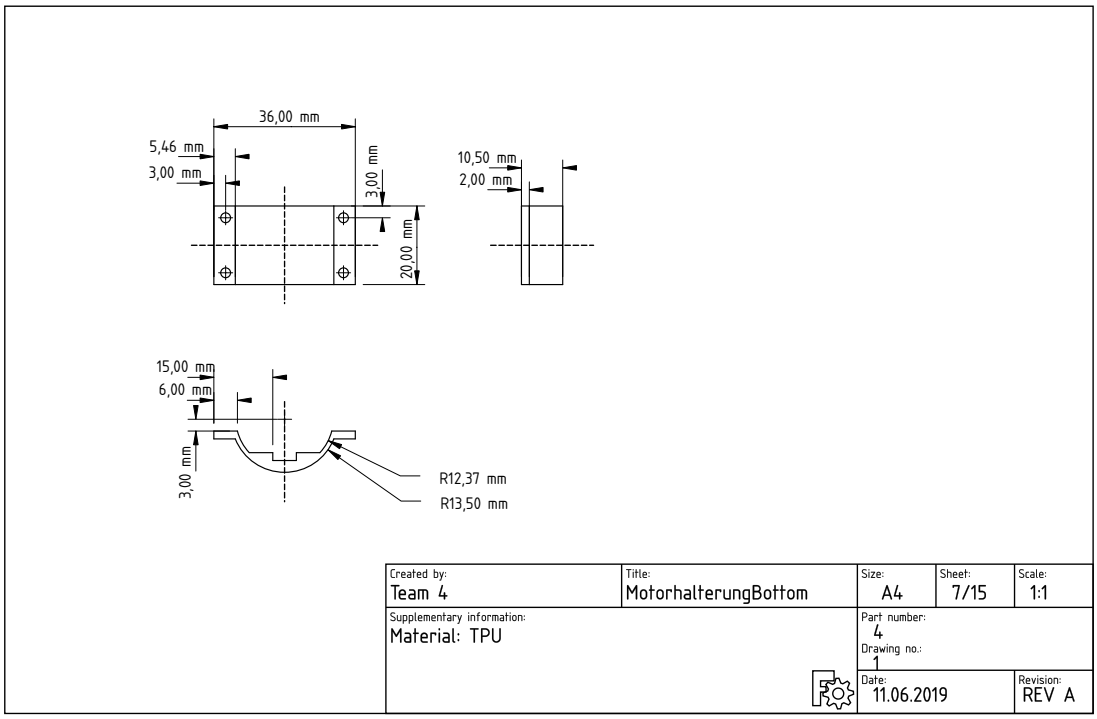

Abbildung B.7: Motorenhalterung Unten

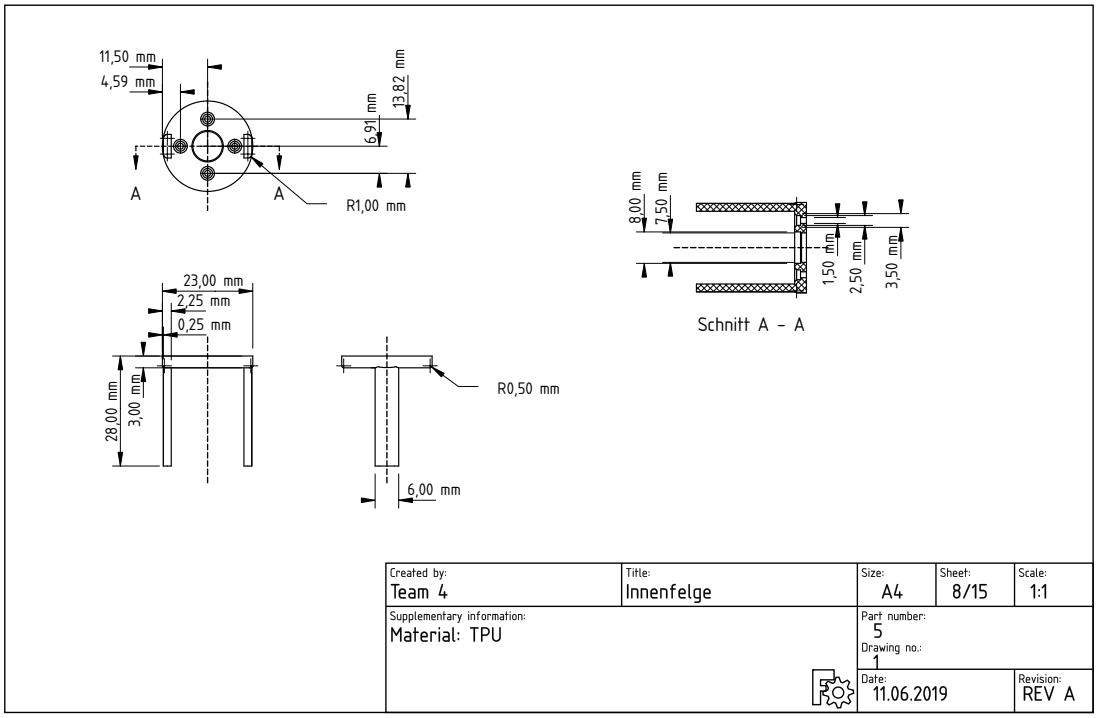

Abbildung B.8: Innenfelge

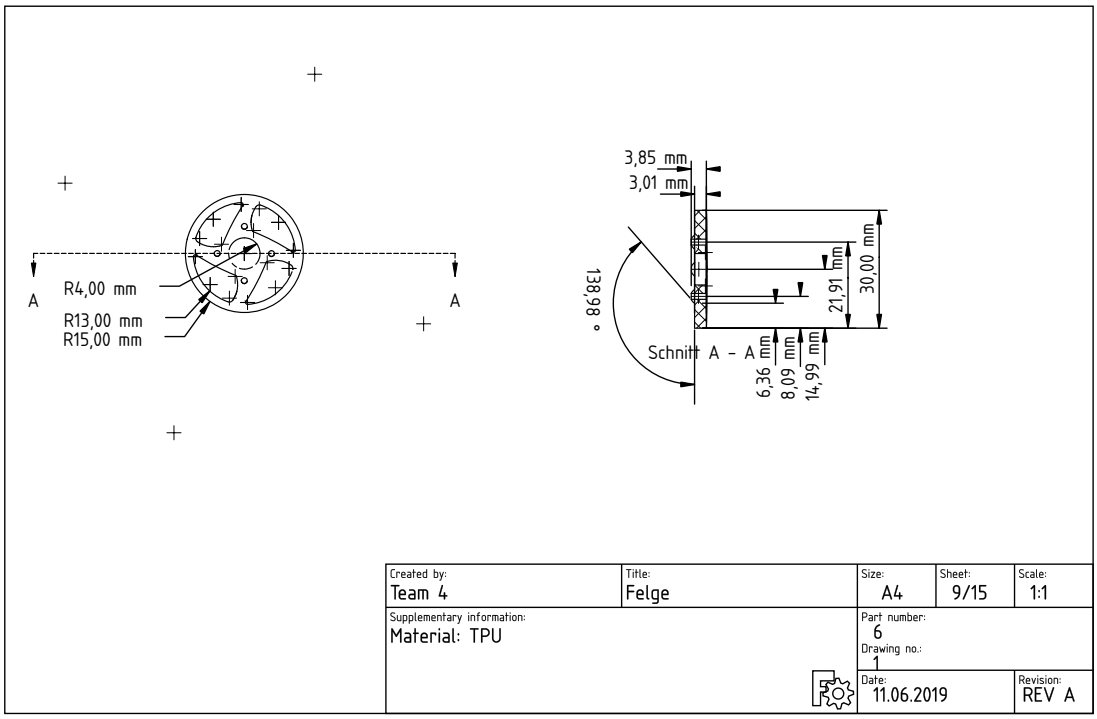

Abbildung B.9: Felge

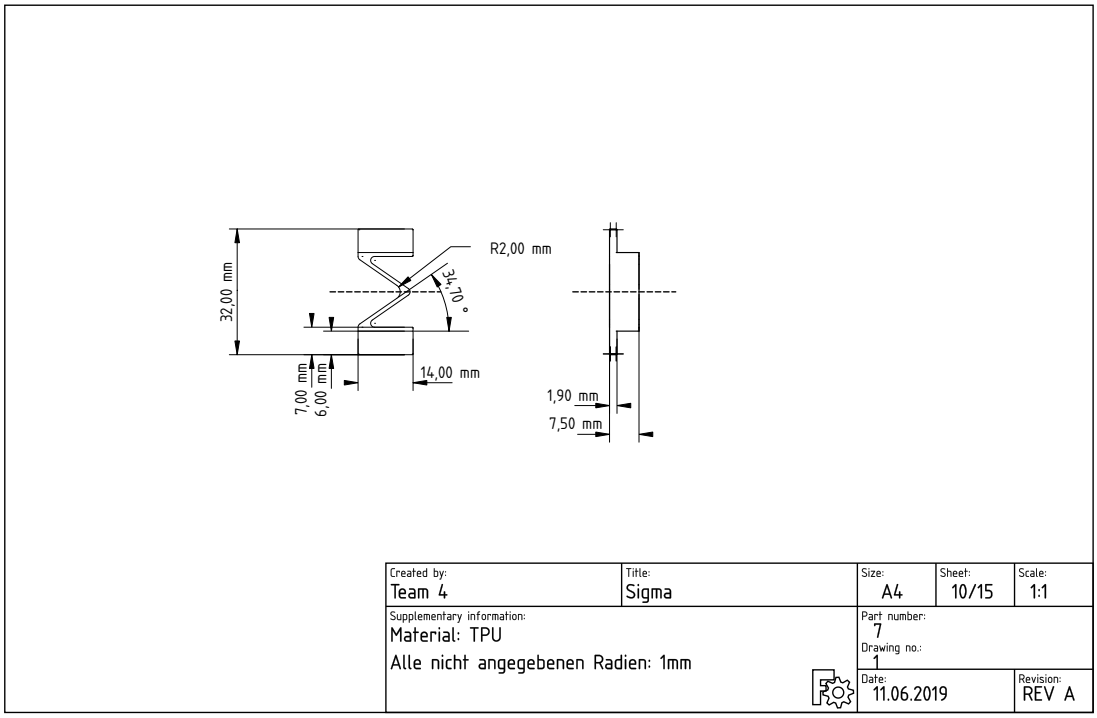

Abbildung B.10: Sigma

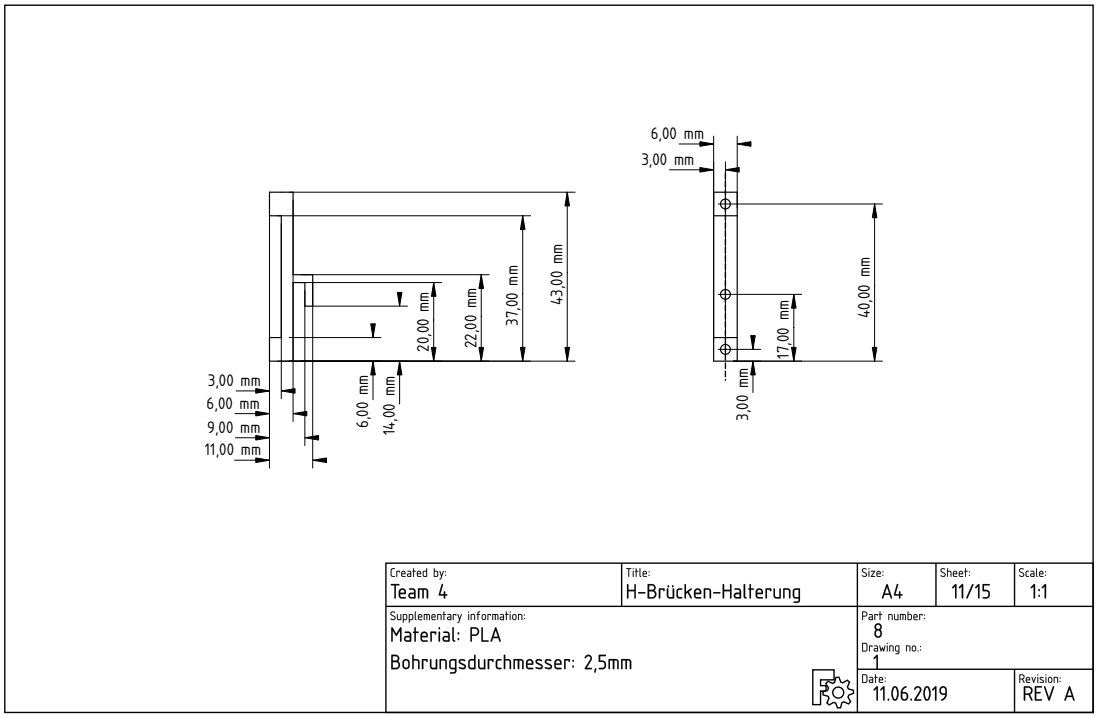

Abbildung B.11: H-Brücken-Halterung

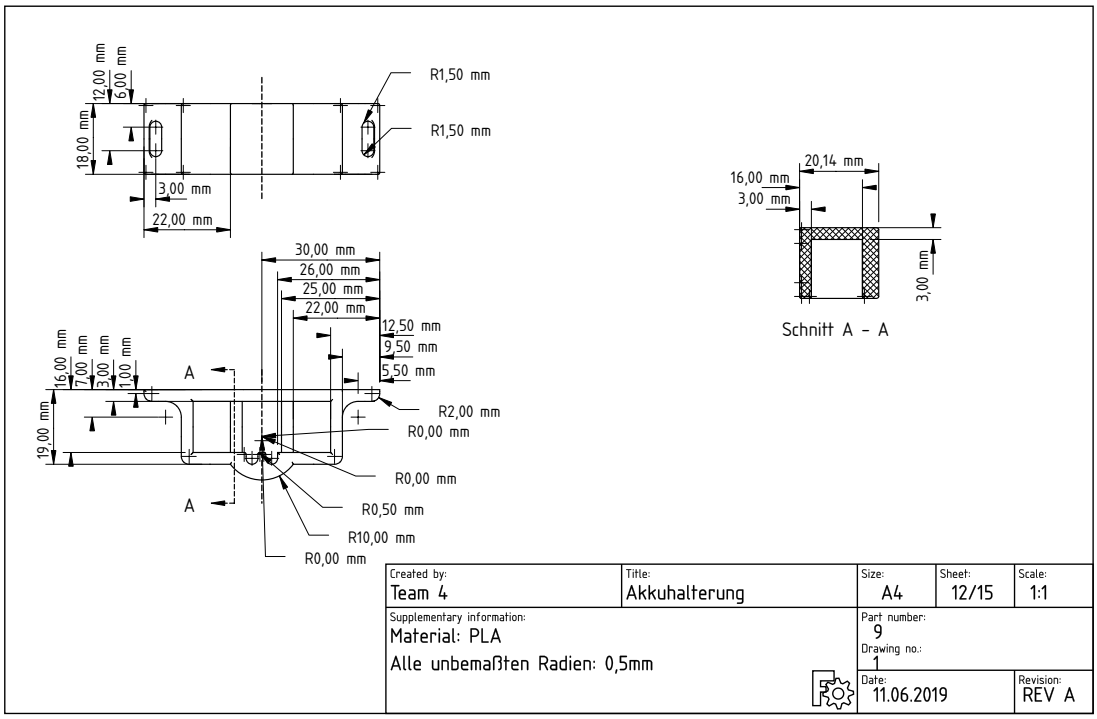

Abbildung B.12: Akkuhalterung

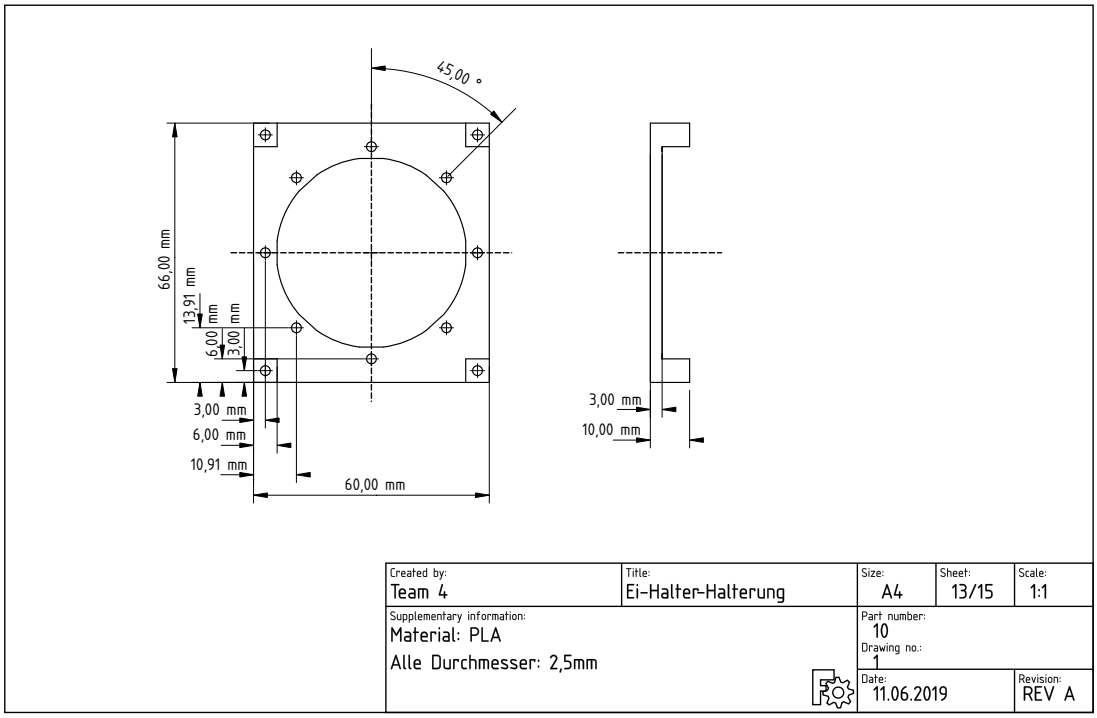

Abbildung B.13: Ei-Halterung-Halterung

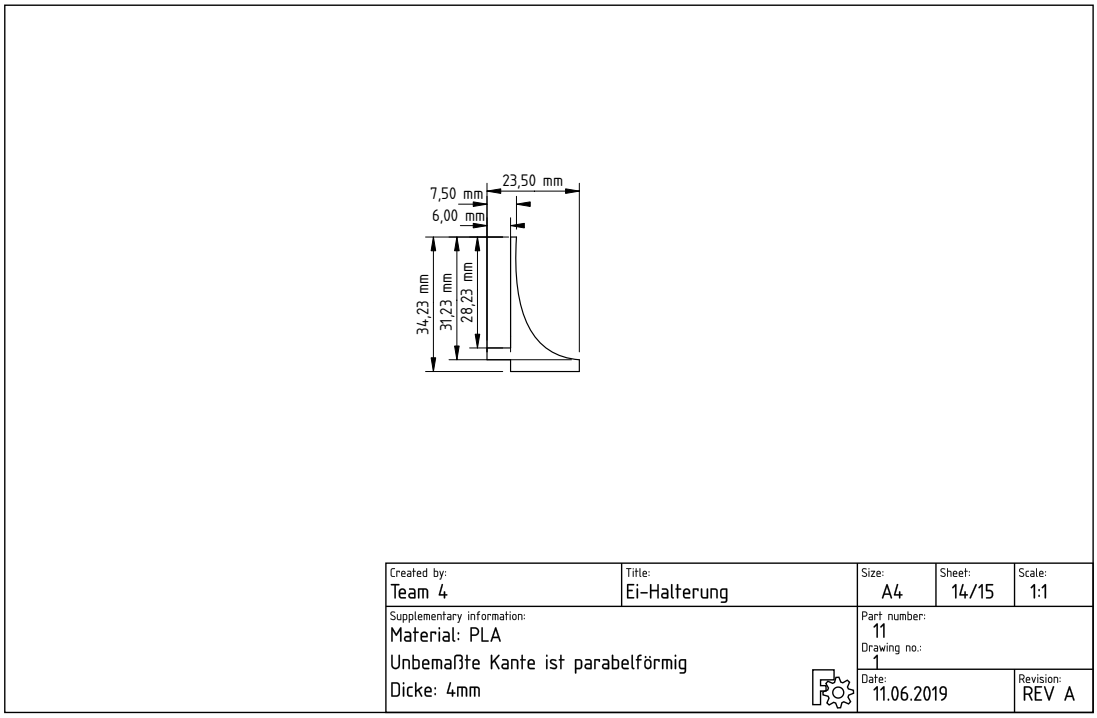

Abbildung B.14: Ei-Halterung

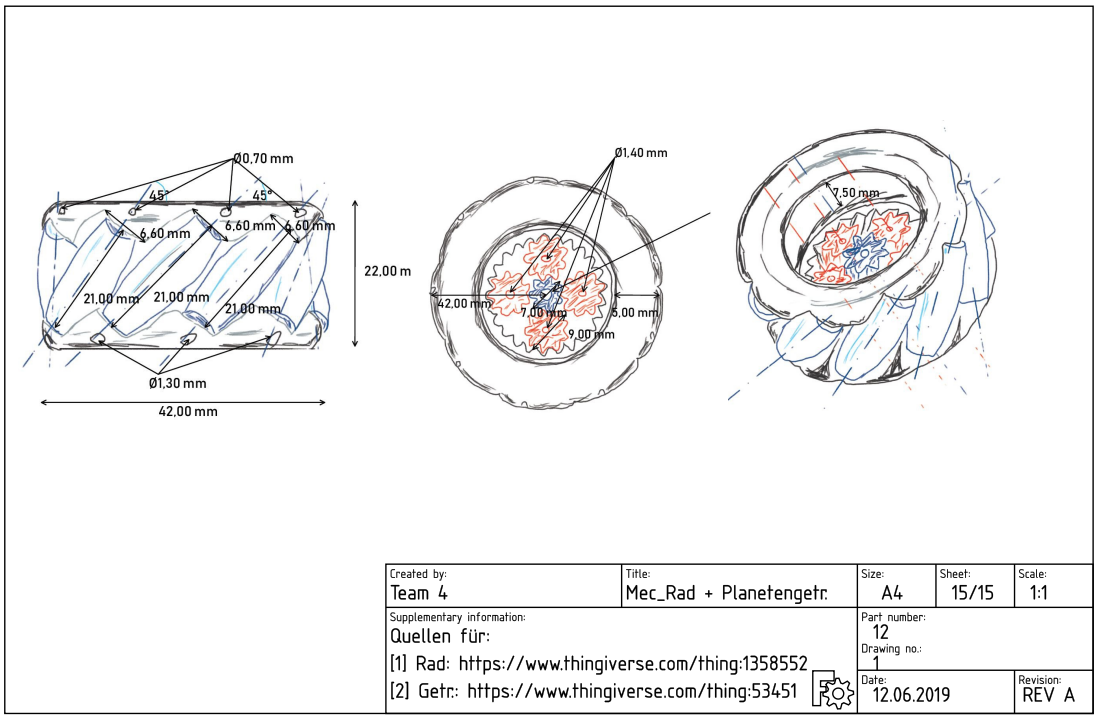

Abbildung B.15: Reifen und Planetengetriebe

Listings

Abbildungsverzeichnis

Tabellenverzeichnis