Follow-Me Fahrroboter: Testdokumentation

Inhaltsverzeichnis

1. Test: Bewegungskalkulation + Verlaufsdokumentation

1.1. Durchführung Test Bewegungskalkulation

1.1.1. Ziel

- Die prozentuale Abweichung zwischen dem gemessenen Wert und dem eingegebenen Wert wird ermittelt, miteinander verglichen und anschließend wird der Durchschnitt dieser Abweichungen berechnet.

1.1.2. Vorbereitung

Auf dem Arduino

- Wir nutzen den standardisierten Code des Prototyps.

Code Arduino

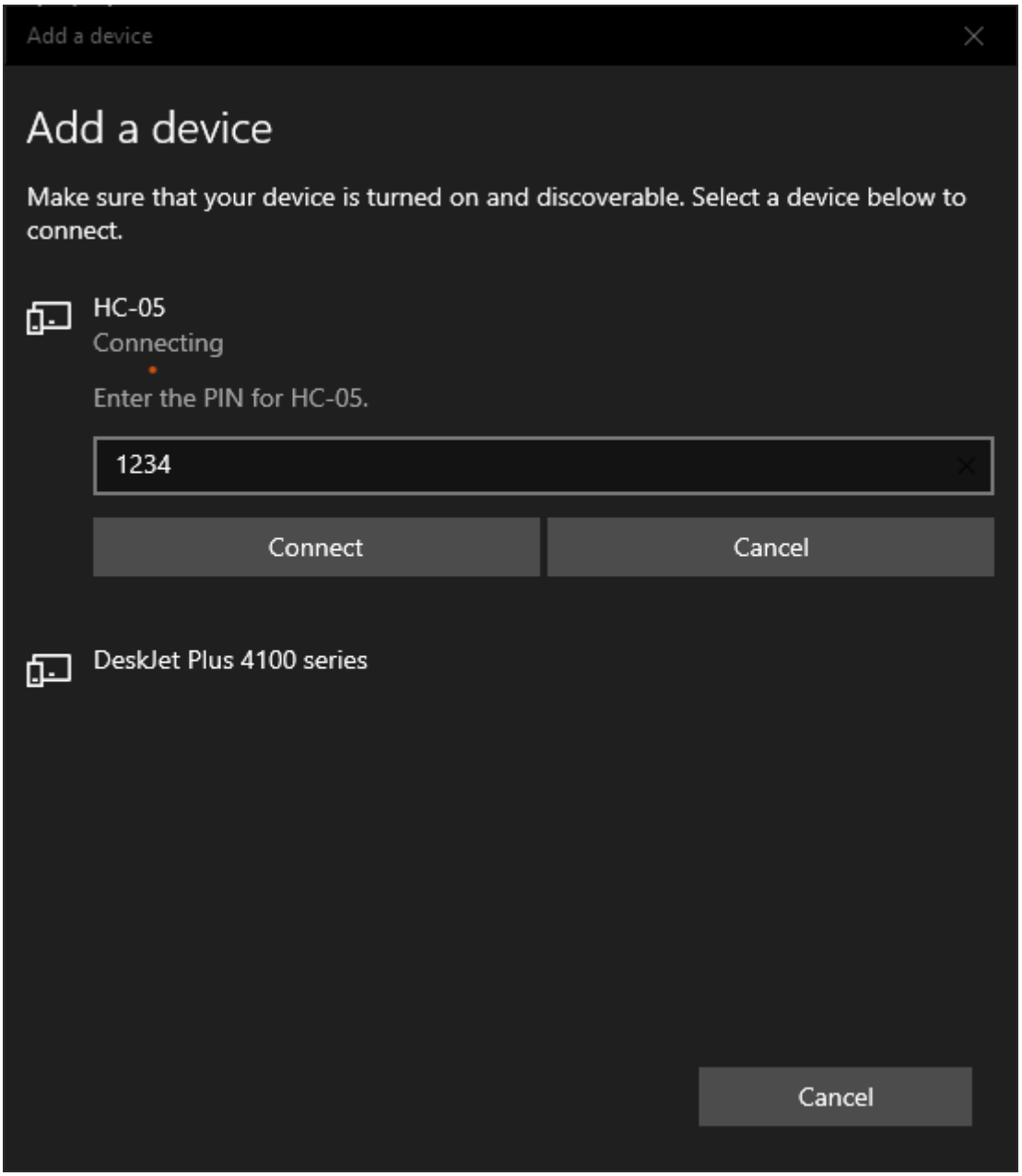
- "Es wird ein Arduino Uno mit einem Adafruit Motor Shield v2.3 und einem Bluetooth-Modul HC-05 verwendet, jedoch ohne eine Verbindung zum Jetson Nano. Das Bluetooth-Modul ist ein Ergänzungsmodul und muss separat installiert werden."

Auf dem PC

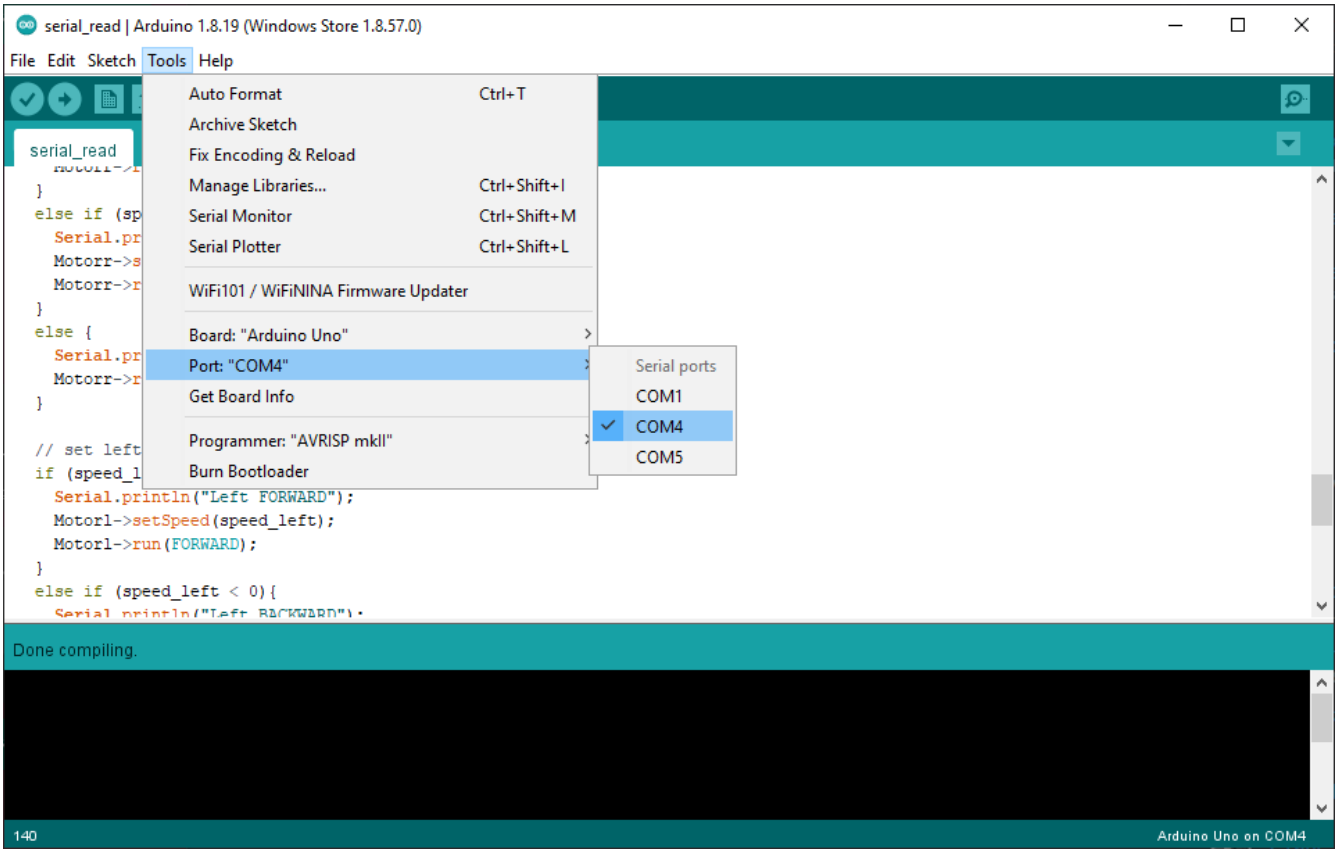
- Für den Test wird folgender Code verwendet:

Code Test

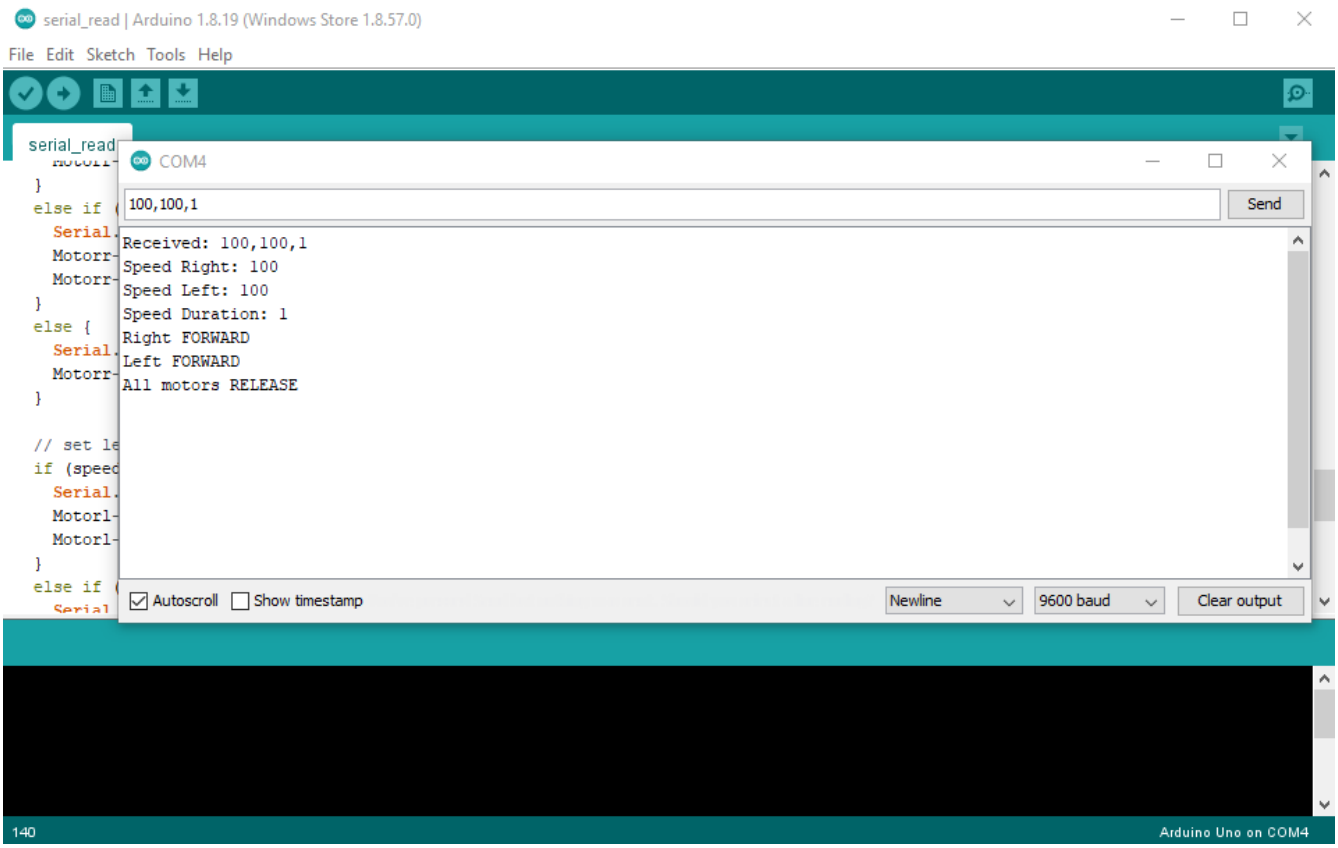
- Vor dem Start des Tests muss der PC über Bluetooth mit dem Arduino verbunden werden. (Passwort: 1234)



- Dann muss der COM-Port des Arduinos ermittelt werden, beispielsweise mittels der Arduino IDE.



- Testen der Connection mit dem Serial Monitor.

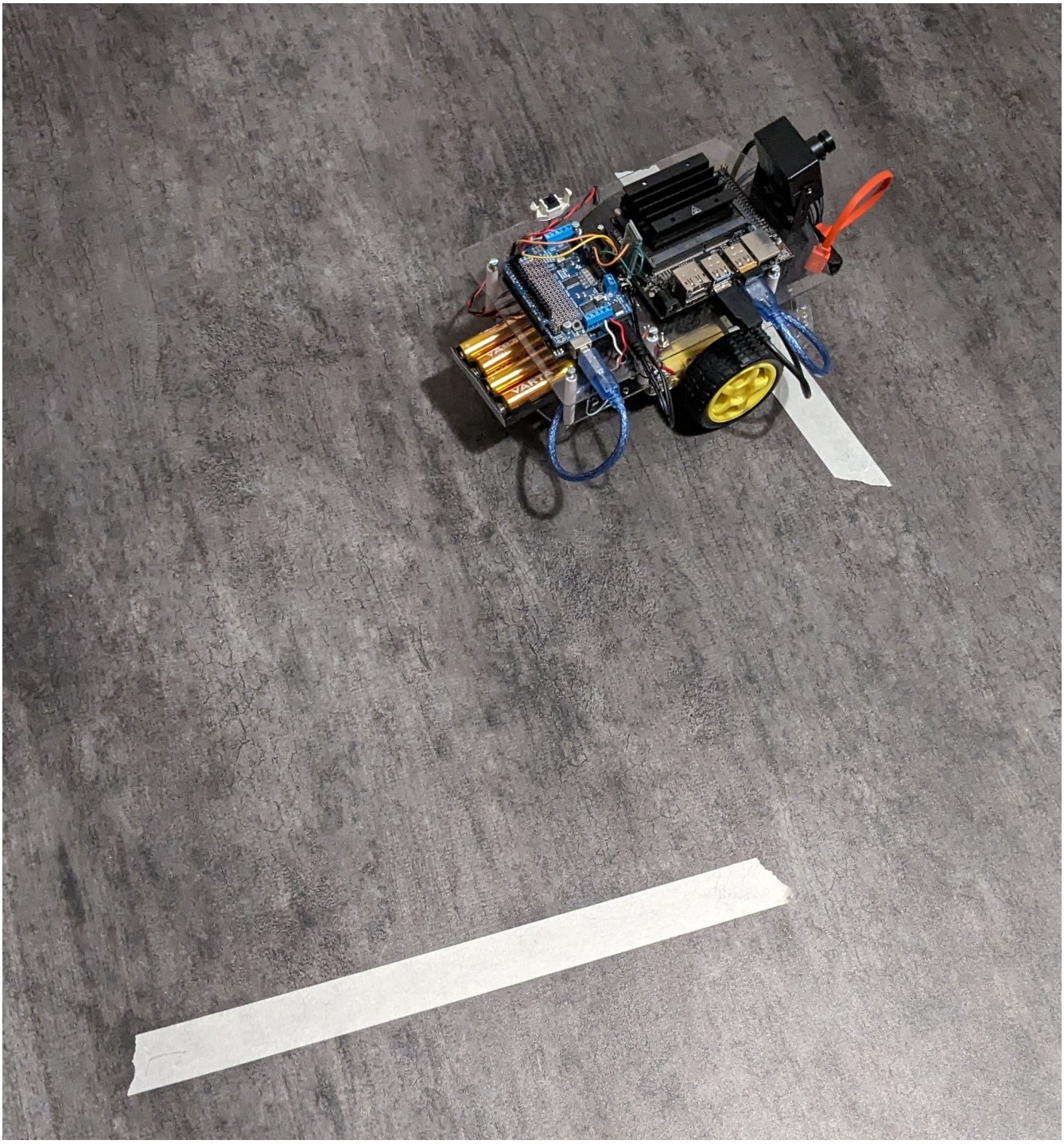


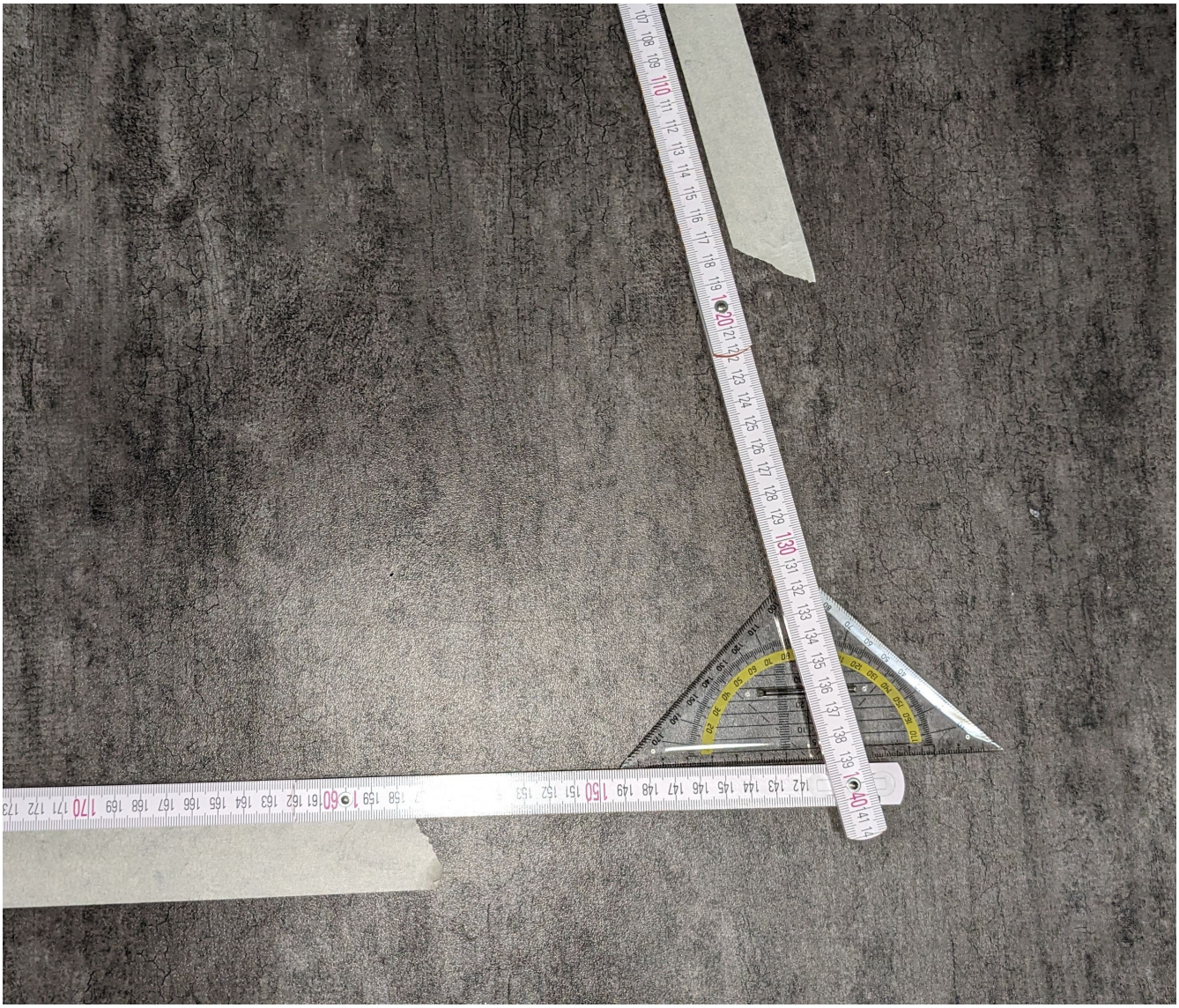
- Anschließend muss der Port im Code auf dem PC angepasst werden.

1.1.3. Durchführung

Auf dem Arduino

- Der Arduino wird auf dem Boden platziert und die Position der Räder wird mit Klebeband markiert.
- Start Test auf PC
- Daraufhin wird die neue Position des Roboters mit Klebeband markiert, und der Winkel zur alten Position wird gemessen.



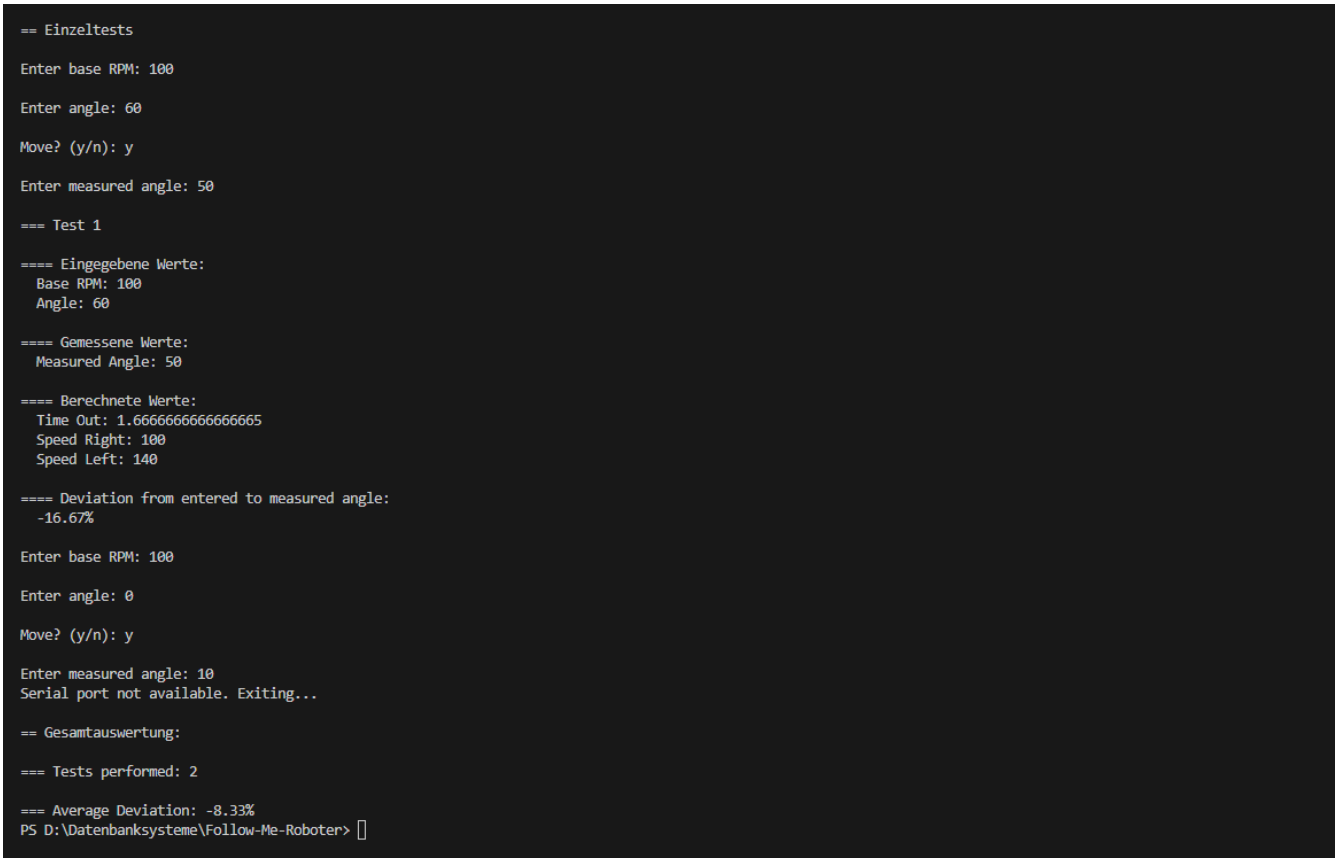


1.1.4. Auf dem PC

- Der Test wird gestartet
- Das Programm fordert den Nutzer dann auf 3 bis 4 Werte einzugeben:
  - Die Base RPM (Basis Geschwindigkeit)
  - Den Winkel
  - Ob sich der Roboter von der Stelle bewegen soll oder nicht. (y/n)
  - Bei komplett gerader Bewegung soll angegeben werden wie lange der Roboter fahren soll
- Nachdem der Test durchgeführt wurde, fordert das Programm den Nutzer auf den gemessenen Winkel einzugeben.

1.1.5. Auswertung

- Um den Nutzer eine Auswertung zu erleichtern, sind Eingabe, Ausgabe und Werte für jeden Test sowie eine Gesamtauswertung für die Werte aller Tests zum am Ende der Tests in der Konsole ausgegeben



1.2. Verlaufsdokumentation Bewegungskalkulation

Enter the number of test runs: 3

1.2.1. Einzeltests

Enter base RPM: 100

Enter angle: 50

Move? (y/n): y

Enter measured angle: 60

Test 1

Eingegebene Werte:

```
Base RPM: 100
Angle: 50
```

Gemessene Werte:

```
Measured Angle: 60
```

Berechnete Werte:

```
Time Out: 1.3888888888888888
Speed Right: 100
Speed Left: 140
```

Deviation from entered to measured angle:

```
20.00%
```

Enter base RPM: 100

Enter angle: 40

Move? (y/n): n

Enter measured angle: 30

Test 2

Eingegebene Werte:

```
Base RPM: 100
Angle: 40
```

Gemessene Werte:

```
Measured Angle: 30
```

Berechnete Werte:

```
Time Out: 0.2222222222222222
Speed Right: -100
Speed Left: 100
```

Deviation from entered to measured angle:

```
-25.00%
```

Enter base RPM: 100

Enter angle: 80

Move? (y/n): y

Enter measured angle: 60

Test 3

Eingegebene Werte:

```
Base RPM: 100
Angle: 80
```

Gemessene Werte:

```
Measured Angle: 60
```

Berechnete Werte:

```
Time Out: 2.2222222222222223
Speed Right: 100
Speed Left: 140
```

Deviation from entered to measured angle:

```
-25.00%
```

1.2.2. Gesamtauswertung:

Tests performed: 3

Average Deviation: -10.00%

2. Test: Humandetection + Verlaufsdokumentation

2.1. Durchführung Human Detection Test

2.1.1. Ziel des Tests

Der Test soll die Effektivität des Human-Detectors-Test und wie lange dieser tatsächlich im Durchschnitt braucht, um eine Person zu erkennen.

2.1.2. Vorbereitung

Projektverzeichnis

- **Verzeichnis:** `C:\Users\jonat\PycharmProjects\pythonProject`
- **Videodatei:** `one-by-one-person-detection.mp4`

Testskript

- **Skript:** `test.py`
- **Funktion:** Human Detection mittels YOLOv3 zur Erkennung und Protokollierung der Erkennungszeiten.

2.1.3. Durchführung des Tests für 50 Frames

Startzeit

- **Datum und Uhrzeit:** 2024-06-06 14:00:00 (wird während der Ausführung protokolliert)

Ausführung des Skripts

- **Befehl:** `python test.py`
- **Beschreibung:** Das Skript wird ausgeführt und die Erkennungszeiten für 50 Frames werden gemessen und protokolliert.

Ergebnisspeicherung

- **Datei:** `erkennungszeiten.txt`
- **Inhalt:**

```
Startzeit: 2024-06-05 23:56:52
```

```
Erkennungszeit: 0.88 Sekunden
Erkennungszeit: 0.23 Sekunden
Erkennungszeit: 0.15 Sekunden
...
Erkennungszeit: 1.09 Sekunden
```

```
Durchschnittliche Erkennungszeit: 0.24 Sekunden
```

```
Endzeit: 2024-06-05 23:57:04
```

2.1.4. Human Detection Testdokumentation Nr. 2

2.1.5. Ziel des Tests

Es soll die Effektivität des Human-Detector getestet werden und wie lange er tatsächlich im Schnitt braucht, um eine Person zu erkennen.

2.1.6. Vorbereitung

Projektverzeichnis

- **Verzeichnis:** `C:\Users\jonat\PycharmProjects\pythonProject`
- **Videodatei:** `one-by-one-person-detection.mp4`

Testskript

- **Skript:** `test.py`
- **Funktion:** Human Detection mittels YOLOv3 zur Erkennung und Protokollierung der Erkennungszeiten.

2.1.7. Durchführung des Tests für 150 Frames

Startzeit

- **Datum und Uhrzeit:** 2024-06-06 14:05:00 (wird während der Ausführung protokolliert)

Anpassung des Skripts

- **Anzahl der Frames:** Von 50 auf 150 ändern.

Ausführung des Skripts

- **Befehl:** `python test.py`
- **Beschreibung:** Das Skript wird erneut ausgegeben

2.2. Verlaufsdokumentation Human Detection

2.2.1. Vorbereitung

Projektverzeichnis

- Verzeichnis: `C:\Users\jonat\PycharmProjects\pythonProject`
- Videodatei: `one-by-one-person-detection.mp4`

Testskript

- Skript: `test.py`
- Funktion: Human Detection mittels YOLOv3 zur Erkennung und Protokollierung der

Erkennungszeiten.

2.2.2. Durchführung des Tests für 50 Frames

Startzeit

- Datum und Uhrzeit: 2024-06-06 14:00:00 (wird während der Ausführung protokolliert)

Ausführung des Skripts

- Befehl: `python test.py`
- Beschreibung: Das Skript wird ausgeführt und die Erkennungszeiten für 50 Frames werden gemessen und protokolliert.

Ergebnisspeicherung

- Datei: `erkennungszeiten.txt`
- Inhalt: Die Erkennungszeiten pro Frame und die durchschnittliche Erkennungszeit werden protokolliert.

Endzeit

- Datum und Uhrzeit: 2024-06-06 14:01:00 (wird während der Ausführung protokolliert)

2.2.3. Durchführung des Tests für 150 Frames

Startzeit

- Datum und Uhrzeit: 2024-06-06 14:05:00 (wird während der Ausführung protokolliert)

Anpassung des Skripts

- Öffne die Datei `test.py` und ändere die Anzahl der Frames von 50 auf 150: ```python num_frames = 150

---

3. Unit-Test: Bewegungskontroll-Node + Verlaufsdokumentation

3.1. Durchführung Unit-Test Bewegungskontroll-Node

3.1.1. Ziel

Der Test zielt darauf ab, die korrekte Funktion der Bewegungskontroll-Node zu verifizieren, indem die erwarteten und empfangenen Ergebnisse für Motor- und Servosteuerungen verglichen werden.

3.1.2. Vorbereitung

- Für den Unit-Test wird die Bewegungskontroll-Node geprüft: https://github.com/cl-ire/camera_package/blob/main/camera_package/movement_control_node.py
- Die Bewegungskontroll-Node wird von einer anderen Node getestet: https://github.com/cl-ire/camera_package/blob/main/camera_package/test_movement_control_node.py
- Die Konsole auf einem PC mit der eingerichteten Projektumgebung muss geöffnet werden und folgende Befehle müssen nacheinander ausgeführt werden:

```
cd ~/ros2_humble/ && source install/setup.bash
cd ~/ros2_ws/ && source install/setup.bash
cd ~/ros2_ws/src/camera_package/launch
```

3.1.3. Durchführung

- Nach abgeschlossener Vorbereitung muss folgender Befehl in der Konsole ausgeführt werden:

```
ros2 launch test_follow_me_arduino_launch.py
```

- Der Test startet automatisch und die Bewegungskontroll-Node wird getestet.
- Der Fortschritt wird in der Konsole angezeigt

3.1.4. Auswertung

- Jeder Test beinhaltet 5 Durchläufe, deren Ergebnisse wie folgt strukturiert sind:

```
[test_movement_control-2] number: 0
[test_movement_control-2]         position_data: [x, x, x, x, x, x, x, x, x, x, x, x]
[test_movement_control-2]         servo_expected_results: [x, x]
[test_movement_control-2]         motor_expected_results: [x, x, x, x, x]
[test_movement_control-2]         servo_received_results: [x, x]
[test_movement_control-2]         motor_received_results: [x, x, x, x, x]
[test_movement_control-2]         servo_test_success: True/False
[test_movement_control-2]         motor_test_success: True/False
```

- Ob der Test erfolgreich war, ist an den letzten Zeilen der Ausgabe zu erkennen.
- Fehlerquoten können durch Vergleichen von expected_results mit received_results errechnet werden.

3.2. Verlaufsdokumentation Bewegungskalkulation

Number of test runs: 2

3.2.1. Test 1

- Erfolgt ohne Testen von Servomotoren

Erwartungen

- *servo_test_succes* soll fehlschlagen, da keine Servos getestet werden
- *motor_test_success* soll gelingen
- *motor_expected_results* soll mit *motor_received_results* übereinstimmen

Durchführung

- erhaltene Daten:

```
[test_movement_control-2] number: 0
[test_movement_control-2]         position_data: [-693, 158, 162, 626, 1920, 1080, 1,
0, 1142, 25306, 1142, 27306]
[test_movement_control-2]         servo_expected_results: []
[test_movement_control-2]         motor_expected_results: [131, 100, 499, 0, 0]
[test_movement_control-2]         servo_received_results: []
[test_movement_control-2]         motor_received_results: [131, 100, 499, 0, 0]
[test_movement_control-2]         servo_test_success: False
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 1
[test_movement_control-2]         position_data: [200, 100, 100, 500, 1920, 1080, 1,
0, 1142, 34306, 1142, 36306]
[test_movement_control-2]         servo_expected_results: []
[test_movement_control-2]         motor_expected_results: [100, 131, 133, 0, 0]
[test_movement_control-2]         servo_received_results: []
[test_movement_control-2]         motor_received_results: [100, 131, 133, 0, 0]
[test_movement_control-2]         servo_test_success: False
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 2
[test_movement_control-2]         position_data: [1000, 50, 170, 700, 1920, 1080, 1,
0, 1142, 43306, 1142, 45306]
[test_movement_control-2]         servo_expected_results: []
[test_movement_control-2]         motor_expected_results: [100, 131, 733, 0, 0]
[test_movement_control-2]         servo_received_results: []
[test_movement_control-2]         motor_received_results: [100, 131, 733, 0, 0]
[test_movement_control-2]         servo_test_success: False
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 3
[test_movement_control-2]         position_data: [-300, -10, 100, 500, 1920, 1080, 1,
0, 1142, 52306, 1142, 54306]
```

```
[test_movement_control-2]         servo_expected_results: []
[test_movement_control-2]         motor_expected_results: [131, 100, 216, 0, 0]
[test_movement_control-2]         servo_received_results: []
[test_movement_control-2]         motor_received_results: [131, 100, 216, 0, 0]
[test_movement_control-2]         servo_test_success: False
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 4
[test_movement_control-2]         position_data: [0, 0, 100, 500, 1920, 1080, 1, 0,
1143, 1306, 1143, 3306]
[test_movement_control-2]         servo_expected_results: []
[test_movement_control-2]         motor_expected_results: [100, 100, 1000, 0, 0]
[test_movement_control-2]         servo_received_results: []
[test_movement_control-2]         motor_received_results: [100, 100, 1000, 0, 0]
[test_movement_control-2]         servo_test_success: False
[test_movement_control-2]         motor_test_success: True
```

Ergebnisse

- *servo_test_succes* ist wie erwartet fehlgeschlagen
- *motor_test_success* ist wie erwartet gelungen
- *motor_expected_results* stimmt wie erwartet mit *motor_received_results* überein

⇒ Der Test war somit erfolgreich.

3.2.2. Test 2

- Erfolgt mit Testen von Servomotoren

Erwartungen

- *servo_test_succes* soll gelingen, da Servos getestet werden
- *motor_test_success* soll gelingen
- *motor_expected_results* soll mit *motor_received_results* übereinstimmen
- *servo_exptected_results* soll mit *servo_received_results* übereinstimmen

Durchführung

- erhaltene Daten:

```
[test_movement_control-2] number: 0
[test_movement_control-2]         position_data: [-693, 158, 162, 626, 1920, 1080, 1,
0, 1148, 34176, 1148, 36176]
[test_movement_control-2]         servo_expected_results: [0, -30]
[test_movement_control-2]         motor_expected_results: [131, 100, 499, 0, 0]
[test_movement_control-2]         servo_received_results: [0, -30]
[test_movement_control-2]         motor_received_results: [131, 100, 499, 0, 0]
```

```
[test_movement_control-2]         servo_test_success: True
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 1
[test_movement_control-2]         position_data: [200, 100, 100, 500, 1920, 1080, 1,
0, 1148, 43176, 1148, 45176]
[test_movement_control-2]         servo_expected_results: [0, -22]
[test_movement_control-2]         motor_expected_results: [100, 131, 133, 0, 0]
[test_movement_control-2]         servo_received_results: [0, -22]
[test_movement_control-2]         motor_received_results: [100, 131, 133, 0, 0]
[test_movement_control-2]         servo_test_success: True
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 2
[test_movement_control-2]         position_data: [1000, 50, 170, 700, 1920, 1080, 1,
0, 1148, 52176, 1148, 54176]
[test_movement_control-2]         servo_expected_results: [0, 22]
[test_movement_control-2]         motor_expected_results: [100, 131, 733, 0, 0]
[test_movement_control-2]         servo_received_results: [0, 22]
[test_movement_control-2]         motor_received_results: [100, 131, 733, 0, 0]
[test_movement_control-2]         servo_test_success: True
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 3
[test_movement_control-2]         position_data: [-300, -10, 100, 500, 1920, 1080, 1,
0, 1149, 1176, 1149, 3176]
[test_movement_control-2]         servo_expected_results: [0, 9]
[test_movement_control-2]         motor_expected_results: [131, 100, 216, 0, 0]
[test_movement_control-2]         servo_received_results: [0, 9]
[test_movement_control-2]         motor_received_results: [131, 100, 216, 0, 0]
[test_movement_control-2]         servo_test_success: True
[test_movement_control-2]         motor_test_success: True
[test_movement_control-2]
[test_movement_control-2]
[test_movement_control-2] number: 4
[test_movement_control-2]         position_data: [0, 0, 100, 500, 1920, 1080, 1, 0,
1149, 10176, 1149, 12176]
[test_movement_control-2]         servo_expected_results: [0, 9]
[test_movement_control-2]         motor_expected_results: [100, 100, 1000, 0, 0]
[test_movement_control-2]         servo_received_results: [0, 9]
[test_movement_control-2]         motor_received_results: [100, 100, 1000, 0, 0]
[test_movement_control-2]         servo_test_success: True
[test_movement_control-2]         motor_test_success: True
```

Ergebnisse

- *servo_test_succes* ist wie erwartet gelungen

- *motor_test_success* ist wie erwartet gelungen
- *motor_expected_results* stimmt wie erwartet mit *motor_received_results* überein
- *servo_expected_results* stimmt wie erwartet mit *servo_received_results* überein

⇒ Der Test war somit erfolgreich.
