Dokumentation

im Fach Künstliche Intelligenz im Rahmen der Ausbildung zum Staatlich geprüften Elektrotechniker an der Staatlichen Technikerschule Neumarkt i.d.OPf.

Roboterspiel

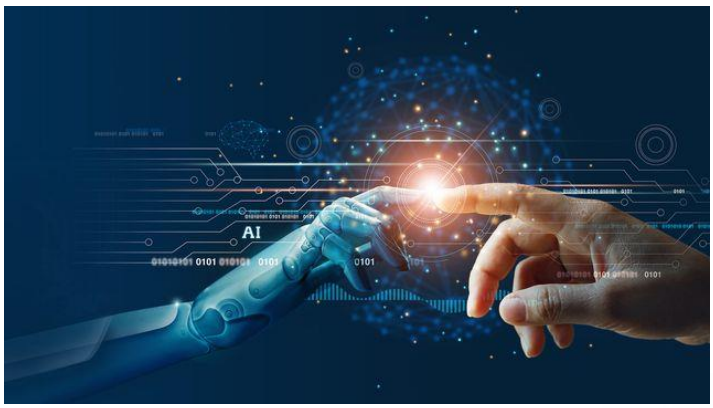
Vorgelegt von: Maximilian Diepold, Michael Meier, Pia Ackermann

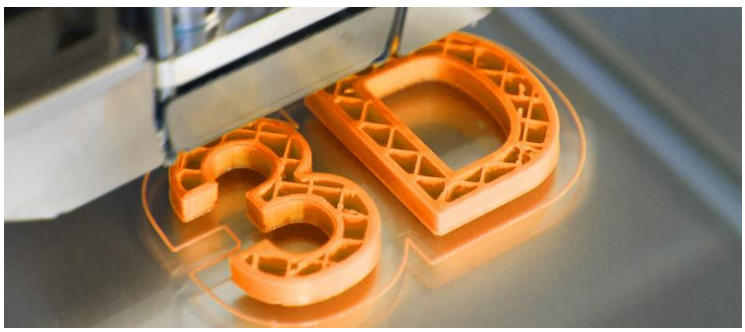
am: 06.04.2022

Inhaltverzeichnis

1 Einführung

Im Rahmen eines KI-Projektes soll mit einem Universal Roboter UR3 eine Erkennung von farbigen Buchstaben umgesetzt werden, um diese im Nachgang farblich zu sortieren. Hierfür wurden insgesamt 6 Buchstaben mit dem 3D-Drucker gedruckt, welche zusammen den Schriftzug „AI FOR U“ ergeben. Hierbei ist es das Ziel, das zunächst die Buchstaben, die zu Beginn in einer zufällig gewählten Reihenfolge aufgestellt sind, mit Hilfe der Kamera des Roboters und einem externe Python-Skript erkannt und anhand deren Form bzw. Farbe sortiert werden. Die Buchstaben stehen hierbei in einer fest vergebenen Position, mit Hilfe einer Platte mit Aussparungen und werden in andere dazu farblich passende Platten mit Aussparungen, im weiteren Boxen genannt, sortiert. Sobald die Sortierung abgeschlossen ist, soll der Roboter die Buchstaben wieder nehmen und zurück an die Ursprungspositionen stellen, jedoch nun in der richtigen Reihenfolge, sodass der Schriftzug zu lesen ist.





2 Hardware

2.2 Universal Robot UR3

Der Roboter, der verwendet werden soll, ist ein UR3 der Firma Universal Robots. Der Greifer am Roboter hat eine eingebaute Kamera von Robtiq. Es gibt zwei Möglichkeiten den Roboter zu steuern, entweder über das Bedienpanel oder über das Internet mit Hilfe eines Remote Access.

2.3 Windows Rechner

Für die Ausführung des Python Skriptes verwenden wir eine handelsüblichen Windows Laptop, jedoch kann hier auch jede andere Plattform, welche pythonfähig und netzwerkfähig ist, verwendet werden. Ein Beispiel wäre hierfür ein Raspberry Pi.

2.4 Buchstaben und Boxen

Die Buchstaben und Boxen wurden zuerst im CAD-Zeichenprogramm SolidWorks 2021 konstruiert, im Programm Ultimaker Cura gesliced und mit einem 3D-Drucker von Anycubic ausgedruckt. Das A hat die Farbe Grau, das I Rot, das F Grün, das O Gelb, das R Blau und das U Weiß. Die Boxen wurden jeweils in den passenden Farben vom Buchstaben gedruckt und die Platte, in welcher die Buchstaben am Anfang stehen, in schwarz.

2.5 3D-Drucker

Um die Buchstaben, die Platte und die Boxen zur drucken wurde der 3D-Drucker „Anycubic i3 mega S“ verwendet. Dieser verwendet das 3D-Druck Verfahren Fused Layer Modeling auch FLM genannt. Die Düse bringt dabei schichtweise den angeschmolzenen pastösen Kunststoff auf. Als Filament wurde für gelb, schwarz, weiß, grau, blau und grün PLA verwendet und für rot PETG.

3 Remote Access

Der Remote Control (Fernzugriff) des Universal Roboters ist mithilfe der Software Robotics Insights umgesetzt.

Die Software befindet sich auf dem weißen USB Stick. Beim Neustart des Roboters wird der USB Stick nicht sofort mit gebootet, deswegen muss man den Stick nochmal neu einstecken.

Überprüfen kann man die erfolgreiche Integration der Software am Panel unter den Reitern „Programmieren“ -> „Installation“ -> „Insights“.

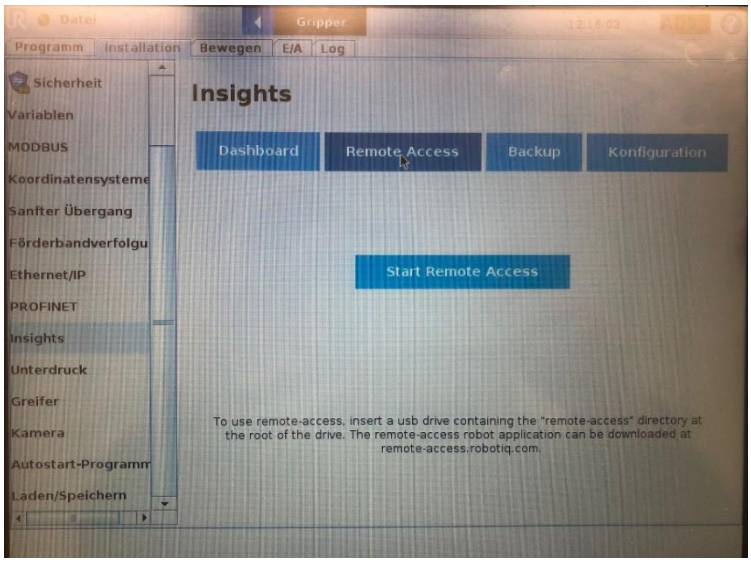
Dort wechselt mit Hilfe des Remote Access Buttons zur Seite um den Fernzugriff zu Starten.

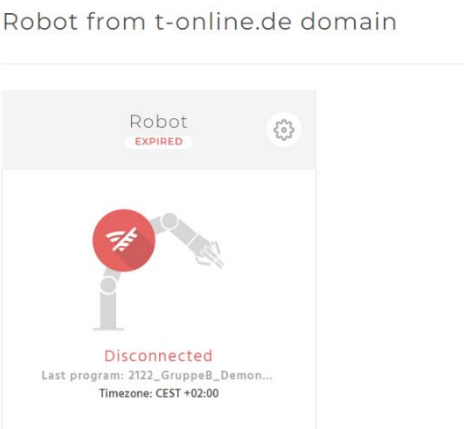
Als Nächste sollte man mit der Hilfe des nachfolgenden Links über den PC auf die Website der Universal Roboter und zur Insight-Website gelangen.

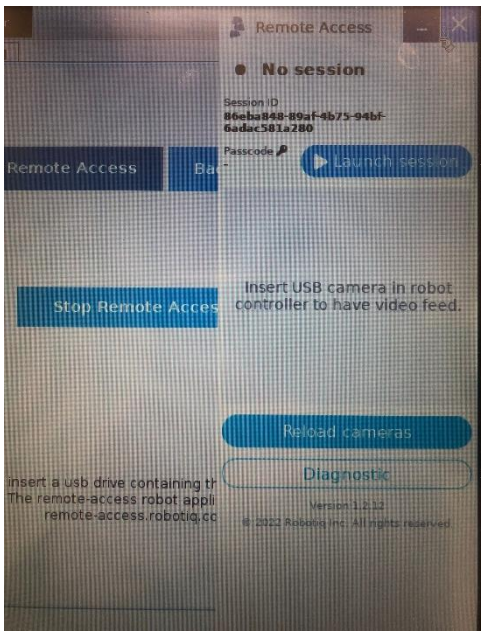
https://insights-app.robotiq.com/robots/fd2f2137-b11b-4063-970a-6712c795c022/logs?selectedTimeUnit=HOURS

Im nächsten Schritt kann man am Roboter “Session starten” und das Passwort ablesen.

Dieses Passwort dann am PC in der Insight App eingeben und der Fernzugriff läuft.





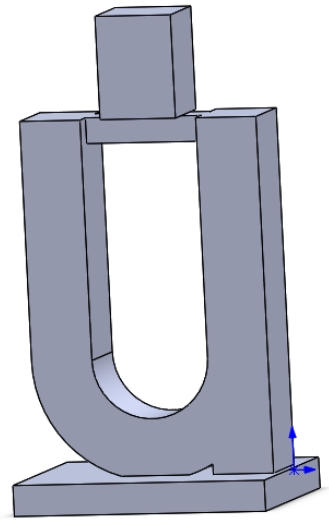


4 Projektumsetzung

4.1 CAD Umsetzung

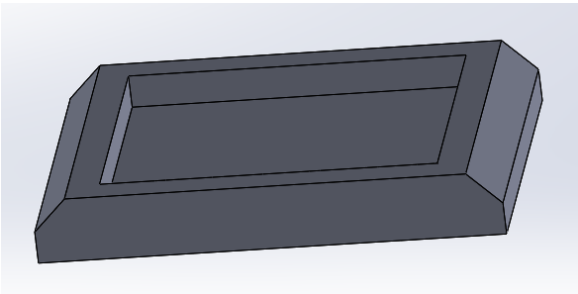
4.1.1 Die Buchstaben

Alle Buchstaben haben für den Roboter zum Hochnehmen einen Quader in der Mitte oben des Bauteils und einen Fuß damit sie stehen können. Die Größe von Quader und Fuß müssen für alle Buchstaben einheitlich sein, damit der Roboter sie problemlos in die Boxen einsortieren kann. Zuerst wurde jeweils der Buchstabe erstellt. Dieser hat immer eine Höhe von 70 mm. Die Skizze wurde dann als Feature linear ausgetragen, mit einer Breite von 15 mm. Das nächste Feature war der Fuß. Hierfür wurde erst eine Skizze unter dem Buchstaben eingefügt und der Fuß mit einer Länge von 50 mm und eine Breite von 40 mm gezeichnet. Beim U und beim O hat der Fuß eine Höhe von 60 mm und bei den anderen eine Höhe von 50 mm, damit die Bögen von U und O nicht in der Luft schweben, sondern etwas im Fuß versinken. Beim U musste ein kleiner Balken oben quer hinzugefügt werden, damit der Quader nicht in der Luft schwebt. Das letzte Feature, das erstellt wurde, waren die Quader zum Hochheben der Buchstaben. Diese sind immer mittig über dem Buchstaben und 20 mm hoch. Sie wind genauso breit wie der Buchstabe.



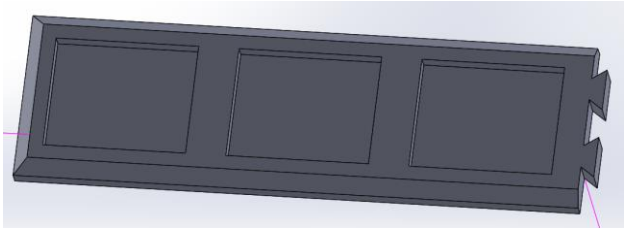
4.1.2 Die Boxen

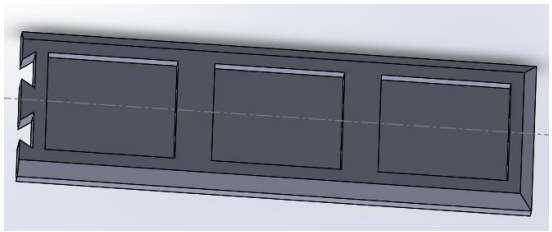
Zuerst wurde die Box mit einer Länge 70 mm und einer Breite 50 mm gezeichnet und im Anschluss mit einer Höhe von 10 mm Linear ausgetragen. Mittig über die Box kam dann ein linear ausgetragener Schnitt mit einer Tiefen von 5 mm, einer Breite von 50,5 mm und einer Länge von 40,5 mm. Links und rechts von der Box kam noch eine Fase mit einem Winkel von 45° und einer Höhe von 5 mm.



4.1.3 Die Platte

Da die Glasplatte vom 3D-Drucker nur eine Länge und Breite von 200 mm mal 200 mm hat und die Platte für die Buchstaben mindestens Sechs mal 50 mm lang sein muss, wurde die Platte auf zweimal gedruckt und mit einer Schwalbenschwanzverbindung zusammengebaut. Zuerst wurde jeweils die Grundfläche mit der Schwalbenschwanzverbindung mit einer Länge von 20 mm und einer Breite von 60,5 mm gezeichnet. Diese wurde dann linear mit einer Höhe von 10 mm ausgetragen. Der zweite Schritt waren die linear ausgetragenen Schnitte mit einer Tiefe von 5 mm, einer Länge von 50,5 mm und einer Breite von 40,5 mm. Der letzte Schritt waren dann die Phasen einmal rum von 45° und 5 mm Höhe.





4.1.4 Der Druck

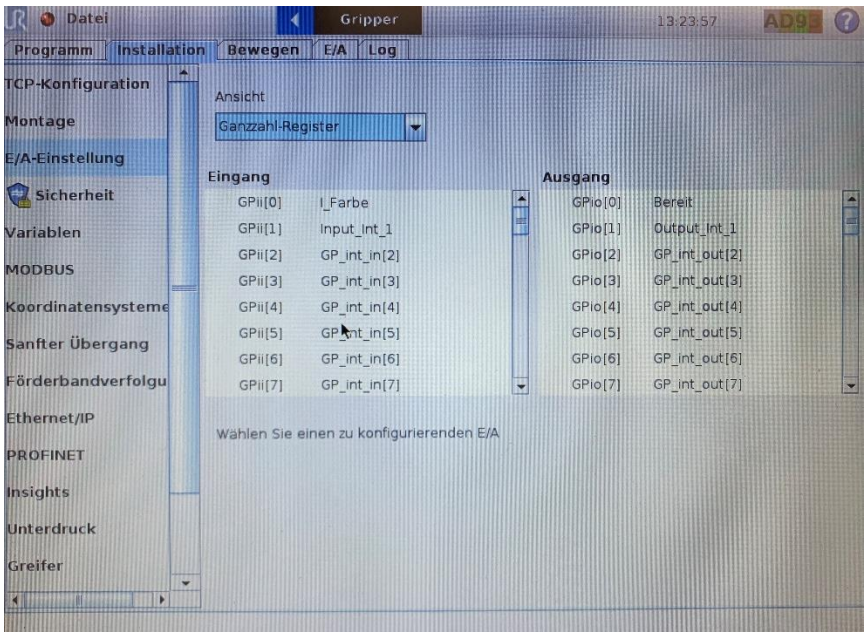
Nachdem alle Bauteile in SolidWorks gezeichnet wurden konnten sie als .STL-Datei abgelegt werden, damit das Programm Ultimaker Cura die Zeichnungen umsetzen kann. Die Standard-Einstellungen wurden übernommen, bis auf die Temperatur, diese steht auf jeder Filament Spule. Außerdem wurde die Stützstruktur aktiviert, aber nur von der Glasplatte aus, da man sonst später große Schwierigkeiten hat, die Stützstruktur wieder zu entfernen. Der nächste Schritt ist „Slicing“ anzuklicken, damit das Programm die Datei als .gcode-Datei auf die vorher eingelegte Speicherkarte speichert. Diese kann man dann in den 3D-Drucker legen und auf Print klicken. Als die Bauteile fertig waren, konnten diese entnommen und entgratet werden.

4.2 Roboterprogramm

Strukturplan Roboterprogramm	Kommentare
Programm Roboterprogramm Schrittnummer:=1 x:=1 Schleife 12 Mal Switch Schrittnummer Fall 1 Switch x Fall 1 FahreAchse Foto1 Einstellen Bereit=1 Warten I_Farbe≠0 FahreAchse ÜberT1 Greifer aktivieren Greiferbewegung50% (1) Greifen1 Greifer aktivieren Greiferbewegung91% (1) FahreLinear Anheben1 SafePos1 Fall 2 FahreAchse Foto2 Einstellen Bereit=1 Warten I_Farbe≠0 FahreAchse ÜberT2 Greifer aktivieren Greiferbewegung50% (1) Greifen2 Greifer aktivieren Greiferbewegung91% (1) FahreLinear Anheben2 SafePos2 Fall 3 FahreAchse Foto3 Einstellen Bereit=1 Warten I_Farbe≠0 FahreAchse ÜberT3 Greifer aktivieren Greiferbewegung50% (1) Greifen3 Greifer aktivieren Greiferbewegung91% (1) FahreLinear	Start des Roboterprogramms. Schrittnummer für die Schrittkette wird auf „1“ für Fall 1 gesetzt. Schleife 12 mal: 6 Buchstaben anschauen und einsortieren Switch Schrittnummer für die Fallentscheidung Fall 1: Erster Buchstabe wird bearbeitet • Foto 1: Roboter fährt zur Position in der das Bild für das KI_Programm gemacht wird • Einstellen Bereit=1: Roboter sendet eine „1“ an das KI-Programm, dass die Position erreicht ist • Warten I_Farbe ist nicht =0: KI-Programm hat an der Roboter einen Integerwert ungleich 0 mit der gesehenen Farbe gesendet • Über T1:Position über halb Buchstabe 1 • Greifer um die Hälfte zu fahren • Teil 1 greifen • Anheben: Buchstabe 1 geradlinig nach oben heben • SafePos: In sichere Position fahren Fall 2: siehe Fall 1; Alle Positionen gleich nur um 3cm verschoben

Anheben3 SafePos3 Fall 4 FahreAchse Foto4 Einstellen Bereit=1 Warten I_Farbe≠0 FahreAchse ÜberT4 Greifer aktivieren Greiferbewegung50% (1) Greifen4 Greifer aktivieren Greiferbewegung91% (1) FahreLinear Anheben4 SafePos4 Fall 5 FahreAchse Foto5 Einstellen Bereit=1 Warten I_Farbe≠0 FahreAchse ÜberT5 Greifer aktivieren Greiferbewegung50% (1) Greifen5 Greifer aktivieren Greiferbewegung91% (1) FahreLinear Anheben5 SafePos5 Fall 6 FahreAchse Foto6 Einstellen Bereit=1 Warten I_Farbe≠0 FahreAchse ÜberT6 Greifer aktivieren Greiferbewegung50% (1) Greifen6 Greifer aktivieren Greiferbewegung91% (1) FahreLinear Anheben6 SafePos6 Schrittnummer:=2 x:=x+1 Fall 2 Switch I_Farbe Fall 1 FahreAchse	Nach jedem Fall wird die Schrittnummer auf 2 gesetzt, um den Vorgang für die Einsortierung zu starten. Das „x“ wird um eins hoch gezählt, um innerhalb des ersten Switches die Fälle abzuarbeiten. Fall 2: • Je nach dem welcher Integerwert (Farbe) geschickt wurde, springt der der Switch I_Farbe in den richtigen Fall zur Einsortierung.

Box1 Greifer offen (1) Einstellen Bereit=0 Fall 2 FahreAchse Box2 Greifer offen (1) Einstellen Bereit=0 Fall 3 FahreAchse Box3 Greifer offen (1) Einstellen Bereit=0 Fall 4 FahreAchse Box4 Greifer offen (1) Einstellen Bereit=0 Fall 5 FahreAchse Box5 Greifer offen (1) Einstellen Bereit=0 Fall 6 FahreAchse Box6 Greifer offen (1) Einstellen Bereit=0 Schrittnummer:=1 Einstellen Bereit=99	• Box 1: Position über Box 1 zur Einsortierung • Greifer öffnen • Bereit =0: Wert an das KI-Programm, dass die Einsortierung des Buchstaben abgeschlossen ist Die folgenden Fälle sind wie Fall 1, nur um 6 cm verschoben. Schrittnummer wird wieder auf „1“ gesetzt, um in den ersten Switch zu springen und den nächsten Buchstaben abzuholen. Bereit=99: Wert für das KI-Programm, dass das der Programmablauf abgeschlossen ist und alle Buchstaben einsortiert wurden.



Die Farbe des Buchstaben, die das KI-Programm ermittelt hat wird als Ganzzahl „I_Farbe“ an den Eingang GPi[0] des Roboters übermittelt.

Bereitschaft/Warten: 0

Grau: 1

Rot: 2

Grün: 3

Gelb: 4

Blau: 5

Weiß: 6

5 KI-Umsetzung

5.1 Grundkonzept

Ergebnis an den Roboter weitergegeben. Im Folgenden wird das Programm im genaueren beschrieben.

5.2 Verwendete Bibliotheken

```
import urllib.request
import cv2
import numpy as np
import rtde.rtde as rtde
import rtde.rtde_config as rtde_config
import sys
```

Als Bibliotheken verwenden wir folgende:

- Urllib – Um das Kamerabild vom Webserver des Roboters zu erlangen
- OpenCV und Numpy – für die Bildverarbeitung und -auswertung
- Real-time Data Exchange (RTDE) – Für die sonstige Kommunikation mit dem Roboter
- Sys – eine Python Standardbibliothek, welche wir für den Programmabbruch verwenden

Im Folgenden wird etwas genauer auf OpenCV, Numpy und RTDE eingegangen.

5.2.1 OpenCV

OpenCV (Open Source Computer Vision Library) ist eine Open Source Bibliothek, die für Computer Vision und Maschinelles Lernen entwickelt wurde. Sie besitzt über 2500 verschiedene Algorithmen, die für Gesichtserkennung, Objekterkennung, Objektverfolgung bis hin zu Interpretation von Gesten eingesetzt werden können.

Die Bibliothek wurde intern bei Intel ab 1998 entwickeln und 2000 erstmals veröffentlicht. Seit 2012 unterliegt sie der BSD Lizenz und ist für kostenlos für den akademischen und kommerziellen Bereich nutzbar. Sie kann in C++, Java, MATLAB und Python eingebunden werden und auf jedem Betriebssystem einsetzbar. Mit der Integration von CUDA und OpenCL wurde die Rechenleistung der Bibliothek gesteigert, da sie nun durch eine Grafikkarte bei ihren Berechnungen unterstützt wird.[1]

5.2.2 Numpy

NumPy ist ein Open Source Projekt, das die mathematischen Funktionen von Python und das Arbeiten mit numerischen Arrays stark über die Grundfunktionen von Python selbst erweitert. Weil OpenCV bei

[1] https://opencv.org/

Verarbeiten von Bildmaterial sich auf Algorithmen für Berechnung von NumPy bezieht, benötigen wir auch dieses in unseren Projekten.[2]

5.2.3 Real-Time Data Exchange

Die RTDE-Schnittstelle (Real-Time Data Exchange) bietet eine Möglichkeit, externe Anwendungen über eine Standard-TCP/IP-Verbindung mit der UR-Steuerung zu synchronisieren, ohne die Echtzeiteigenschaften der UR-Steuerung zu beeinträchtigen. Diese Funktionalität ist unter anderem nützlich für die Interaktion mit Feldbustreibern (z. B. Ethernet/IP), die Manipulation von Roboter-E/A und das Plotten des Roboterstatus (z. B. Robotertrajektorien). Die RTDE-Schnittstelle ist standardmäßig verfügbar, wenn die UR-Steuerung in Betrieb ist. Wir verwenden dieses das Ergebnis der Bildverarbeitung an den Roboter zu senden und von ihm seinen Status zu bekommen.[3]

5.3 Programmablauf

5.3.1 Deklaration einiger Variablen

Bei Start werden zunächst einige Standardvariablen deklariert, welche je nach Setup angepasst werden müssen.

```
# Inizialisierung
# Variablen Deklarieren
hostname = "10.1.91.167"
ROBOT_PORT = 30004
config_filename = 'control_loop_configura-
tion.xml'
DEBUG = True
```

- Hostname – die IP-Adresse des Roboters
- ROBOT_PORT – Der RTDE Port des Roboters
- Config_filename – der Pfad des Konfigurationsfile der RTDE Schnittstelle
- DEBUG – Wenn diese Variable True ist werden während der Ausführung des Programms immer wieder ausgaben auf der Konsole getätigt

5.3.2 RTDE Kommunikation aufbauen

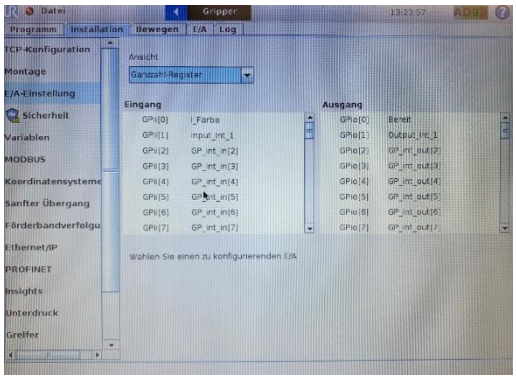
Um mit dem Roboter mit RTDE zu kommunizieren, muss zunächst ein Konfigurationsfile angelegt werden in Form einer XML-Datei. Diese sieht bei uns wie rechts zu sehen ist aus. Die Grundnode

```
<?xml version="1.0"?>
<rtde_config>
    <recipe key="input_vals">
        <field name="input_int_register_0" type="INT32"/>
    </recipe>
    <recipe key="output_vals">
        <field name="output_int_register_0" type="INT32"/>
    </recipe>
</rtde_config>
```

[2] https://numpy.org/
[3] https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/

<rtde_config> muss immer vorhanden sein und die Nodes <recipe> (im folgenden Rezept genannt) werden je nach nutzen angelegt und angepasst.

Jedes Rezept enthält eine beliebige Anzahl an Variablen, jedoch entweder Eingänge oder Ausgänge, und hat einen eindeutigen Key, um dieses zu identifizieren. Jede Variable braucht auch einen eindeutigen Namen und ein Datentyp Zuweisung, welche Datentypen es gibt kann aus der Dokumentation von Universal Robots entnommen werden (Fußnote 3). Auf der Website können auch Beispiel Programme zu RTDE heruntergeladen werden.
Wir haben zwei Rezepte erstellt mit je einer Integer Variabel. Das Rezept „input_vals“ sind die Input Variablen des Roboters, welche wir an ihn senden. Hier haben wir eine Variable deklariert, welche für die Übertragung der ermittelten Farbe vorgesehen ist. Mit der Variabel im Rezept „output_vals“ empfangen wir Daten vom Roboter. Mit dieser Variabel erfahren wir, ob der Roboter gerade bereitsteht für die Bildauswertung oder ob er alle Buchstaben einsortiert hat.



Ist diese Datei konfiguriert, müssen die Variablen auch im Roboter angelegt sein. Dies geschieht bei den E/A-Einstellungen (sieh Bild), nur wenn sie hier angelegt sind können sie auch im Programmablauf des Roboters verwendet werden.

Gehen wir zurück zu unseren Python-Skript. Hier wird nun die Konfigurationsdatei geladen und die einzelnen Rezepte werden aus dieser ausgelesen.

```
# Kommunikation mit Roboter herstellen und Konfigurieren
# Konfigurationsfile für Kommunikation laden
conf = rtde_config.ConfigFile(config_filename)
input_names, input_types = conf.get_recipe('input_vals')
output_names, output_types = conf.get_recipe('output_vals')
```

Daraufhin wird sich ein neuer RTDE-Client erstellt und sich auf diesen verbunden. Dafür verwenden wir die bereits festgelegte IP-Adresse sowie Port.

```
# RTDE Client erstellen und verbinden
con = rtde.RTDE(hostname, ROBOT_PORT)
con.connect()
```

Nun müssen wir dem RTDE-Interface des Roboters unsere Rezepte mitteilen.

```
# setup recipes
input_vals = con.send_input_setup(input_names, input_types)
output_vals = con.send_output_setup(output_names, output_types)
```

Danach belegen wir noch alle Variablen, welche wir zum Robotersenden mit einem Default Wert und Starten daraufhin die Kommunikation. Sollte dies nicht funktionieren wird das Programm hier abgebrochen.

```
# Variablen Initialisieren
input_vals.input_int_register_0 = 0

#start data synchronization
if not con.send_start():
    sys.exit()
```

5.3.3 Main-Loop

Nun wird der Main-Loop des Programmes gestartet, diesen beschreiben wir nun im folgenden:

```
# Main Loop
running = True
letter_detected = False
while running:
    output_vals = con.receive()
    if output_vals is None:
        break
```

Hier wird zunächst eine Laufvariable „running“ deklariert, welche den Loop am Laufen hält und eine Variable „letter_detected“ welche nach dem erkennen einen Buchstaben gesetzt wird und die erneute Erkennung verhindert.

Als nächstes wird zunächst einmal der Status des Programms abgebrochen. Werden keine Ausgangsvariablen empfangen wird das Programm abgebrochen. Dies passiert z.B., wenn man den Roboter während des Ablaufes ausschaltet.

```
# Main Loop
running = True
letter_detected = False
while running:
    output_vals = con.receive()
    if output_vals is None:
        break

    # Warte darauf das der Roboterbereit steht
    while output_vals.output_int_register_0 == 0:
        # Wenn RB nicht bereit Inputdaten rückset-
          zen und senden
        input_vals.input_int_register_0 = 0
        if DEBUG:
            print(input_vals.input_int_register_0)
        letter_detected = False
        con.send(input_vals)

        # kurz warten und dann Daten vom Roboter
          erneut überprüfen
```

Nun startet ein weiterer Loop, innerhalb des Main-Loop. In dieser läuft so lange, bis der Roboter nicht den Status 0 sendet. Wenn er diesen Status sendet, wird an ihn auch nur der Status 0 als ermittelte Farbe zurückgegeben und nach einer Wartezeit von 500ms wird der Status des Roboters erneut abgefragt. Auch hier wird die gleiche Abbruchbedingung, wie schon vorher.

```
        cv2.waitKey(500)
        output_vals = con.receive()
        # Wenn keine Daten empfangen abbrechen
        if output_vals is None:
            running = False
            break
```

```
# Sobald Roboter bereitsteht -> Buchstaben erkennen
    if output_vals.output_int_register_0 == 1 and let-
    ter_detected == False:
        while True:
            feedback = [99, 99, 99]
            for i in range(3):
                image = get_image()
                feedback[i] = get_letter_from_image(image)
            if feedback[0] == feedback[1] == feedback[2]:
                break
        input_vals.input_int_register_0 = feedback[0]
        letter_detected = True
    # Programm abbrechen
    elif output_vals.output_int_register_0 == 99:
        input_vals.input_int_register_0 = 0
        con.send(input_vals)
        break

    con.send(input_vals)
```

Wenn als Status keine 0 gesendet wird, wird der Loop abgebrochen und es kommt zur linksstehenden if-Abfrage.

Wird ein Status von 1 empfangen heißt, dass das der Roboter bereitsteht für die Bildauswertung und diese wird ausgeführt und das erkannte an ihn zurück gesendet (siehe nächstes Kapitel).

Wird der Status 99 empfangen heißt das das der Roboter alle Buchstaben sortiert hat und der Main-Loop wird abgebrochen.

Nach der If-Abfrage wird noch der aktuelle Status der Eingänge des Roberters gesendet.

```
# RTDE Client Verbindung trennen
con.disconnect()
```

Nach dem Main-Loop wird noch die Verbindung des RTDE-Clients beendet.

5.3.4 Bildauswertung

```
while True:
    feedback = [99, 99, 99]
    for i in range(3):
        image = get_image()
        feedback[i] = get_letter_from_image(image)
    if feedback[0] == feedback[1] == feedback[2]:
        break
input_vals.input_detected_color = feedback[0]
letter_detected = True
```

Bei der Bildauswertung wird zunächst eine Liste mit drei Elementen erzeugt. Nun wird dreimal ein Bild von der Kamera empfangen, dieses ausgewertet und ein eines der Elemente der Liste geschrieben. Dies passiert so lange, bis alle Elemente der Liste gleich sind, damit das Bild eindeutig Identifiziert wurde.

Danach wird der erkannte Wert auf den Eingang des Roboter geschrieben, damit dieser später gesendet wird und die Variable letter_detected wird gesetzt, damit dieser Ablauf bis zum nächsten Status 1 des Roboters nicht noch einmal ausgeführt wird.

5.3.4.1 Bild aufnehmen

```
# Kamerabild auslesen
def get_image():
    global hostname
    cv2.waitKey(1) # Verzögerung, damit Kamera den Fokus findet (nur 1ms evtl.
                   unnötig)
    resp = urllib.request.urlopen(f"http://{hostname}:4242/cur
    rent.jpg?type=color")
    image = np.asarray(bytearray(resp.read()), dtype="uint8")
    image = cv2.imdecode(image, cv2.IMREAD_COLOR)
    return image
```

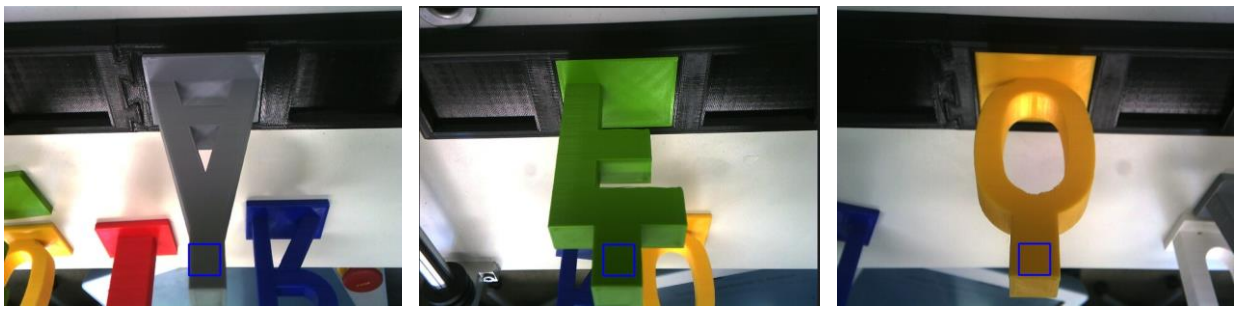
Mit der Funktion get_image() wird das Bild von der Kamera abgerufen und in einer Variable gespeichert. Hierfür wird sich auf den HTTP-Server, dieser Verbunden und der Empfangen Inhalt wird in einer Variablen gespeichert. Dieser Inhalt wird nun in ein Numpy Array umgewandelt und noch zu einem OpenCV Bild dekodiert. Ist dies geschehen wird das fertige Bild von der Funktion zurückgegeben.

5.3.4.2 Bildausschnitt herausfiltern

```
# Buchstaben aus Bild erkennen
def get_letter_from_image(image):
    # Bild zu HSV-Format umwandeln
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Bereich der geprüft werden soll extrahieren
    height, widgth, _ = image.shape
    squareToCheck = hsv[height - 95: height - 55, int(widgth / 2 - 20): int(widgth / 2 +
20), :]
    return get_color_of_area(squareToCheck)
```

Nachdem das Bild „aufgenommen“ wurde, wird dieses ausgewertet, dazu wird es an die Funktion get_letter_from_image übergeben, dieses wandelt es zunächst vom BGR-Format ins HSV-Format um. Daraufhin schneidet die Funktion aus dem Bild einen festdefinierten Bereich heraus, welche immer an der Halterung des Buchstaben ist. Dieser Bereich entspricht den blaue Umrahmten Bereich, welcher in den Bildern auf der nächsten Seite zu sehen ist.



Der extrahierte Bereich wird an eine weiter Funktion übergeben, welche diesen dann auswertet.

5.3.4.3 Farbe erkennen

```
# Durschschnitts HSV-Werte des Bildbereichs ermitteln
    sum = [0, 0, 0]
    posibility = 0
    numberOfPixelToCheck = 0
    for line in area:
        for pixel in line:
            sum[0] += pixel[0]
            sum[1] += pixel[1]
            sum[2] += pixel[2]
            numberOfPixelToCheck += 1

            if pixel[0] < 10 or pixel[1] > 330:
                posibility += 1

    average = [sum[0] / numberOfPixelToCheck, sum[1] / numberOfPixelToCheck,
    sum[2] / numberOfPixelToCheck]
    posibility_red = posibility / numberOfPixelToCheck
```

In der Funktion get_color_of_area wird zunächst für jeden Pixel des übergeben Bildbereichs ein Durchschnittswert ermittelt. Hierfür werden der H-, S- und V-Wert jedes Pixel aufsummiert und durch die Anzahl der Pixel geteilt. Dadurch lässt sich in einem recht homogenen Farbbereich die Durchschnittsfarbe recht leicht ermitteln. Da der HSV-Farbbereich keine Skala mit einem Anfang und einem Ende ist, sondern ein Kegel, bei dem der H-Wert (Farbwert) einen Wert von 0° bis 360° annehmen kann, ist es bei der Farbe Rot, welche im Bereich von ca. 340° bis 10° ist, schwierig sein einen Durchschnittswert zu ermitteln.[4] Hierfür wird zählen wir bloß wieviel Pixel sich in diesem Bereich befinden. Die Anzahl der roten Pixel Teilen wir nun auch durch die Gesamtzahl der Pixel, um eine Wahrscheinlichkeit zu erhalten ob der Bereich rot ist.

```
# Funktion zum umnormieren des Durchschnittwertes (H - 360, S - 100, V - 100)
    def normalize(value, value_max, norm_max):
        return value * norm_max / value_max
```

[4] https://de.wikipedia.org/wiki/HSV-Farbraum

Da wird durch die Durchschnittswert Ermittlung nur Werte zwischen 0 und 255 bekommen (uint8-Werte), die Werte des HSV-Bereiches jedoch folgende Werte annehmen können:

	Bedeutung	minimal	maximal
H	Farbwert	0°	360°
S	Farbsättigung	0 %	100 %
V	Hellwert	0 %	100 %

Haben wir hier noch eine Funktion zur Normalisierung der Werte auf diese Werte eingefügt. Damit es einfach ist, wenn man diese Funktion erweitern möchte.

```
# Mögliche Rückgaben
dict_letters = {"A":1, "I":2, "F":3, "O":4,"R":5, "U":6, "Error": 99}
dict_colors = {"grau":1, "rot":2, "grün":3, "gelb":4,"blau":5, "weiß":6, "Error": 99}
# Anhand der Durchschnittswerte, Farbe (Buchstabe) erkennen und Ergebnis zurückgeben
    if posibility_red > 0.75:
        if DEBUG:
            print("I")
        return dict_letters["I"]
    elif normalize(60, 360, 255) <= average[0] <= normalize(140, 360, 255):
        if normalize(0, 100, 255) <= average[1] <= normalize(50, 100, 255):
            if DEBUG:
                print("A")
            return dict_letters["A"]
        elif normalize(50, 100, 255) <= average[1] <= normalize(100, 100, 255):
            if DEBUG:
                print("F")
            return dict_letters["F"]
    elif normalize(22, 360, 255) <=  average[0] <= normalize(55, 360, 255):
        if normalize(0, 100, 255) <= average[1] <= normalize(50, 100, 255):
            if DEBUG:
                print("U")
            return dict_letters["U"]
        elif normalize(50, 100, 255) <= average[1] <= normalize(100, 100, 255):
            if DEBUG:
                print("O")
            return dict_letters["O"]
    elif normalize(150, 360, 255) <=  average[0] <= normalize(240, 360, 255):
        if DEBUG:
            print("R")
        return dict_letters["R"]

    # Rückgabe, wenn kein Buchstabe erkannt wurde
    return dict_letters["Error"]
```

Im nächsten Teil der Funktion get_color_of_area werden nun die Durchschnittswerte ausgewertet, je nachdem in welchem Bereich sich die Durchschnittswerte Befinden wird nun ein anderer Wert zurückgegeben. Ist die Wahrscheinlichkeit für rot größer als 75% so wird „rot“ bzw. „I“ zurückgegeben. Sollte der ermittelte Durchschnittswert in keinen Bereich passen wird Fehler (99) zurückgegeben.

5.4 Weitere Dateien

```
01_Robotoerspiel.py: Ausführbares Programm
02_Roboterspiel.ipynb: Ausführbares Programm mit Zusatzinformationen
03_take_photo.ipynb: Programme um auf die Kamera zuzugreifen
04_HSV-Wert_ermitteln.py: Programm, um HSV-Bereiche zu ermitteln
control_loop_configuration.xml: Konfigurationsdatei für RTDE
```

Github: https://github.com/Madipodo/Roboterspiel

6 Ausblick

Als Ausblick/Erweiterungen für die nachfolgenden Teams haben wir folgende Projekte zusammengetragen:

1. Variable Boxenreihenfolge. Dazu müsste man eine Vorrichtung bauen/drucken um die Boxen variable durch Tauschen zu können und trotzdem immer die Position einhalten zu können. Weiterhin muss man, dass Roboterprogramm, sowie das KI-Programm so anpassen, dass die jeweils neue Reihenfolge der Boxen vom Roboter vor dem Beginn der Sortierung erkannt wird.
2. Die Buchstaben unabhängig von der Vorrichtung erkennen und greifen können. Dazu gehört die exakte Bestimmung der Position und Lage der Buchstaben und die Übermittlung der Daten an den Roboterarm für die perfekte Position, um den Buchstaben zu greifen.
3. Das zurück sortieren des Wortes „AIFORU“ in richtiger Reihenfolge zurück in die Ausgangsvorrichtung.

Auch kann man das Projekt in der aktuellen Form noch optimieren, der Buchstabe O ist etwas zu klein geraten. Zudem sollte die schwarze Grundplattform noch einmal gedruckt werden, da diese beim ersten Mal mit einer hohen Geschwindigkeit gedruckt wurde, wodurch sie nicht sehr passgenau ist.

7 Fazit

Als Fazit des Projektes Roboterspiel im Fach Künstliche Intelligenz bleiben ausschließlich positive Erfahrungen in Erinnerung. Mit dem Zusammenspiel aus Hardwarebeschaffung im Sinne des 3D-Druckes der Buchstaben, der Programmierung des Roboters und auch die KI im Python Programm ließ jedem im Team neue Fähigkeiten erwerben. Neben dem Zeichnen der Buchstaben im CAD Programm lernte man auch die Vor- und Nachteile, sowie die Eigenschaften und Umgang mit dem Drucker genauer kennen. Beim Umsetzen des Roboterprogramms setzten wir uns mit den verschiedenen Möglichkeiten, Daten mit einem Python Programm auszutauschen auseinander. Wir lernten verschiedenen Verfahr- und Strukturabläufe im Roboter kennen und einzusetzen. Außerdem führten wir einige Updates durch und ermöglichten dadurch unter anderem den Remote Zugriff. Bei der Umsetzung der künstlichen Intelligenz nutzten wir zunächst die Teachable Machine, um die Erkennung der Buchstaben zu gewährleisten. Durch die eher schlechten Lichtverhältnisse und der verzögernden Fokussierung der Kamera, entschlossen wir uns im zweiten Anlauf auf eine eigene Farb- und Kontrasterkennung. Das KI-Projekt Roboterspiel ist im aktuellen Zustand in der Lage die Buchstaben von unterschiedlicher Ausgangsposition die Farbe zu erkennen und in die richtige Box abzulegen. Für uns persönlich war das Projekt damit ein voller Erfolg und unser Ziel ein, grundsolides Projekt für weitere Gruppen zu liefern, haben wir erreicht. Im Abschnitt Ausblick haben wir unsere Idee für die Erweiterung niedergeschrieben. Alle Teile des Projektes sind auf eine leichte Weiterentwicklung abgestimmt.

8 Anhang

- CAD-Files für alle Buchstaben und Boxen (insgesamt 9)
- Python-Skript mit fertigem Programm und Konfigurations-XML für RTDE