

# Test Cases: Follow-Me Fahrroboter

## Test: Bewegungskalkulation + Verlaufsdokumentation

### Durchführung Test Bewegungskalkulation

---

#### Ziel

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

#### Vorbereitung

##### Auf dem Arduino

- Wir nutzen den standardisierten Code des Prototyps.

##### Code Arduino

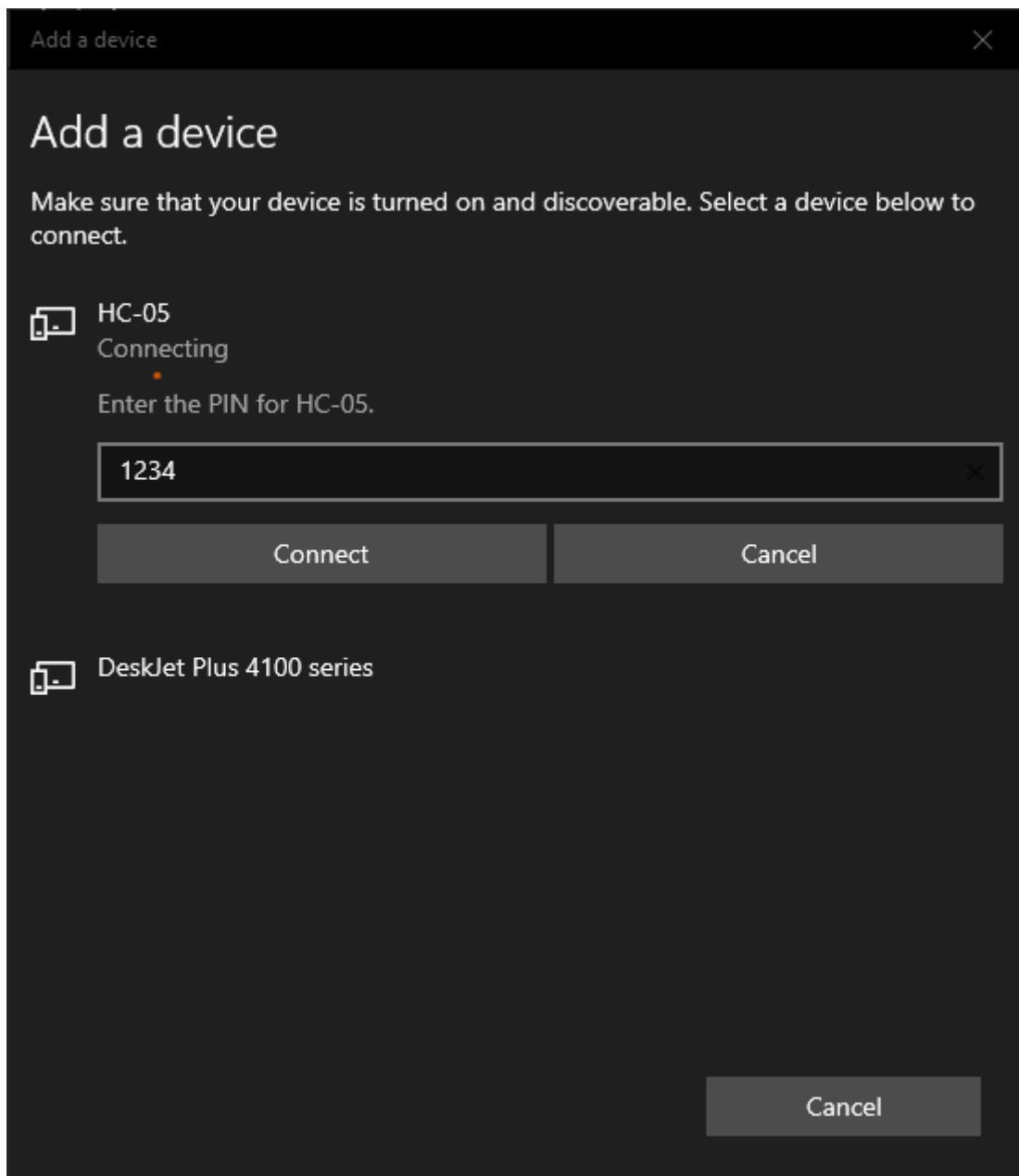
- "Es wird ein Arduino Uno mit einem Adafruit Motor Shield v2.3 und einem Bluetooth-Modul HC-05 verwendet, jedoch ohne eine Verbindung zum Jetson Nano. Das Bluetooth-Modul ist ein Ergänzungsmodul und muss separat installiert werden."

##### 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)



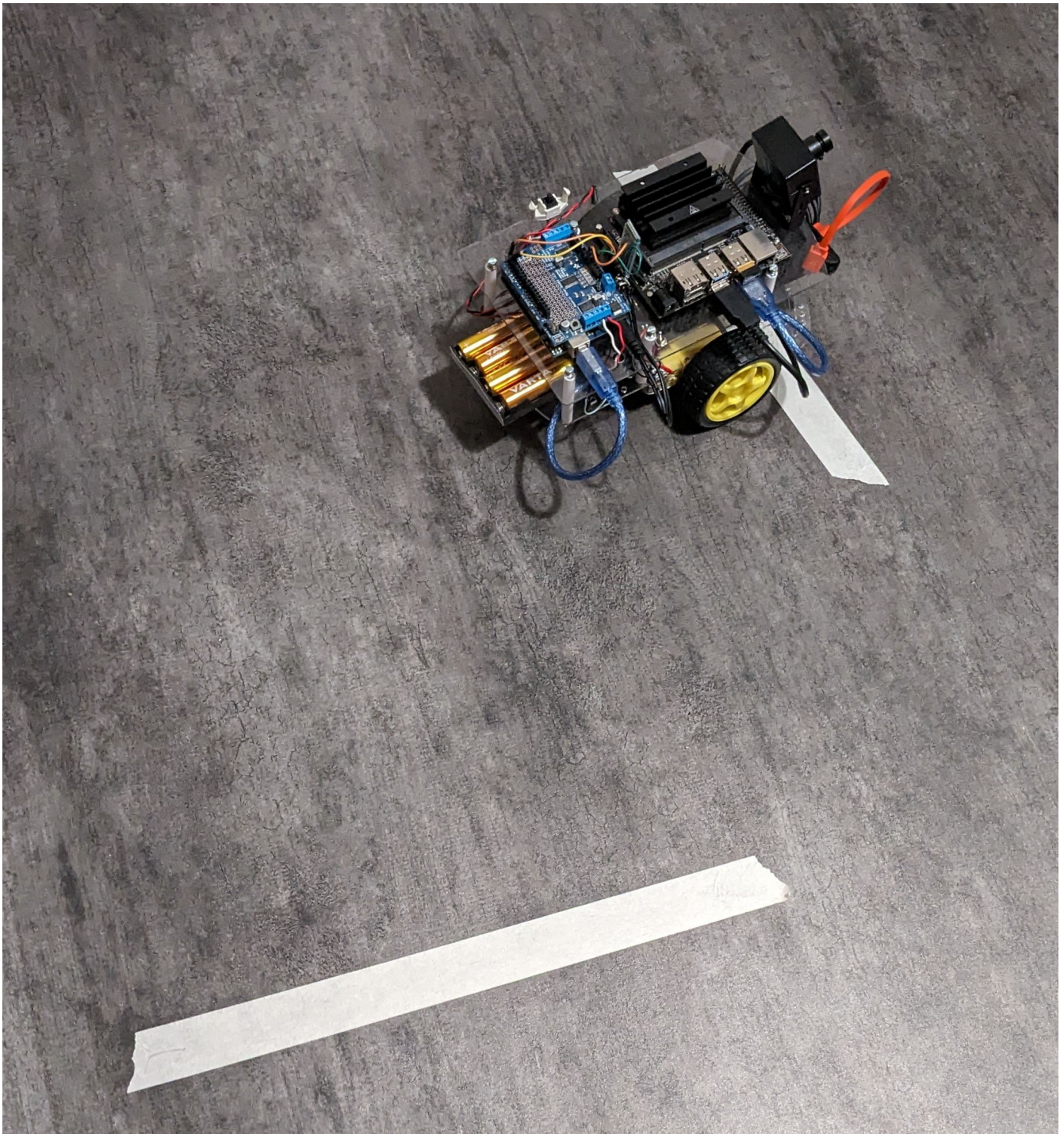
- Dann muss der COM-Port des Arduinos ermittelt werden, beispielsweise mittels der Arduino IDE.



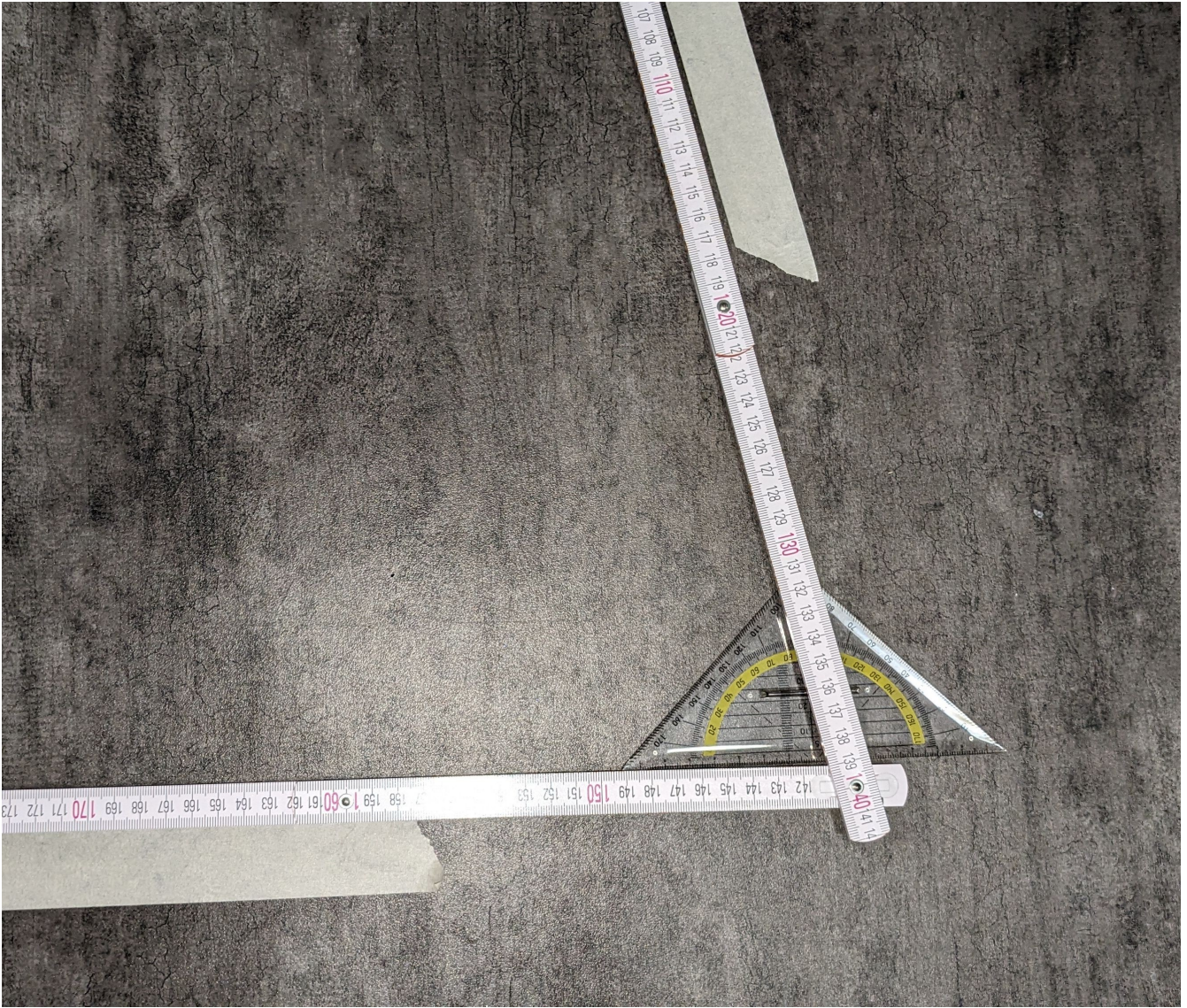
# Durchführung

## 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.







## 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.

---

## 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

```

== Einzeltests

Enter base RPM: 100

Enter angle: 60

Move? (y/n): y

Enter measured angle: 50

=== Test 1

==== Eingegebene Werte:
Base RPM: 100
Angle: 60

==== Gemessene Werte:
Measured Angle: 50

==== Berechnete Werte:
Time Out: 1.6666666666666665
Speed Right: 100
Speed Left: 140

==== Deviation from entered to measured angle:
-16.67%

Enter base RPM: 100

Enter angle: 0

Move? (y/n): y

Enter measured angle: 10
Serial port not available. Exiting...

== Gesamtauswertung:

=== Tests performed: 2

=== Average Deviation: -8.33%
PS D:\Datenbanksysteme\Follow-Me-Roboter> 

```

# Verlaufsdokumentation Bewegungskalkulation

Enter the number of test runs: 3

## 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: 2.222222222222223  
Speed Right: 100  
Speed Left: 140

#### Deviation from entered to measured angle:

-25.00%

### Gesamtauswertung:

Tests performed: 3

Average Deviation: -10.00%

## Test: Humandetection + Verlaufsdokumentation

### Durchführung Human Detection Test

#### 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.

#### 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.

## 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:** erknennungszeiten.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

## Human Detection Testdokumentation Nr. 2

## 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.

## 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.

## 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

---

# Verlaufsdokumentation Human Detection

## 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.

## 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)

## 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`
-