

## Inhalt

1. Projektübersicht.....	3
1.1. Motivation .....	3
1.2. Geplante Umsetzung.....	3
2. Organisation .....	4
2.1. Zeitmanagement.....	4
2.2. Versionsmanagement.....	4
2.3. Meetings.....	4
3. Entwicklungswerkzeuge .....	5
3.1. Hardware.....	5
3.2. Software .....	5
4. Entwicklung .....	6
4.1. Roboter und Software.....	6
4.2. Firmware .....	8
4.3. Apps .....	10
4.4. Antrainieren des neuronalen Netzes .....	11
5. Ergebnisse .....	11
6. Diskussion und Ausblick.....	12
6.1. Gelöste und offene Probleme.....	12
6.2. Erweiterungsmöglichkeiten.....	13
7. Fazit.....	13

# 1. Projektübersicht

## 1.1. Motivation

In erster Linie besteht unser Antrieb für den Bau eines autonom fahrenden Roboters darin, unser in den letzten 8 Monaten angeeignetes Wissen im Zusammenhang mit der Programmiersprache Python in der Praxis anzuwenden und umzusetzen. Auch das Erlernen neuronaler Netze, was zum aktuellen Trend der künstlichen Intelligenz zählt, wird eine große und spannende Herausforderung. Ein weiteres Highlight wird das Erlernen neuer Programmiersprachen, sowie das Zusammenarbeiten in Teams und das Kombinieren von unterschiedlichem Vorwissen. Nicht zuletzt ist die praktische Arbeit eine willkommene Abwechslung.

## 1.2. Geplante Umsetzung

Im Vorfeld wurden die Einzelteile des Gehäuses mit einem 3D-Drucker hergestellt und sowohl Stützgewebe entfernt als auch Kanten entgratet.

Das Projekt startete am 29.04.2022 mit ersten Informationen und Materialausgabe.

Zu Beginn wird das Fahrzeug nach offiziellem Plan von OpenBot zusammengebaut, sowie die Elektronik integriert.

Im Anschluss wird der Arduino-Nano mit dem vorhandenen Programmcode von OpenBot bespielt und gegebenenfalls eine Fehleranalyse und -behebung durchgeführt.

Wenn das Grundkonzept funktioniert und somit manuelles Fahren möglich ist, werden im nächsten Schritt die verbauten Sensoren wie Ultraschallsensor und Drehzahlmesser auf Funktion geprüft und eventuell kalibriert. Die letzte Aufgabe ist, das neuronale Netz anzutrainieren.

## 2. Organisation

### 2.1. Zeitmanagement

Um sicherzustellen, zum Stichtag 03.06.2022 einen funktionierenden, autonom fahrenden OpenBot in Händen zu halten, weisen wir folgenden Meilensteinen eine Deadline zu:

Meilenstein	Deadline	Erreicht am
1. Zusammenbau des OpenBot mit Testlauf	2022-05-06	2022-05-12
2. Installation der Firmware auf dem Arduino-Nano	2022-05-12	2022-05-12
3. Installation und Ausführen der OpenBot-App auf dem Smartphone	2022-05-12	2022-05-12
4. Programmerstellung für eigenhändige Steuerung	2022-05-13	2022-05-12
5. Anlernen der Bahn	2022-05-19	2022-05-18
6. Eigenständiges Fahren auf der Bahn	2022-05-20	2022-05-20
7. Anlernen zu umfahrender Gegenstände	optional	
8. Eigenständiges Umfahren von Gegenständen	optional	

#### Verzögerungen:

Zu 1: Der Zusammenbau des Roboters verzögerte sich, da am ersten Tag 3 der 4 Teammitglieder abwesend waren und es Platzprobleme im Chassis zu lösen gab.

### 2.2. Versionsmanagement

Für die verteilte Versionsverwaltung aller Dokumente, Quellcode-Management und zum Teilen der Fotos greifen wir auf die freie Software Git zurück.

### 2.3. Meetings

Zu Beginn jedes Projekttag besprechen wir, was zu tun ist und wer welche Aufgabe übernimmt. Am Ende erfolgt ein Statusabgleich was geschafft wurde, welche Probleme auftraten und welche Lösungsansätze für diese in Frage kommen.

## 3. Entwicklungswerkzeuge

### 3.1. Hardware

#### **Werkzeug:**

3D-Drucker, Akkuschauber, Multitool (Dremel), diverse Schraubendreher, Spitzzange, Seitenschneider, Lötkolben, Innensechskant-Schlüssel, Heißklebepistole, Alleskleber, Feile, Cuttermesser, Heißluftföhn, Schere

#### **Daten Verarbeitung:**

PC bzw. Notebook, Smartphone

#### **Hardware OpenBot:**

Arduino-Nano, Motoren mit Getriebe, Motorenregler, Reifen, Drehzahlmesser, Ultraschallsensor, OLED-Display, LEDs, 12V Lithium-Ionen-Akku, diverse Kabel, Widerstände, diverses Kleinmaterial z.B. Schrauben


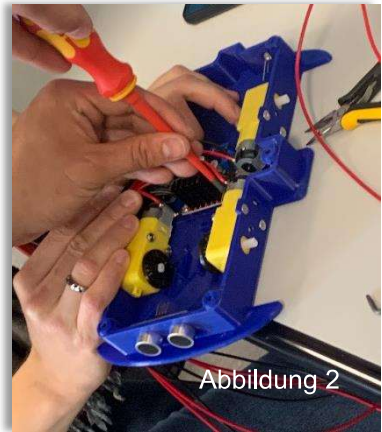


### 3.2. Software



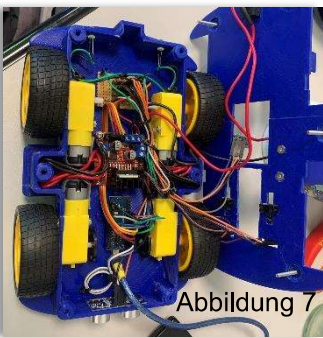

Folgende Software wird benötigt:

Jupyter Notebook	Entwicklungsumgebung für strukturierte Programme
Visual Studio Code	Quelltext-Editor
Anaconda Navigator	Distribution für die Programmiersprachen Python und R, die unter anderem die Entwicklungsumgebung Spyder, den Kommandozeileninterpreter Python und die Webanwendung Jupyter Notebook enthält
Arduino IDE	Entwicklungsumgebung
Python	Objekt-orientierte Programmiersprache

## 4. Entwicklung

### 4.1. Roboter und Software

Tag	Agenda
29.04.2022	<p>Erste Informationen erhalten, Überblick verschaffen, installieren und einrichten der Software Git und OpenBot-OneNote erhalten. Materialausgabe an die Teams, einlesen in die Konstruktion des Roboters, ausbrechen von Strukturmaterial, welches Reste vom 3D-Druck des OpenBot Grundgerüst sind.</p>
05.05.2022	<p>Download und Installation der Programmene Visual-Studio-Code und Anaconda-Navigator. Absprechen der Gruppenverteilung, wer macht was.</p> <p>Zusammenbau und Einbau der Komponenten in den Body des OpenBot, Löten der Motoranschlüsse, Kontakte der Sensoren und des Arduino-Nano um 90 biegen, einkleben der LED's und Montage der einzelnen Komponenten.</p> <div data-bbox="1005 761 1388 1041" data-label="Image">  <p>Abbildung 1</p> </div> <div data-bbox="1005 1048 1388 1478" data-label="Image">  <p>Abbildung 2</p> </div>
06.05.2022	<p>Löten einer Platine und bestücken mit Widerständen und Anschlusspins für Spannungsteiler und Vorwiderstände der LED's (Verbesserung um ein ordentliches Inneres zu generieren)</p> <p>Problem: Deckel des OpenBot lässt sich nicht schließen, da die Steckkontakte des Arduino-Nano die Höhe des Roboters übersteigen.</p> <p>Einlesen in Jupyter-Notebooks und Fehlerbeseitigung, sowie hinzufügen von fehlenden Bibliotheken.</p> <div data-bbox="1149 1512 1388 1758" data-label="Image">  <p>Abbildung 3</p> </div> <div data-bbox="1149 1780 1388 2038" data-label="Image">  <p>Abbildung 4</p> </div>

<p>11.05.2022</p>	<p>Lösen des Platzmangels im Roboter-Body: Ausschneiden des Roboterdeckels mittels Multitool (Dremel) und versetzen der Handy-Halterung. Einsetzen der Verteiler Platine und restliche Verdrahtungs- und Lötarbeiten. Fehlerbehebung in Anaconda auf Grund von Problemen bei der Installation. Da der eigentliche Bot noch in Arbeit ist, mit bestehendem Test-OpenBot Bilder auf der Strecke aufgezeichnet um vorab Tests mit dem Jupyter-Notebook durchführen zu können. Automatische Synchronisierung der Bilder mit dem PC eingerichtet.</p>	 <p>Abbildung 5</p>  <p>Abbildung 6</p>
<p>12.05.2022</p>	<p>Es wurde linksseitig die Polarität der Motoren getauscht, um die Rotationsrichtung zu korrigieren. Restliche Verdrahtungsarbeiten und Fertigstellung der Hardware. Erste manuelle Testfahrt mittels Steuerung per Bluetooth-Controller gelungen.</p>	 <p>Abbildung 7</p>
<p>13.05.2022</p>	<p>Weitere Testfahrten durchgeführt und letzte Probleme der Hardware beseitigt. Der OpenBot ist nun bereit für das Einlernen der Strecke.</p>	 <p>Abbildung 8</p>
<p>18.05.2022</p>	<p>Einlernen der Daten durch Abfahren der Strecke und aufnehmen von Bildern via Data-Collection. Anschließendes auswerten der Daten via Google-Colab und erste autonome Testfahrt, welche fehlschlug.</p>	
<p>20.05.2022</p>	<p>Nach dem Einlernen neuer Daten und korrekter Positionierung des Smartphones auf dem OpenBot, ist nun eine autonome Fahrt möglich. (siehe Video)</p>	



--	--

## 4.2. Firmware

```
//-----//
// DEFINITIONS - DO NOT CHANGE!
//-----//
#define DIY 0      // DIY without PCB
#define PCB_V1 1   // DIY with PCB V1
#define PCB_V2 2   // DIY with PCB V2
#define RTR_TT 3   // Ready-to-Run with TT-motors
#define RC_CAR 4   // RC truck prototypes
#define LITE 5     // Smaller DIY version for education
#define RTR_520 6  // Ready-to-Run with 520-motors --> select ESP32 Dev Module as board!
#define MTV 7      // Multi Terrain Vehicle --> select ESP32 Dev Module as board!

//-----//
// SETUP - Choose your body
//-----//

// Setup the OpenBot version (DIY,PCB_V1,PCB_V2, RTR_TT, RC_CAR, LITE, RTR_520)
#define OPENBOT DIY
```

Abbildung 9

```
//-----//
// CONFIG - update if you have built the DIY version
//-----//
// HAS_VOLTAGE_DIVIDER          Enable/Disable voltage divider (1,0)
// VOLTAGE_DIVIDER_FACTOR       The voltage divider factor is computed as (R1+R2)/R2
// HAS_INDICATORS               Enable/Disable indicators (1,0)
// HAS_SONAR                    Enable/Disable sonar (1,0)
// SONAR_MEDIAN                 Enable/Disable median filter for sonar measurements (1,0)
// HAS_BUMPER                   Enable/Disable bumper (1,0)
// HAS_SPEED_SENSORS_FRONT      Enable/Disable front speed sensors (1,0)
// HAS_SPEED_SENSORS_BACK       Enable/Disable back speed sensors (1,0)
// HAS_SPEED_SENSORS_MIDDLE     Enable/Disable middle speed sensors (1,0)
// HAS_OLED                     Enable/Disable OLED display (1,0)
// HAS_LEDS_FRONT               Enable/Disable front LEDs
// HAS_LEDS_BACK                Enable/Disable back LEDs
// HAS_LEDS_STATUS              Enable/Disable status LEDs

// PIN_TRIGGER                  Arduino pin tied to trigger pin on ultrasonic sensor.
// PIN_ECHO                     Arduino pin tied to echo pin on ultrasonic sensor.
// MAX_SONAR_DISTANCE           Maximum distance we want to ping for (in centimeters).
// PIN_PWM_T                     Low-level control of throttle via PWM (for OpenBot RC only)
// PIN_PWM_S                     Low-level control of steering via PWM (for OpenBot RC only)
// PIN_PWM_L1,PIN_PWM_L2        Low-level control of left DC motors via PWM
// PIN_PWM_R1,PIN_PWM_R2        Low-level control of right DC motors via PWM
// PIN_VIN                       Measure battery voltage via voltage divider
// PIN_SPEED_LB, PIN_SPEED_RB    Measure left and right back wheel speed
// PIN_SPEED_LF, PIN_SPEED_RF    Measure left and right front wheel speed
// PIN_LED_LB, PIN_LED_RB        Control left and right back LEDs (indicator signals, illumination)
// PIN_LED_LF, PIN_LED_RF        Control left and right front LEDs (illumination)
// PIN_LED_Y, PIN_LED_G, PIN_LED_B Control yellow, green and blue status LEDs
```

Abbildung 10

Abbildung 11

```
//-----DIY-----//
#if (OPENBOT == DIY)
const String robot_type = "DIY";
#define HAS_VOLTAGE_DIVIDER 1
const float VOLTAGE_DIVIDER_FACTOR = (20 + 10) / 10;
const float VOLTAGE_MIN = 2.5f;
const float VOLTAGE_LOW = 9.0f;
const float VOLTAGE_MAX = 12.6f;
const float ADC_FACTOR = 5.0 / 1023;
#define HAS_INDICATORS 1
#define HAS_SONAR 1
#define SONAR_MEDIAN 1
#define HAS_SPEED_SENSORS_FRONT 1
#define HAS_OLED 1
const int PIN_PWM_L1 = 5;
const int PIN_PWM_L2 = 6;
const int PIN_PWM_R1 = 9;
const int PIN_PWM_R2 = 10;
const int PIN_SPEED_LF = 2;
const int PIN_SPEED_RF = 3;
const int PIN_VIN = A7;
const int PIN_TRIGGER = 12;
const int PIN_ECHO = 11;
const int PIN_LED_LI = 4;
const int PIN_LED_RI = 7;
//-----PCB_VI-----//
```

Da die Firmware für alle OpenBot-Varianten ausgelegt ist (siehe 1), ist dieser Code-Abschnitt für unseren OpenBot zutreffend. Die if-Abfrage `OPENBOT == DIY` liefert das Ergebnis „True“ wodurch der untenstehende Code ausgeführt wird.

`#define` beschreibt, welche unten aufgeführten Bauteile im OpenBot verwendet werden. Deshalb haben wir mit der Software Arduino-IDE die markieren Stellen im Programm-Code von „0“ auf „1“ geändert und

anschließend den Arduino-Nano damit bespielt. Die Zahl „1“ steht für „True“, was „verbaut“ oder „aktiv“ entspricht.

Mit den Befehlen `const int` wird den jeweiligen Variablen der Pin auf dem Arduino-Nano zugewiesen.

#### Begriffserklärung:

VOLTAGE_DIVIDER	Spannungsteiler
INDICATORS	Blinker
SONAR	Ultraschallsensor
SONAR_MEDIAN	Median-Filter des Ultraschallsensors
SPEED_SENSORS_FRONT	Geschwindigkeitssensoren
OLED	OLED-Display



### 4.3. Apps

#### **Default:**

Über den Punkt Default kann man mehrere Modi, wie z.B. fahren über die Sensoren auswählen.

#### **Free Roam:**

Mit der „Free Roam“-Funktion lässt sich der OpenBot mittels eines Bluetooth-Controllers steuern.

#### **Data Collection:**

Dieser Punkt wird verwendet um Bild- und Sensordaten der Teststrecke aufzunehmen, welche später über Jupyter-Notebook verarbeitet werden.

#### **Controller Mapping:**

Hier kann man die Eingabe des Bluetooth-Controllers auf einzelne Funktionen des OpenBots überprüfen.

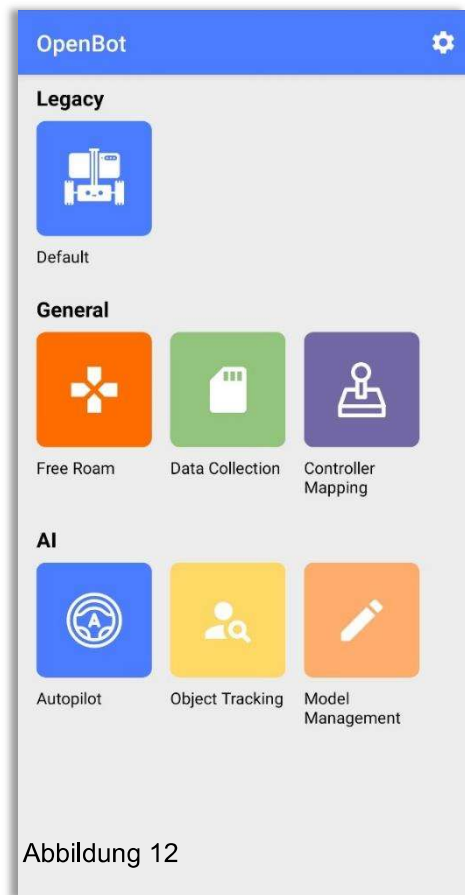


Abbildung 12

#### **Autopilot:**

Nach der Verarbeitung der Daten-Pakete von „Data Collection“ kann man bei der Funktion Autopilot sein bestfunktionierendes oder letztes Datenpaket auswählen. Jetzt kann der OpenBot die Teststrecke autonom abfahren, indem er die aktuellen Bilder der Handy-Kamera mit dem Datenpaket vergleicht.

#### **Object Tracking:**

Bei dieser Option erkennt der OpenBot eine Person mittels der Handy-Kamera und folgt dieser.

#### **Model Managment:**

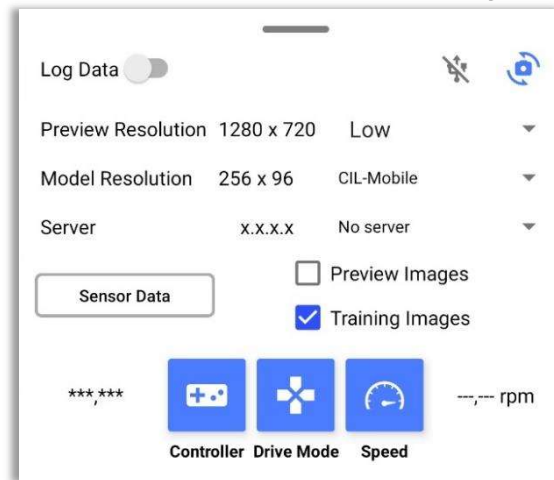
Bei der Einstellung „Model Managment“ kann man die jeweils verarbeiteten Trainingsdateien, welche benötigt werden um autonom zu fahren, importieren oder entfernen.

#### 4.4. Antrainieren des neuronalen Netzes

Um später Referenzbilder für das autonome Fahren zur Verfügung stellen zu können, müssen im ersten Schritt Daten der Strecke gesammelt zu werden. Die Bilder werden über die App „OpenBot“ unter der Einstellung „Data Collection“ aufgenommen. Das Handy, welches am OpenBot montiert ist, wird via Bluetooth mit einem Controller (in unserem Fall Playstation-4) zur manuellen Steuerung verbunden. Jetzt wird die Strecke mehrmals

manuell abgefahren, möglichst ohne Störeinflüsse. Diese sind z.B. sich verändernde Lichteinflüsse, Schattenwurf durch anwesende Personen oder wechselnde Umgebungssituation, wie etwa verstellen von Gegenständen in der Umgebung der Strecke. Hierbei ist es wichtig, dass die Kamera immer in derselben Position auf dem Roboter montiert ist, um auswertebare Daten zu sammeln. Es handelt sich um Bilddateien in Verbindung mit Momentaufnahmen der Sensorik, sprich die aktuelle Drehzahl der Motoren. Anschließend werden die Dateien als Zip-Datei auf den PC übertragen und mittels Jupyter-Notebook verarbeitet (genaue Anleitung siehe Video). Es wird ein neuronales Netz, welches die Bilddateien mit den zugehörigen Sensorstatus verbindet, angelegt. Diese Daten werden nun auf das Smartphone übertragen und die erste autonome Testfahrt kann beginnen.

Abbildung 13



### 5. Ergebnisse

Nach mehrmaligem Einlernen der Teststrecke mit und gegen den Uhrzeigersinn, stellen wir fest, dass sich verändernde Einflüsse, wie der Kamerawinkel, Lichteinflüsse sowie Schattenwurf z.B. durch anwesende Personen eine große Rolle spielen.

Je größer die Menge der aufgenommenen Bilder, desto besser fährt der OpenBot autonom. Bei zu großen Datenmengen (in unserem Fall ca. 50.000 Bilder), ist der Roboter, bzw. die Handy-App, nicht mehr in der Lage die Daten zu verarbeiten, da es schlicht an Rechenleistung des Mobiltelefons scheitert. Der OpenBot bleibt stehen. Hier ist somit der Mittelweg ideal. Diesen erreichen wir, indem wir je ca. 15 Minuten in beiden Fahrtrichtungen Daten aufnehmen und diese auswerten (ca. 25.000 Bilder). Unser OpenBot fährt jetzt die Strecke autonom, sowohl mit als auch gegen den Uhrzeigersinn, perfekt ab (siehe Video).

## **6. Diskussion und Ausblick**

### **6.1. Gelöste und offene Probleme**

Aufgetretene Probleme mit Verbesserungsvorschlägen und Lösungen):

1. Die Drehzahlmessschreibe schleift am Rand der Aussparung. Dies kann durch eine Abstandshülse gelöst werden.
2. Im Gehäuse ist zu wenig Platz für die Steckkontakte, welche auf den Arduino-Nano gesteckt werden. Hierfür wurde eine Aussparung in den Gehäusedeckel geschnitten und die Handyhalterung, welche sich dort befand, nach rechts verschoben.
3. Der Mini-USB-Anschluss am Arduino-Nano ist zu raumfordernd für das Gehäuse. Durch Entfernen des Kunststoff-Mantels am Stecker und ersetzen durch einen Schrumpfschlauch kann dieses Platzproblem behoben werden.
4. Um Ordnung im Gehäuse sicher zu stellen und zur Fixierung der Widerstände wurde eine kleine Platine konstruiert, welche im Gehäuse verklebt wurde.
5. Bei Rechts- und Linksblinken, blinkt immer die linke LED. Nach aktualisieren der OpenBot-App war dies behoben.
6. Häufiger Verlust von Reifen, da diese nur gesteckt waren. Was kein Problem bei Fahrzeugen mit gelenkter Achse ist. Dadurch, dass der OpenBot mittels gegenläufiger Motoren gesteuert wird, entsteht eine starke Kraft, welche die Reifen von den Achsen zieht. Kann gelöst werden mit Kleber oder einer Schraube, welche die Reifen an der Achse fest montiert.
7. Der OpenBot ist sehr anfällig auf äußere Einflüsse wie Änderungen von Licht und Schatten. Dies kann behoben werden mit einem festinstallierten Licht und einem einheitlichen Hintergrund wie z.B. einer weißen Trennwand bis ca. Hüfthöhe um die Teststrecke herum.
8. Die Position des Mobiltelefons muss immer dieselbe sein, was bei dieser Art von Halterung eher schwer zu ermöglichen ist. Hier empfehlen wir, eine nach unten geneigte Halterung, welche zusätzlich das Handy perfekt in einer Position hält, z.B. mit Anschlagpunkt auf einer Seite.

## 6.2. Erweiterungsmöglichkeiten

Viele Punkte haben wir bereits in den Verbesserungsvorschlägen thematisiert wie z.B. ein größeres Gehäuse und den Eigenbau einer Platine, um bessere Organisation der Kabel zu gewährleisten. Außerdem würden wir Motoren mit einem stabileren Getriebe empfehlen, um den Verschleiß gering zu halten.

GPS-Verfolgung und Aufzeichnen dieser Daten, damit der OpenBot sich auch unabhängig von der Kamera orientieren kann und somit eine höhere Ausfallsicherheit gewährleistet werden kann.

Denkbar wäre auch eine Erweiterung mit Sprachsteuerung, möglicherweise über Google-Assistant, z.B. starten und stoppen der Person-Following-Funktion.

## 7. Fazit

Das Projekt „OpenBot“ hat uns ca. 1 Monat im Unterricht begleitet und uns mit völlig neuen Themen wie Git, Arduino und weiteren Programmen konfrontiert. Anfangs war es schwierig sich einen Überblick über das gesamte Projekt und die To-Do's zu verschaffen und diese in leicht zu bearbeitende Häppchen zu unterteilen. Es hat sich jedoch schnell herauskristallisiert, wer in welchem Gebiet Vorreiter ist und wo er seine Stärken am Besten einbringen kann. Somit hat sich die Aufteilung der zu bearbeitenden Punkte dann überraschen einfach gestaltet.

Am Ende wurden alle Abschnitte und angefertigten Dokumente so wie das Video zusammen besprochen und gemeinsam verbessert, was unseren Teamgeist sehr gefördert hat.

Wir empfehlen jeder technisch versierten Klasse dieses oder ein vergleichbares Projekt einmal umzusetzen, da es eine schöne Abwechslung zum oft sehr theoretisch geprägtem Schulalltag ist.

Ein Projekt von A bis Z durchzuführen, zu Dokumentieren und die letzten 5 Wochen Revue passieren zu lassen, hat uns auf die anstehende Projektarbeit und das spätere Berufsleben sehr gut vorbereitet. Alles in allem gehen wir aus dem Projekt „OpenBot“ sehr positiv heraus und sind dankbar für diese Erfahrung.