Universität der Bundeswehr München
ETTI 2: Verteilte Intelligente Systeme
Prof. Dr. rer. nat. Antje Neve
in Zusammenarbeit mit
WIWeB GF250

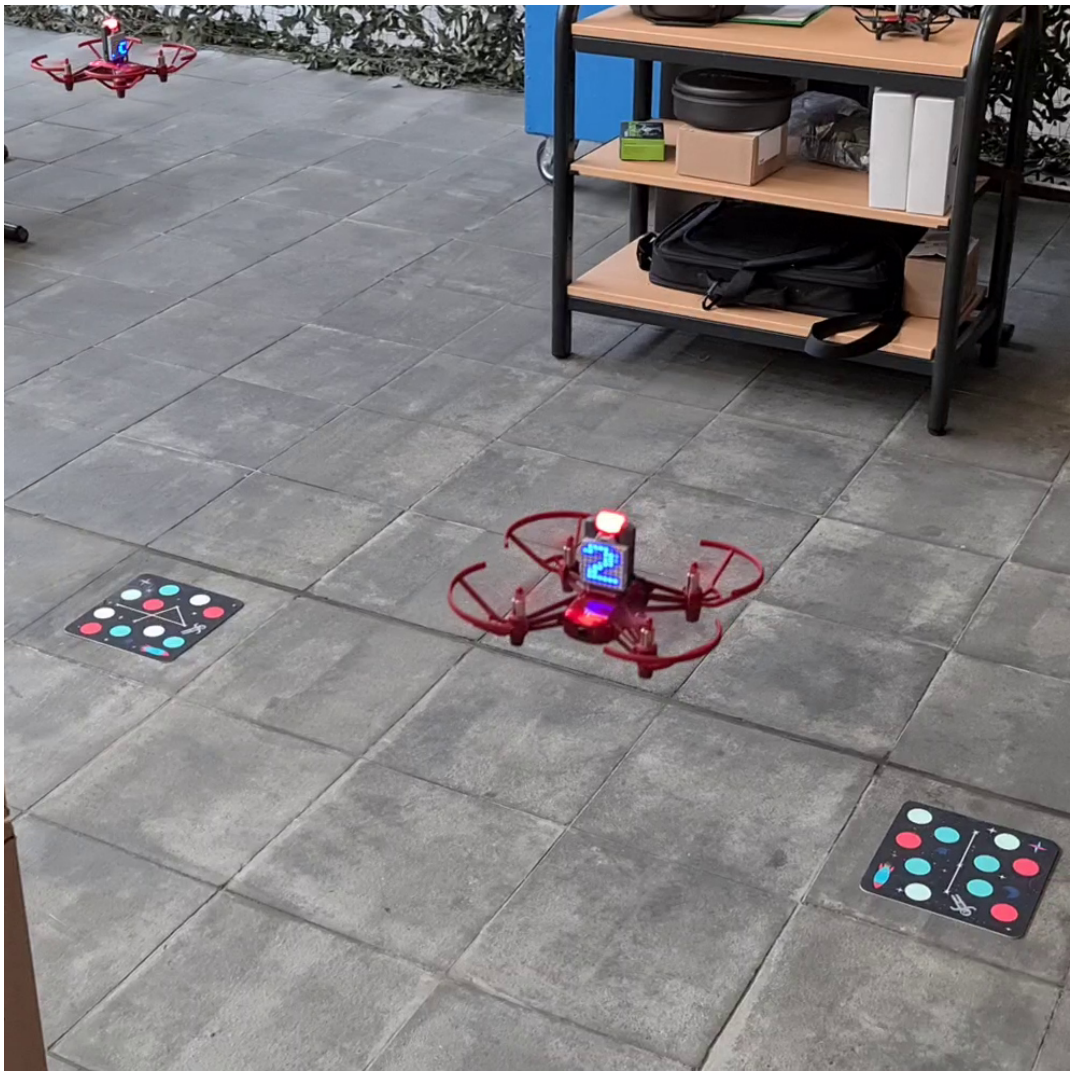
Frühjahrstrimester 2024
Verfasser: Karl Scholz

Internet of Things Praktikum

Teil 4

Informationen zum fliegenden Programmabschnitt

Lesen bis 17.05.2024 08:00



Arbeitsanweisung

Lesen Sie sich dieses Dokument bis zum 17.05.2024 um 08:00 durch. Sie müssen in diesem Dokument in der Vorbereitung nicht aktiv werden. Eine Schritt-für-Schritt-Anleitung, wie sie es bisher gewohnt waren wird es für diesen Teil nicht geben, hier sollen Sie freier Programmieren. Dieses Dokument soll daher auch als Nachschlagewerk während des Praktikums dienen.

Sollten sich bei der Vorbereitung Fragen ergeben, können Sie diese gerne im Vorfeld adressieren und sich per Mail melden. Auch zu Beginn des Praktikumstermins wird es die Möglichkeit geben einzelne Fragen zu stellen. Beachten Sie jedoch, dass das Praktikum eine gewissenhafte Vorbereitung voraussetzt.

Inhaltsverzeichnis

Inhalt

Abkürzungsverzeichnis

COM	serielle COMmunication Schnittstelle
ESP32	Espressif 32
IoT	Internet of Things
GPIO	General Purpose Input / Output
GND	Ground / Masse
I^2C	Inter-Integrated Circuit Bus
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit (Gyroskop und Beschleunigungsmesser)
IPv4	Internet Protocol Version 4
LED	Light Emitting Diode
LiDAR	Light Detection And Ranging
MP	Mission Pad
MQTT	Message Queuing Telemetry Transport
PIO	PlatformIO
PWM	Pulse Width Modulation
QoS	Quality of Service (MQTT)
RGB	Red Green Blue
SDK	Software Development Kit
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VSCode	Visual Studio Code
WiFi	Wireless Fidelity IEEE-802.11

1 Mission Pads

Die *Tello EDU* arbeitet mit sog. *Mission Pads (MP)*, welche sie mit ihrer Optical Flow Kamera erkennen kann. Diese 20 x 20 cm Matten sind mit den Zahlen von 1-8 kodiert.

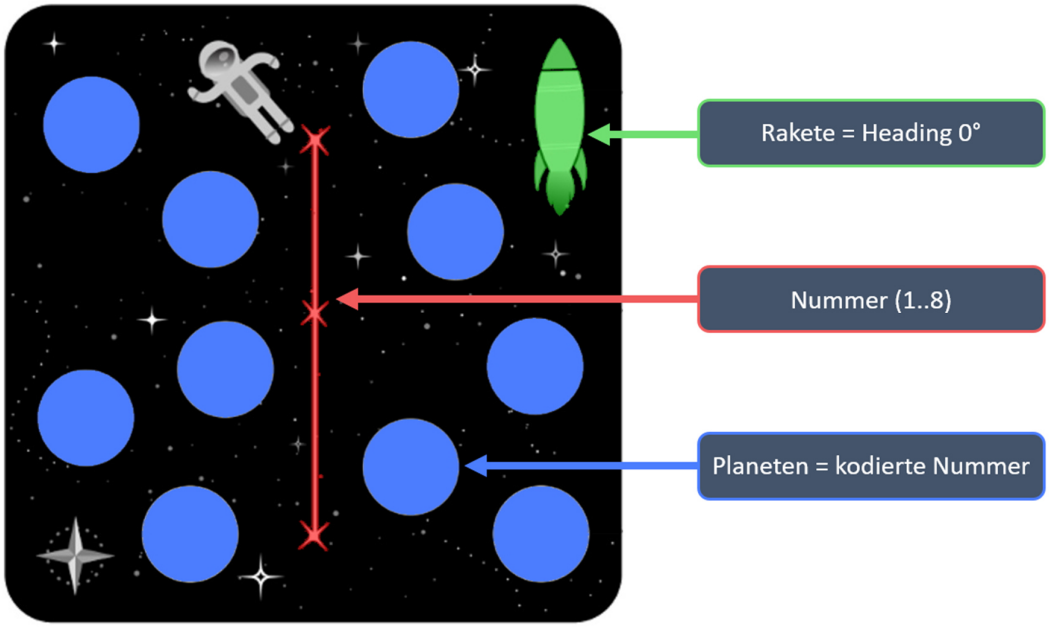


Abbildung 1: Elemente des Mission Pads

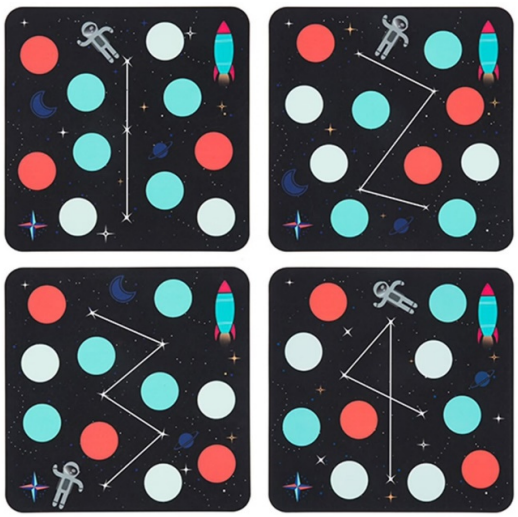


Abbildung 2: kodierte Mission Pads 1-4

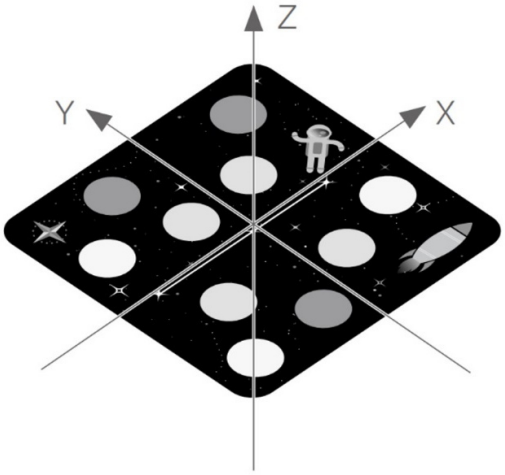


Abbildung 3: Koordinatensystem des Mission Pads

Kommandos für Mission Pads

Für die Mission Pads gibt es zwei wichtige Kommandos:

1. Go Kommando

```
"go <X> <Y> <Z> <Speed> <MP>"
"go  0   0   50   100    m-2" //Beispiel: Auf 50cm Höhe mittig Ausrichten
```

Mit dem *go* Kommando lässt sich die Tello absolut an einer Position über dem Mission Pad positionieren. Ein Anwendungsbeispiel wäre das genaue Ausrichten nach dem Startvorgang mittig über dem *Mission Pad*.

2. Jump Kommando

```
"jump <X> <Y> <Z> <Speed> <Heading> <currentMP> <targetMP>"
"jump 100  0   0    100       0         m1          m2");
//Beispiel M2 liegt ca. einen Meter vor M1
```

Das *jump* Kommando bedeutet, dass die *Tello EDU* erst einen groben Flug zur Position X, Y relativ vom jetzigen Mission Pad, auf der Höhe Z ausführt. Dort angekommen sucht sie das Ziel-Mission-Pad, richtet sich mittig aus und dreht sich, sodass die Frontkamera relativ zum Raketensymbol den Winkel <Heading> einschließt.

<MP>	Bedeutung
m1	Mission Pad 1
m2	Mission Pad 2
…	…
m8	Mission Pad 8
m-1	Zufälliges Mission Pad (sinnlose Funktion, da Sichtbereich sehr eingeschränkt)
m-2	Das näheste Mission Pad

Informationen zu weiteren Befehlen → *Tello_SDK_3.0_User_Guide_en.pdf*

2 Versuchsaufbau

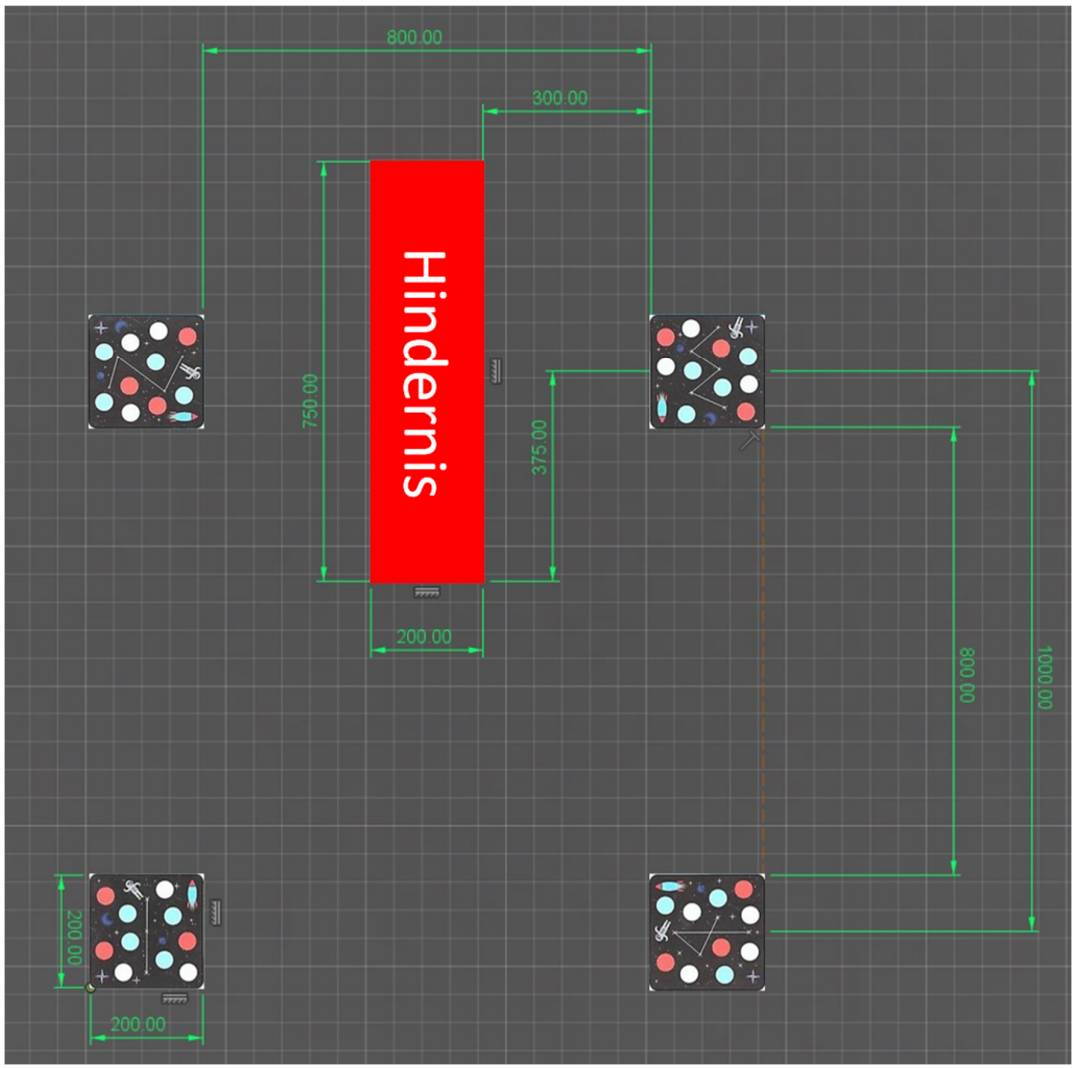


Abbildung 4: bemaßter Versuchsaufbau [mm]

Zu Beginn ihres Praktikums werden Sie aufgefordert die Mission Pads wie in der Abbildung gezeigt auf dem Boden zu platzieren und den Weg zwischen Mission Pad 2 und 3 mit einem Hindernis, welches höher ist als Sie fliegen wollen, zu blockieren. Die Mission Pad Erkennung funktioniert auf 80 cm Höhe am besten.

3 Szenario und Missionsablauf

Sie werden in diesem Internet of Things Praktikum allein oder in einer Gruppe eine Drohne programmieren, die einen Parcours bewältigen soll. Allerdings müssen Sie sich diesen Parcours mit einem Partner bzw. einer Partnergruppe teilen. Um eine Kollision zu vermeiden, müssen ihre Drohnen miteinander über das *Message Queuing Telemetry Transport (MQTT)* Protokoll kommunizieren. Dafür bekommen Sie eine *AircraftID*, zum Beispiel *UniBwTello_3*, sodass sie im Netzwerk eindeutig sind.

Ablauf:

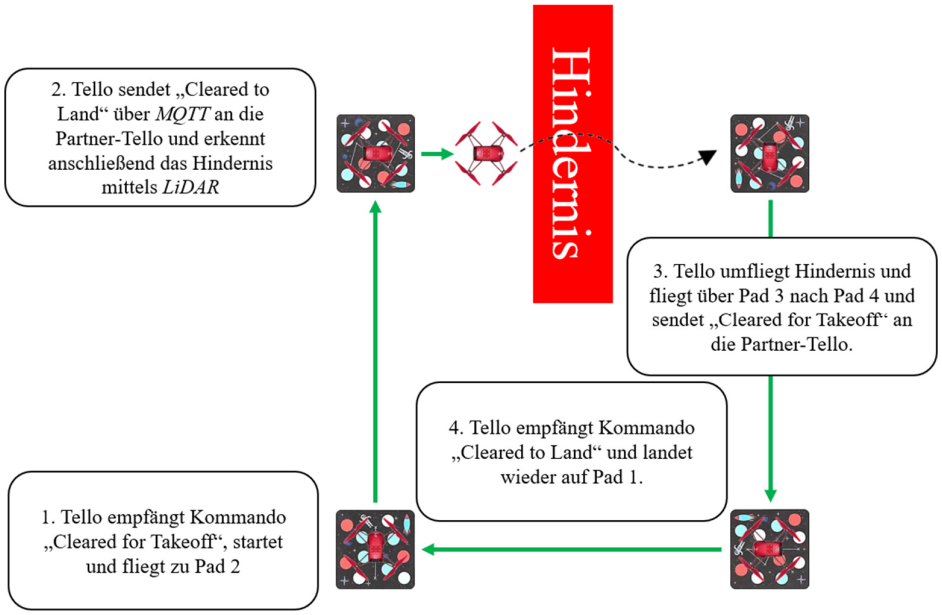


Abbildung 5: Szenario und Missionsablauf

Tello1 wird auf das *MP1* gestellt, bekommt über das *MQTT* Protokoll manuell den Startbefehl und fliegt anschließend um den Kurs. Sie erkennt das Hindernis mit dem *LiDAR* Sensor und umfliegt es automatisch. Währenddessen wird *Tello2* auf das *MP1* gestellt und angeschaltet. Kommt die *Tello1* zum *MP4*, wird Sie darüber schwebend stehen bleiben und mittels *MQTT* der *Tello2* den Startbefehl geben. *Tello2* startet dann und gibt *Tello1,* sobald Sie weg ist, die Landeerlaubnis. *Tello1* landet und die beiden Rollen sind getauscht. Theoretisch sollte der Ablauf dann unbegrenzt wiederholbar sein und die beiden *Tellos* den Parcours abfliegen, das Hindernis umfliegen und miteinander kommunizieren.

4 Fliegender Programmabschnitt

4.1 Kurzerklärung SDK

Zuletzt geht es noch um den Zustandsautomaten und die eigentliche Logik hinter dem Ablauf, wie er in der Einleitung im letzten Unterpunkt skizziert wurde. Folgende Informationen fehlen Ihnen für die Umsetzung noch:

- *SDK* Kommandos an die *Tello EDU* schicken

Alle *SDK* Kommandos, die sie in der Datei *Tello_SDK_3.0_User_Guide_en.pdf* finden, können Sie an die Funktion

```
int RMTT_Protocol::sendTelloCtrlMsg(char *cmd_str)
```

des in *main.cpp* erstellten Objekts *tt_sdk* der Klasse *RMTT_Protocol* übergeben. Diese Funktion sendet das Kommando nicht einfach, sondern stellt auch sicher, dass es ausgeführt wird. So lange blockiert die Funktion, das heißt, dass die nächste Zeile ihres Programms erst ausgeführt wird, wenn die Ausführungsbestätigung von der *Tello* zurückgekommen ist. Wenn Sie zum Beispiel das Kommando

```
tt_sdk.sendTelloCtrlMsg("takeoff");
```

schicken und in der nächsten Zeile

```
iZahl = 3;
```

steht, dann wird die Variable *iZahl* erst nachdem die *Tello* ihre Motoren angeschaltet, gestartet und sich auf 50 cm stabilisiert hat auf den Wert *3* gesetzt.

Wenn Sie einen Befehl ausführen wollen, auf den es keinen Rückgabewert gibt, z.B.:

rc a b c d	Set the lever force values for the four channels of the remote control. a: roll (-100 to 100) b: pitch (-100 to 100) c: throttle (-100 to 100) d: yaw (-100 to 100)	No response

Abbildung 6: Auszug SDK: "rc" Kommando

Dann nutzen Sie nicht die *sendTelloCtrlMsg()* Funktion, weil die ja auf einen Rückgabewert wartet, sondern

```
tt_sdk.SendCMD("rc 0 20 0 0");
```

Dies würde zum Beispiel „Fliege mit 20% Geschwindigkeit nach vorne“ bedeuten. Dieser Flugbefehl ist als einer der wenigen relativ. Absolute Flugbefehle wie zum Beispiel:

```
tt_sdk.sendTelloCtrlMsg("forward 20");
```

benutzen den nach unten gerichteten *Optical Flow Sensor* um 20 cm vorwärts zu fliegen.

- *LiDAR* Sensor auslesen

Das Objekt *tt_lidar* der Klasse *RMTT_TOF* wird im *Background* verwaltet und hat eine Funktion, um den Wert des *LiDAR* Abstands in **Millimetern** als *unsigned integer* auszugeben.

```
uint16_t RMTT_TOF::ReadRangeSingleMillimeters(void)
```

- *LiDAR* Daten auf *LED*-Matrix anzeigen lassen

In *Background.cpp* ist folgende Funktion definiert:

```
void DisplayLidarOnMatrix(float fRange)
```

Rufen Sie diese auf und übergeben Sie dieser einfach einen Wert vom Datentyp *float* in der Einheit **Meter** und sie wird den Abstand visuell auf der *LED*-Matrix darstellen.

4.2 Zusammenfassung wichtiger SDK-Kommandos

1. Abhebe Kommando

```
"takeoff" //Tello startet und schwebt auf 50 cm Höhe über dem Startpunkt
```

2. Lande Kommando

```
"land" //Tello sinkt und stellt die Motoren ab sobald sie am Boden ist
```

3. Bewege Kommandos

```
"<Direction> <x in cm [20..500]>"
"forward 100" //Beispiel: Einen Meter nach vorne fliegen
//<Direction> = up, down, left, right, forward, back
```

4. Dreh Kommando

```
"<Direction> <x degrees>"
"cw 90" //Beispiel: Rechtsdrehung um 90°
//<Direction> = cw, ccw
```

5. Steuerknüppel Kommando

```
"rc <roll> <pitch> <height> <yaw>"
"rc   20     100       0      25 " //Die Tello fliegt eine Kurve bei 100%
Vorwärtsgeschwindigkeit mit 20% seitlicher Geschwindigkeit und 25% Dreh-
geschwindigkeit um die Hochachse auf gleichbleibender Höhe.
```

6. Positionier Kommando

```
"go <X> <Y> <Z> <Speed> <MP>"
"go  0   0   50   100    m-2" //Beispiel: Auf 50cm Höhe mittig Ausrichten
```

7. Missionpadwechsel Kommando

```
"jump <X> <Y> <Z> <Speed> <Heading> <currentMP> <targetMP>"
"jump 100  0   80    100      0         m1          m2";
//Beispiel M2 liegt ca. einen Meter vor M1
```

8. Missionpaderkennungs Kommando

```
"<mon / moff>"
"mon";
//MP-Erkennung muss einmalig nach dem Abhebe Kommando aktiviert werden
```

9. Missionpadskonfigurations Kommando

```
"mdirection <0 = untere Kamera / 1 = Frontkamera / 2 = beide>"
"mdirection 2";
//Beispiel M2 liegt ca. einen Meter vor M1
```

Weitere Kommandos → *Tello_SDK_3.0_User_Guide_en.pdf*

4.3 Zustandsautomat

Als Unterstützung wird Ihnen noch folgendes Zustandsdiagramm zur Verfügung gestellt:

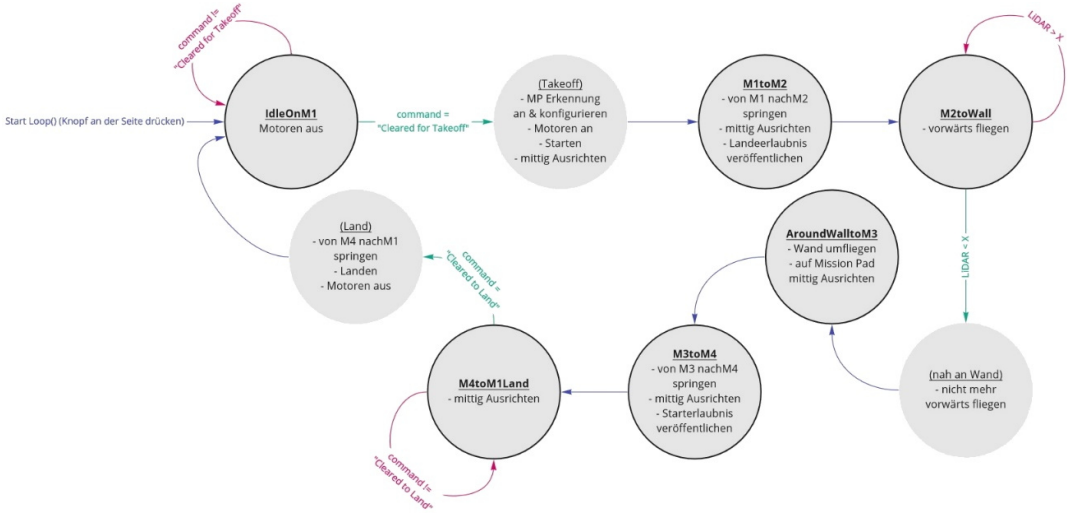


Abbildung 7: Zustandsdiagramm des Automaten

Um Ihnen das Scrollen zu ersparen, finden Sie hier nochmal ein Bild des Versuchsaufbaus.

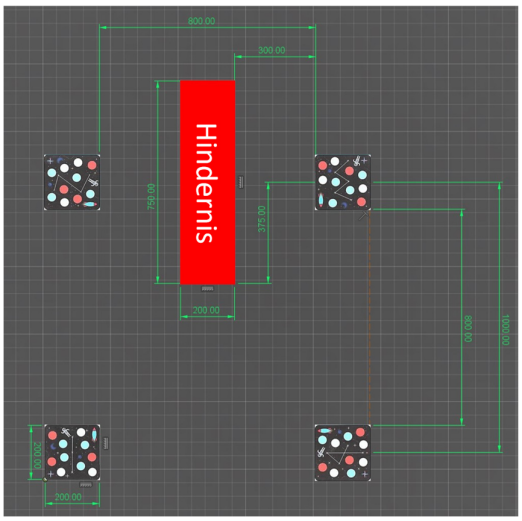


Abbildung 8: bemaßter Versuchsaufbau [mm]

„*Tello1* wird auf das *MP1* gestellt, bekommt über das *Message Queuing Telemetry Transport (MQTT)* Protokoll manuell den Startbefehl und fliegt anschließend um den Kurs. Sie erkennt das Hindernis mit dem *LiDAR* Sensor und umfliegt es automatisch. Währenddessen wird *Tello2* auf das *MP1* gestellt und angeschaltet. Kommt die *Tello1* zum *MP4*, wird Sie darüber schwebend stehen bleiben und mittels *MQTT* der *Tello2* den Startbefehl geben. *Tello2* startet dann und gibt *Tello1,* sobald Sie weg ist, die Landeerlaubnis. *Tello1* landet und die beiden Rollen sind getauscht. Theoretisch sollte der Ablauf dann unbegrenzt wiederholbar sein und die beiden *Tellos* den Parcours abfliegen, das Hindernis umfliegen und miteinander kommunizieren.“

Abbildungsverzeichnis

Alle Abbildungen dieses Dokumentes sind Screenshots von Windows, Linux, Mac, Visual Studio Code oder MQTT-Explorer und wurden vom Verfasser erstellt.