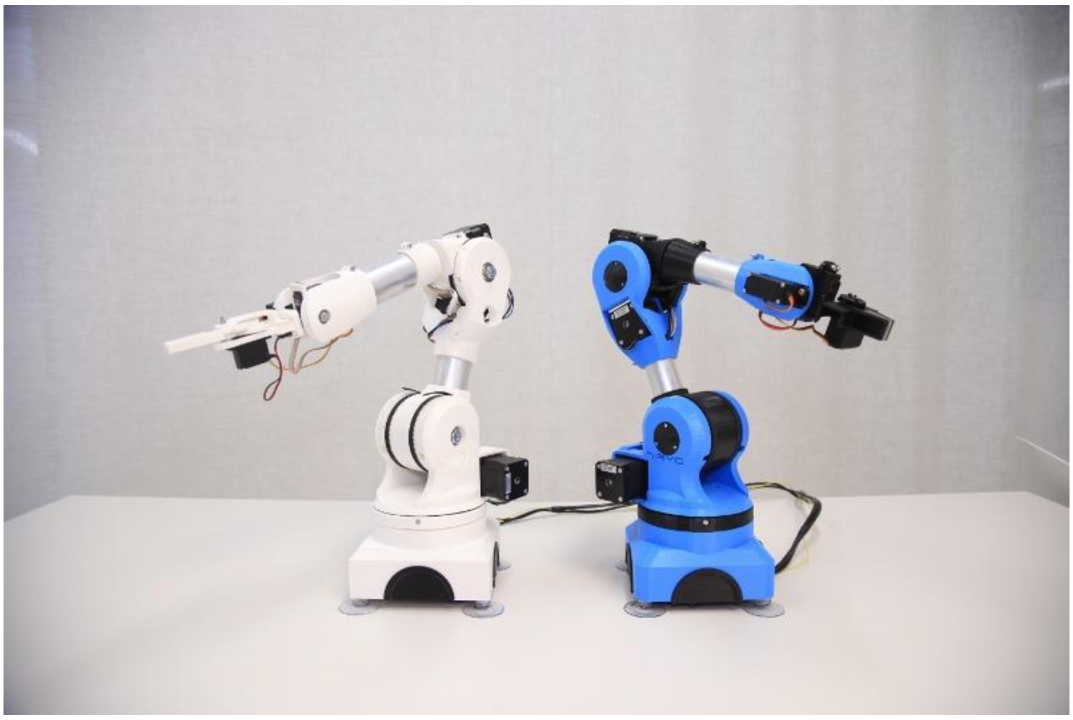
London South Bank University
EST 1892

Robotics Workshop



EEE-7-ROB – Robotics

School of Engineering

2024-2025

Revised by Juan Felipe Proano Vasconez
Electronics Engineering Technician

Contents

EEE_7_ROB: Robotics Laboratory Exercises

1. Introduction

The robotics lab-work includes four exercises that each student should complete. All lab sessions are carried out in the robotics lab (Room T411, located on the fourth floor of the Tower Building). Each lab session is scheduled for two hours. Some of the lab exercises may be completed in one session and some may take longer. You will be working in groups of two students. Each student should record the experimental work carried out with associated results and discussion in a neat logbook. Submit the completed logbook in electronic format of a single pdf file via Moodle VLE by 15:00 on Friday of the 11th week of the semester for assessment. Students with DDS needs are given an extra week; the deadline for submission of their logbook will be 15:00 on Friday of 12th week of the semester.

The lab exercises are based around the Niryo 1 Robot (six axis robot arm) designed for education and research purposes. You will learn how to calibrate and program the robot for carrying out certain tasks. The Python programming language is used for programming the robot. Thus, to gain the most from the lab exercises the first two weeks of the semester are allocated to familiarisation with Python and the Integrated Development Environment (IDE) PyCharm. You may complete this task either in the lab or elsewhere using your own computer.

To write programs in Python one will need a development environment. Although one can work with the basic IDLE (Integrated Development and Learning Environment) that comes with a basic Python installation, developers are encouraged to use IDEs such as PyCharm for better software development. The IDEs will allow you to develop and run Python programs. To familiarise yourself with Python programming language, install Python (https://www.python.org/downloads/) and PyCharm (https://www.jetbrains.com/edu-products/download/other-PCE.html) to your computer and learn how to develop and run Python programs. You can find a comprehensive set of tutorials in the following website, which will help you familiarise yourself with the language and its syntax.

Python 3 Tutorial official python site

Niryo One Python commands

LinkedIn Learning Python course

Niryo One Manual and Mechanical Specifications

You will also find detailed documentation on the Python language in the above site. There are also a range of books available on Python programming.

1.1. Safety Precautions

It is forbidden to have open drinks in the lab such as cans and paper coffee cups. You are allowed to have drinks with a secured closed lid on the bench during the lab sessions. This is to prevent spillage and damage to the equipment.

1.2. Powering the robot

The robots will be turned on by the Technician and there is no need to follow the power on and off procedures. The steps detailed below are only for familiarisation with the procedure. **DO NOT** turn the robots on or off.

1.3. Robot angles programming

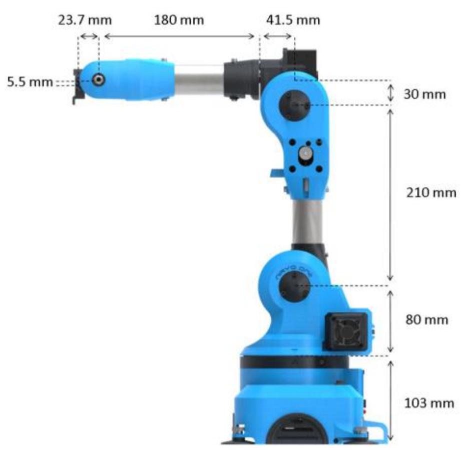
The experiments have been programmed with tested angles to avoid the robot hitting the table, the user or other equipment on the bench. **DO NOT** enter angles which are not specified in the experiments. For example, if the experiments suggest doing increments of 0.1 do not try increments of 0.5 or 1 unless suggested in the experiment. This will cause the robot to do an unpredicted angle which can cause harm or robot damage. The maximum angle in radians for all joint is 3+- radians, not all the joint are able to travel the full range of the 3 degree radians due to mechanical limitations refer to Mechanical Specifications

1.4. Robot IP connection

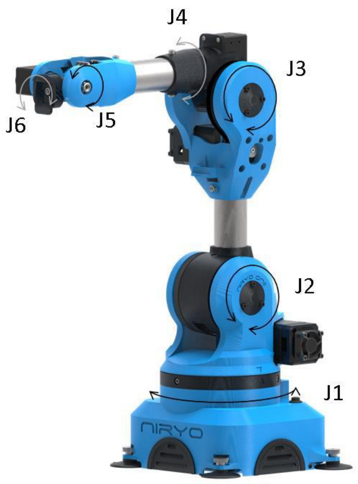
All robots are connected via IP address which is written at the front of each robot. **DO NOT** try to connect to a robot which is not in front of you, or which is not part of your group. This will interfere with other student's work and can cause harm as the robot can move unexpectedly when another student is taking measurement of the target robot.

1.5. Laptop use

ONLY ONE PC/Laptop can connect to a robot at a time. We will primarily be using the lab PCs. However, Laptops can also connect to the robot following the connection procedure detailed below. **DO NOT** try to connect to a robot which is not in front of you as described in the previous section 1.1.3.

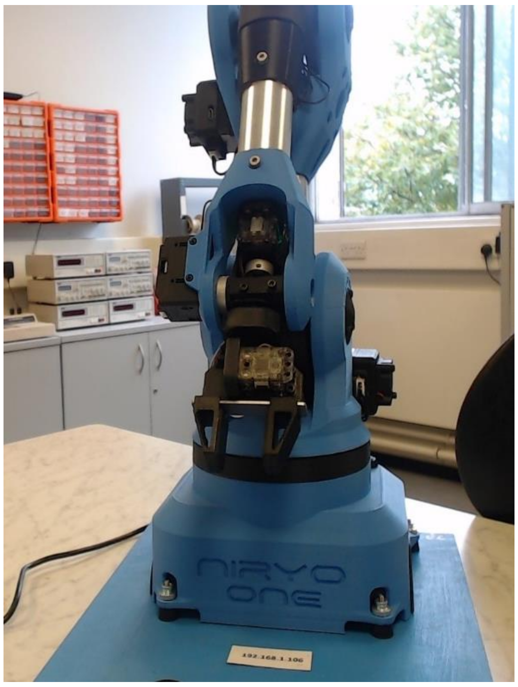


Robot dimensions

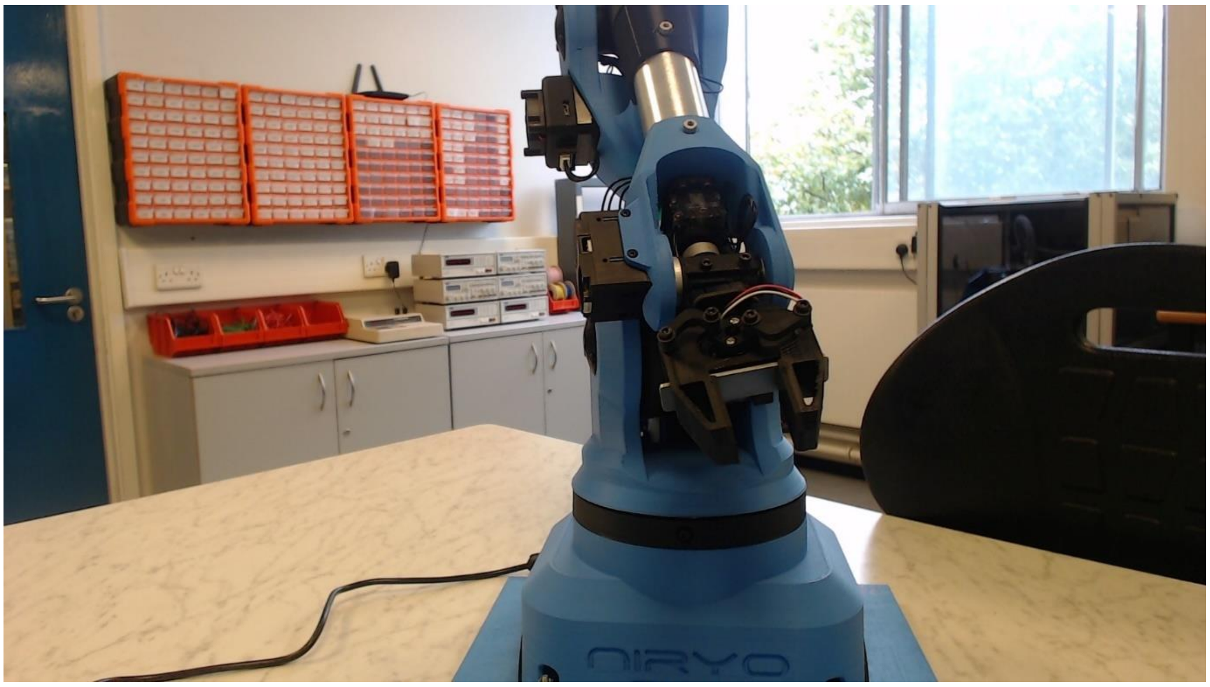


Robot joints

1.6. Correct robot starting position and robot manipulation.



The robot should start and finish at this position. The gripper should be facing towards the user with no tangled wires around the joint. Below an example of what should be avoided when manipulating the J6 axis.



2. Niryo One – Setup

In this laboratory session you will be introduced to Niryo One Robot and how to operate a robot safely. You will be programming the robot in the Python language under the PyCharm IDE.

2.1 How to logon to PCs in T411

Log in with your student email and password.

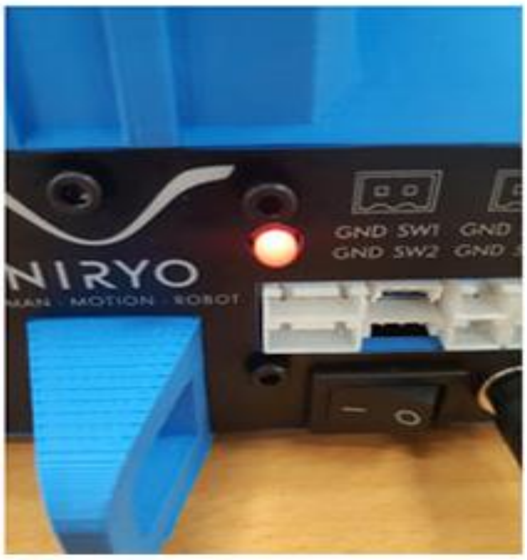
2.2 Power up/shutdown procedure

Ensure that the power cable to the Robot is connected properly at the Robot Base and at the Mains Adapter. ***Accidental disconnection of power can damage the Robot memory***.

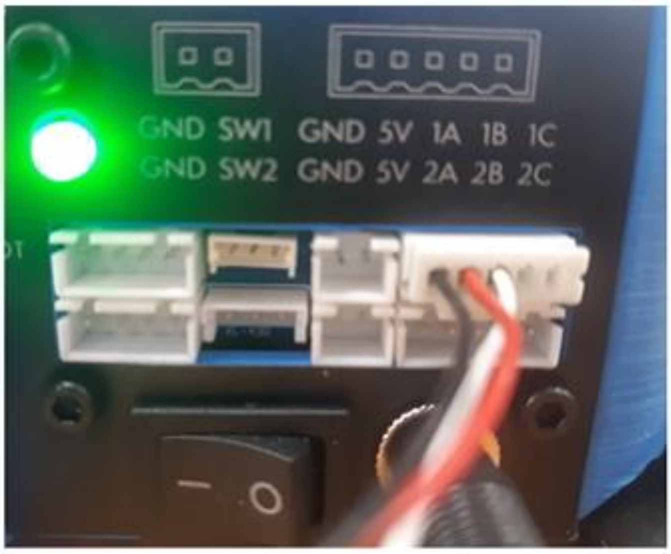
The Push button on the right-hand side of the Base body is used only for the shutdown procedure which will be discussed later in this document. You must not switch the Robot power OFF, without shutting it down first.

2.3 Power Up Procedure

Plug in the Robot's Power Supply (11.1V , 7 Amp) to the Mains and turn ON the power Switch at the back of the Robot. You will see a RED LED light up.



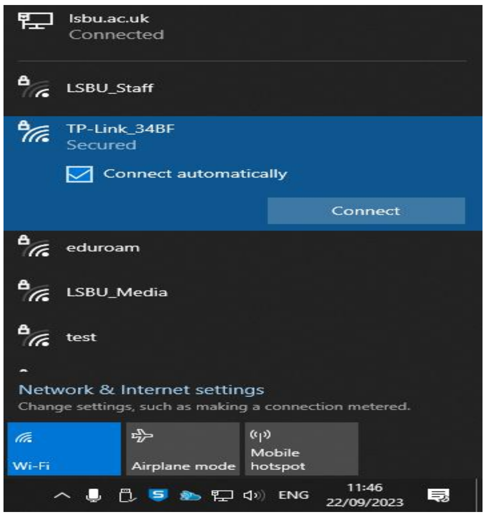
Please wait for the LED to Turn GREEN. ***You must not switch the Robot OFF or interfere with it until this process is completed***. This is the Boot cycle of the Raspberry PI-3 controller within the Robot.



The Robot connects using Wi-Fi to the lab Router automatically, the GREEN light means there is a successful connection to the router. The computers in T-411 are also connected to the same router using a Wi-Fi dongle. The robots can be addressed via PyCharm using the robots own IP address. This address will be entered in your Python program as shown later in the Sample program.

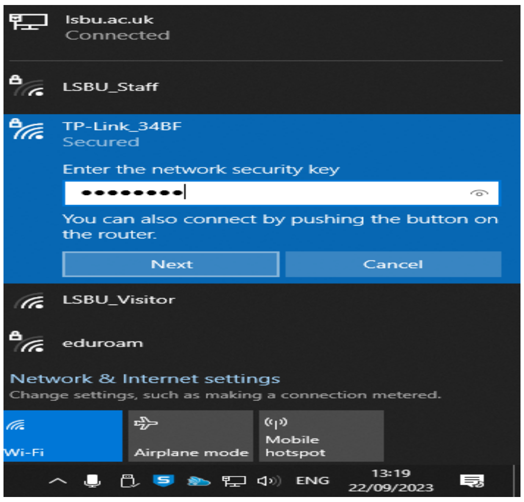
2.4 Wireless Setup

Ensure the LED at the back of the Robot is GREEN before you go to the next step. Next click on the Network ICON ‘square’ in the bottom right hand of the screen.


lsbu.ac.uk
Connected
LSBU_Staff
TP-Link_34BF
Secured
Connect automatically
Connect
eduroam
LSBU_Media
test
Network & Internet settings
Change settings, such as making a connection metered.
Wi-Fi
Airplane mode
Mobile hotspot
ENG
11:46
22/09/2023

Select the **TP-Link_34BF** network. Click to make a connection.

The **Connect Automatically** box should be ticked and then Click ‘Connect’. Wait for the connection, as this may take up to one minute.

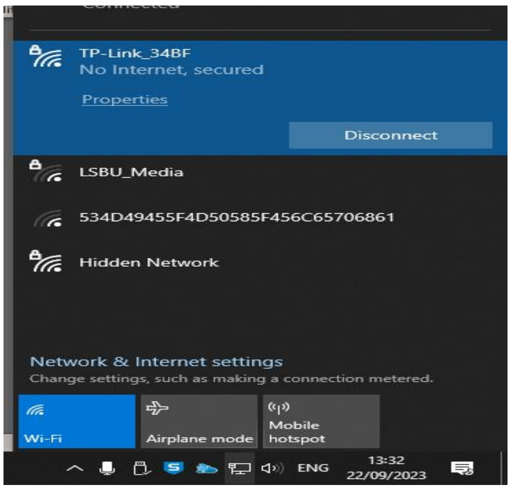

lsbu.ac.uk
Connected
LSBU_Staff
TP-Link_34BF
Secured
Enter the network security key
••••••••
You can also connect by pushing the button on the router.
Next
Cancel
LSBU_Visitor
eduroam
Network & Internet settings
Change settings, such as making a connection metered.
Wi-Fi
Airplane mode
Mobile hotspot
ENG
13:19
22/09/2023

A password will be requested in order to connect to the wireless router.

Enter in the box the following password:

96679708

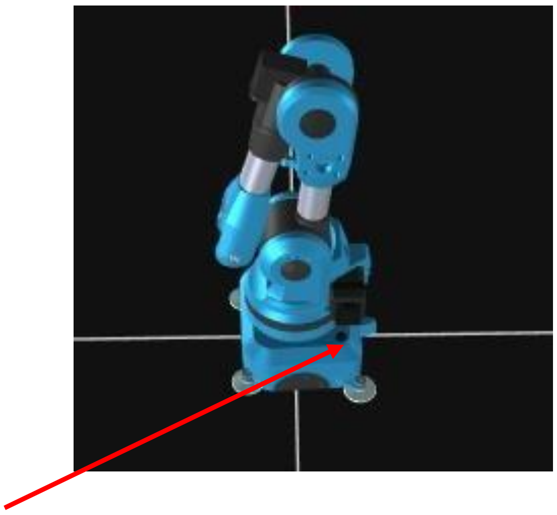
Click next to connect.


TP-Link_34BF
No Internet, secured
Properties
Disconnect
LSBU_Media
534D49455F4D50585F456C65706861
Hidden Network
Network & Internet settings
Change settings, such as making a connection metered.

A successful connection is shown as in this image.

2.5 Shutdown Button

This Button should not be touched by Students. This is a shutdown button and must only be operated by Technicians. **PLEASE NO DOT TOUCH THIS**.



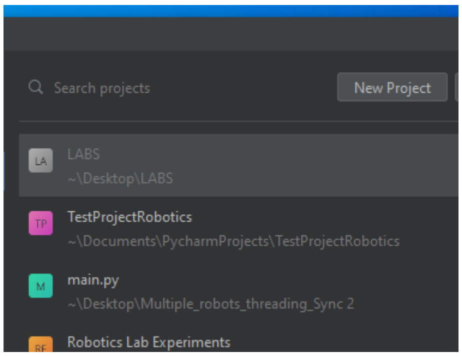
3. Creating a Project in PyCharm using Niryo One with the TCP Client Library

In this lab exercise, we will walk you through the process of creating a project in PyCharm, integrating the Niryo One Python TCP Client library into your project, marking the cloned folder as a sources root, and providing a sample short program. We will assume you already have Python and PyCharm installed on your machine.

3.1. Setting up PyCharm Project

(i) Launch PyCharm

(ii) Click on "New Project".



Search projects

New Project

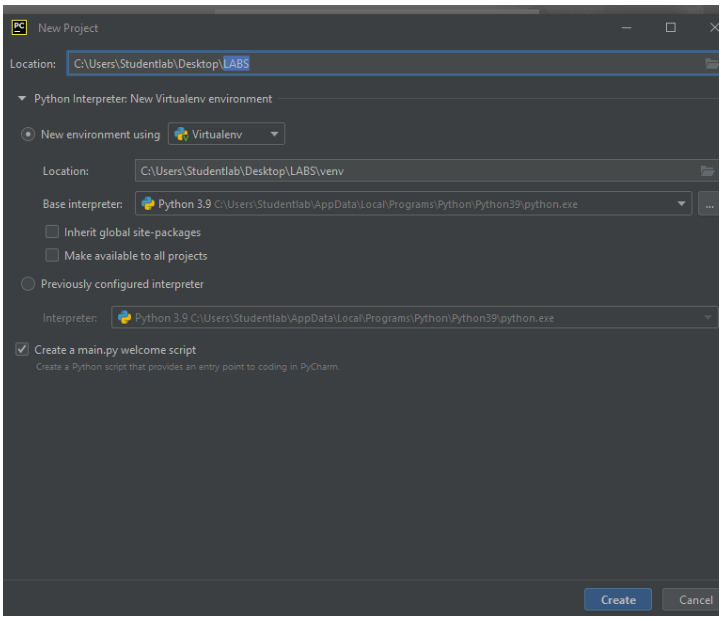
LABS
~\Desktop\LABS

TestProjectRobotics
~\Documents\PycharmProjects\TestProjectRobotics

main.py
~\Desktop\Multiple_robots_threading_Sync 2

Robotics Lab Experiments

(iii) Select the Desktop for your project location and create a folder for your project by adding \LABS as shown in the image.



New Project

Location: C:\Users\Studentlab\Desktop\LABS

Python Interpreter: New Virtualenv environment

(●) New environment using Virtualenv

Location: C:\Users\Studentlab\Desktop\LABS\venv

Base interpreter: Python 3.9 C:\Users\Studentlab\AppData\Local\Programs\Python\Python39\python.exe

☐ Inherit global site-packages

☐ Make available to all projects

( ) Previously configured interpreter

Interpreter: Python 3.9 C:\Users\Studentlab\AppData\Local\Programs\Python\Python39\python.exe

☑ Create a main.py welcome script

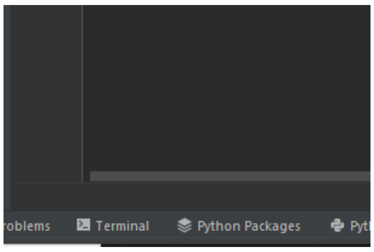
Create a Python script that provides an entry point to coding in PyCharm.

Create Cancel

(iv) Select the Python interpreter you want to use for this project and click on create.

3.2. Installing Niryo One Python TCP Client Library

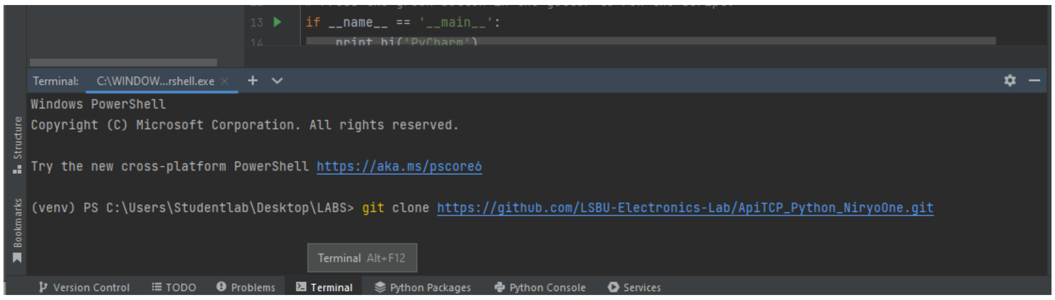
(i) Open the terminal within PyCharm or your system's terminal at the bottom of the Pycharm IDE.



(ii) Clone the Niryo One Python TCP Client library repository from GitHub using the terminal in PyCharm with the following command and press enter:

Copy the following command in the terminal as shown below:

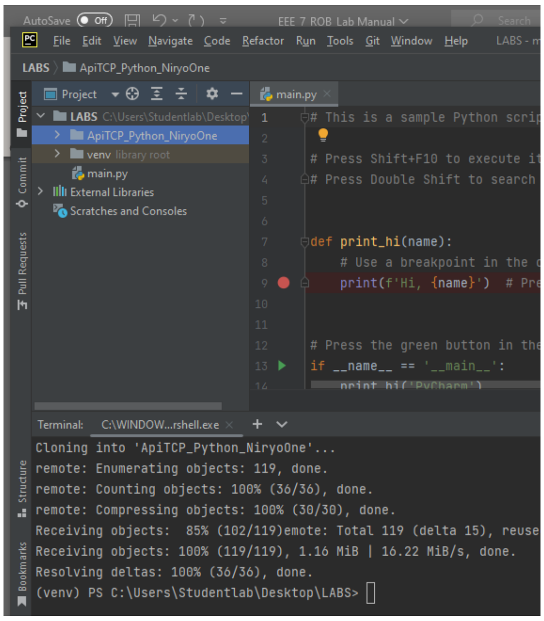
```
git clone https://github.com/LSBU-Electronics-
Lab/ApiTCP_Python_NiryoOne.git
```



```
if __name__ == '__main__':
    print_hi('PyCharm')
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

(venv) PS C:\Users\Studentlab\Desktop\LABS> git clone https://github.com/LSBU-Electronics-Lab/ApiTCP_Python_NiryoOne.git
```

(iv) Click on the main project folder called LABS in the picture below and a new folder should appear within your project called “ApiTCP_Pyhton_NiryoOne” after successful installation.



```
# This is a sample Python scrip

# Press Shift+F10 to execute it
# Press Double Shift to search


def print_hi(name):
    # Use a breakpoint in the c
    print(f'Hi, {name}')  # Pre


# Press the green button in the
if __name__ == '__main__':
    print_hi('PyCharm')
```

```
Cloning into 'ApiTCP_Python_NiryoOne'...
remote: Enumerating objects: 119, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (30/30), done.
Receiving objects:  85% (102/119)emote: Total 119 (delta 15), reuse
Receiving objects: 100% (119/119), 1.16 MiB | 16.22 MiB/s, done.
Resolving deltas: 100% (36/36), done.
(venv) PS C:\Users\Studentlab\Desktop\LABS>
```

3.3. Marking the Cloned Folder as Sources Root

(i) In PyCharm, right-click on the `niryo_one_python_tcp_client` folder in the "Project" view.

(ii) Select "Mark Directory as" and then choose "Sources Root". This step tells PyCharm to treat this folder as a source of Python code.

3.4. Creating a Basic Niryo One TCP Client Program

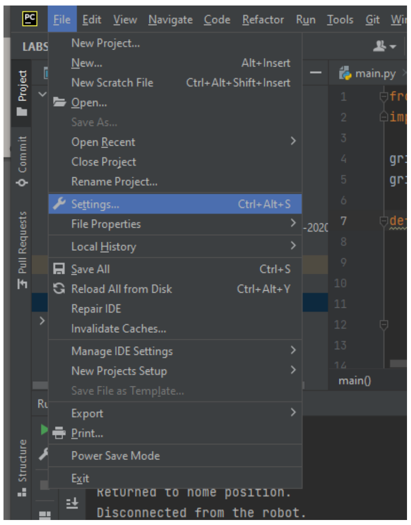
(i) In PyCharm, right click in your project directory named LABS.

(ii) Create a new Python file (e.g., `niryo_tcp_demo.py`) within the project.

(iii) Open the newly created Python file.

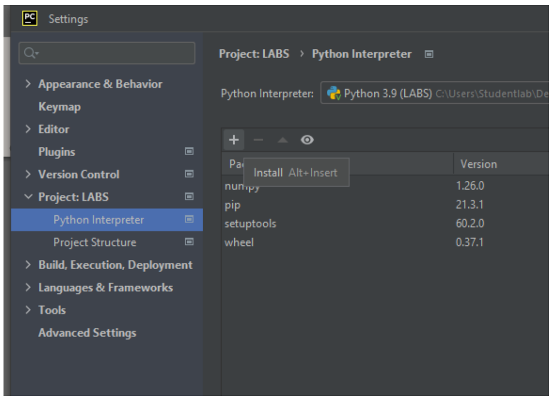
(iv) Add the Sample code to the newly created python file

(v) Click on File -> Settings



```
Returned to home position.
Disconnected from the robot.
```

(vi) Click on Project -> Python interpreter

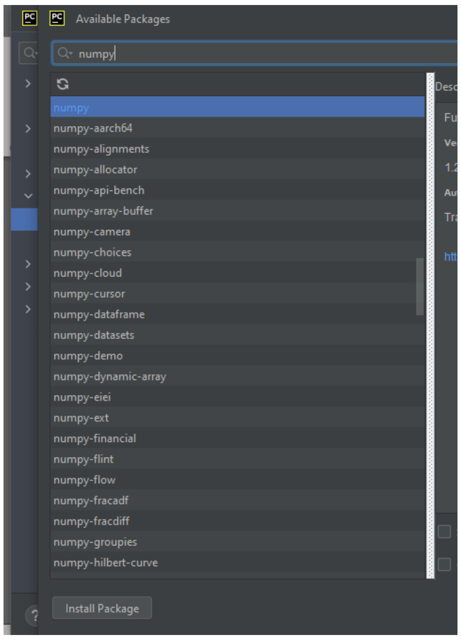


Project: LABS › Python Interpreter

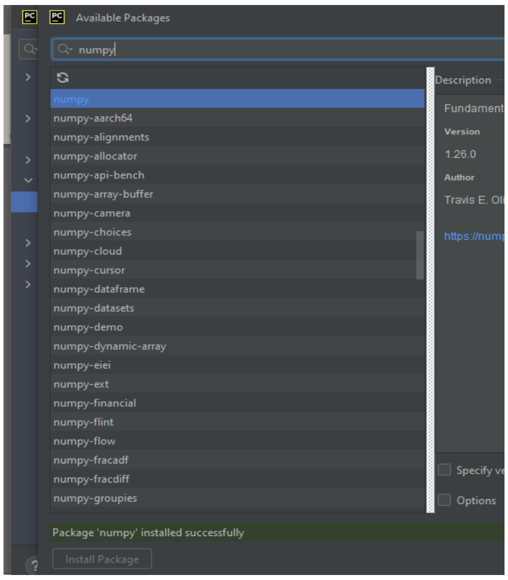
Python Interpreter: Python 3.9 (LABS) C:\Users\Studentlab\De

Pa	Version
numpy	1.26.0
pip	21.3.1
setuptools	60.2.0
wheel	0.37.1

(vii) Search for numpy and click on install on the bottom left-hand corner

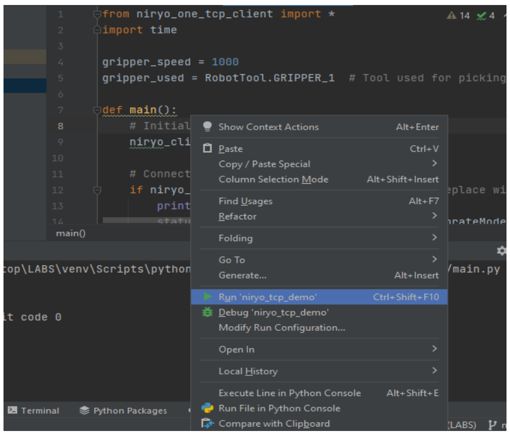


(viii) Close the modules window and click on ok on the setting window



Package 'numpy' installed successfully

(ix) Right click on the file and chose run niryo_tcp_demo.py



```
from niryo_one_tcp_client import *
import time

gripper_speed = 1000
gripper_used = RobotTool.GRIPPER_1  # Tool used for picking

def main():
    # Initial
    niryo_cli

    # Connect
    if niryo_
        print
```

Show Context Actions Alt+Enter
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Go To
Generate... Alt+Insert
Run 'niryo_tcp_demo' Ctrl+Shift+F10
Debug 'niryo_tcp_demo'
Modify Run Configuration...
Open In
Local History
Execute Line in Python Console Alt+Shift+E
Run File in Python Console
Compare with Clipboard
Terminal
Python Packages

```
op\LABS\venv\Scripts\python        /main.py

t code 0
```

3.5. Running the Sample Program

The Sample Program and all other subsequent programs in other experiments must ensure that the Robot is first Calibrated before it can be operated. The program calls the calibration function. If the Robot is not calibrated it will go through a series of movements for each joint to ensure the Robot coordinates for each joint is set to an initial value. If the Robot has already been calibrated, then it will continue to execute the next instruction in the program. This will be demonstrated when running the Sample program.

Below is a short sample program that demonstrates connecting to the Niryo One robot, moving it to a target pose, opening and closing the gripper, and returning the robot to its home position. Make sure to adjust the IP address to match your robot's actual IP:

3.6. Sample code

```
from niryo_one_tcp_client import *
import time

gripper_speed = 1000
gripper_used = RobotTool.GRIPPER_1  # Tool used for picking

def main():
    # Initialize the Niryo One TCP Client
    niryo_client = NiryoOneClient()

    # Connect to the robot
    if niryo_client.connect('192.168.1.100'):  # Replace with your robot's
IP address
        print("Connected to Niryo One robot.")
        status, data = niryo_client.calibrate(CalibrateMode.AUTO)
        niryo_client.change_tool(gripper_used)
        niryo_client.open_gripper(gripper_used, gripper_speed)
        try:
            # Move to a target pose
            target_pose = [0.2, 0.0, 0.2, -1.57, 0.0, 0.0]
            niryo_client.move_pose(0.2, 0.0, 0.2, -1.57, 0.0, 0.0)
            print("Moved to target pose.")
            niryo_client.change_tool(gripper_used)

            # Open the gripper
            niryo_client.open_gripper(gripper_used, gripper_speed)
            print("Gripper opened.")

            # Wait for a few seconds
            time.sleep(2)

            # Close the gripper
            niryo_client.close_gripper(gripper_used, gripper_speed)
            print("Gripper closed.")

            niryo_client.open_gripper(gripper_used, gripper_speed)
            print("Gripper opened.")

            # Wait for a few seconds
            time.sleep(2)

            # Move back to the home position
            niryo_client.move_joints(0.0,0.0,0.0,0.0,0.0,0.0)
            print("Returned to home position.")

            status, data = niryo_client.set_learning_mode(True)
        except Exception as e:
            print(f"An error occurred: {e}")
            status, data = niryo_client.set_learning_mode(True)

        # Disconnect from the robot
        niryo_client.quit()
        print("Disconnected from the robot.")

    else:
        print("Failed to connect to the Niryo One robot.")

if __name__ == "__main__":
    main()
```

3.7. Conclusion

In this lab exercise, you have learned how to set up a PyCharm project, install the Niryo One Python TCP Client library from a GitHub repository, mark the cloned folder as a sources root, and run a sample program to control the Niryo One robot arm using the TCP Client library.

Remember that working with robotics involves safety precautions. As you explore further, make sure to have a good understanding of the robot's capabilities and limitations before attempting complex tasks.

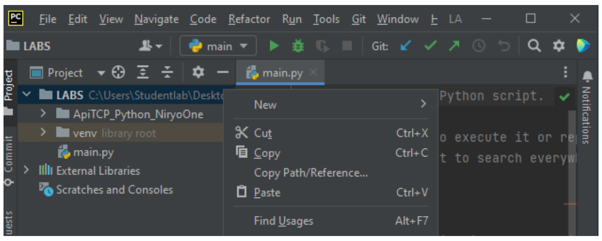
4. Lab Exercise 1: Niryo One Calibration and Basic Movement

In this lab exercise, we will create a Python script to calibrate the Niryo One robot and execute a simple program that moves the robot to a specific set of joint angles.

4.1. Procedure

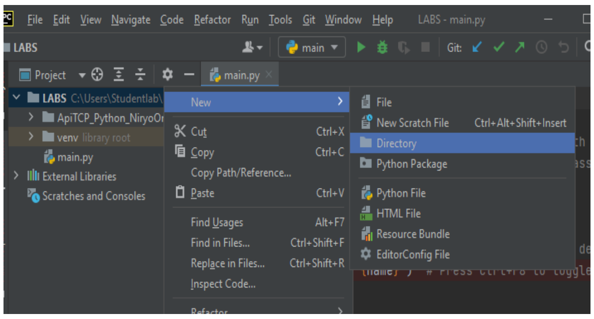
The program code for the laboratory exercises has been divided into **functions.py, main.py** and **variables.py**.

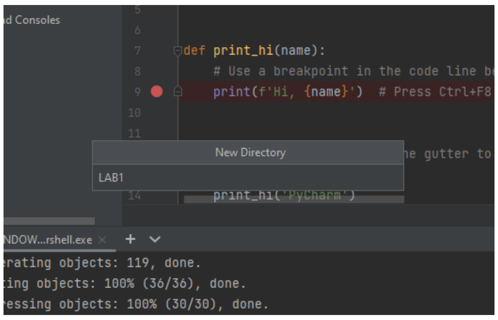
(i) Right click on the project folder called LABS and click on new -> directory



```
Python script.
o execute it or re
t to search everyw
```

(ii) Name the directory LAB1

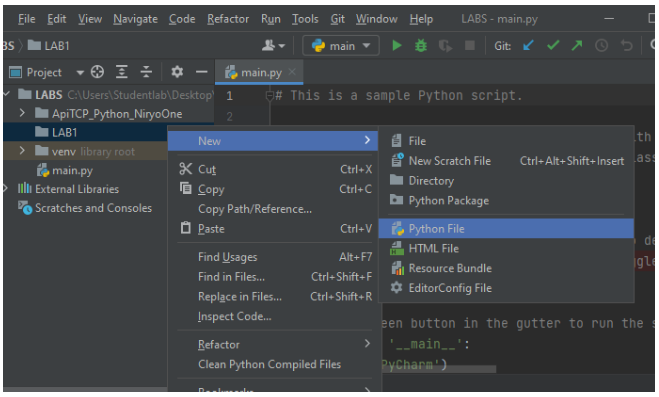




```
def print_hi(name):
    # Use a breakpoint in the code line b
    print(f'Hi, {name}')  # Press Ctrl+F8

                                      he gutter to

    print_hi('PyCharm')
```

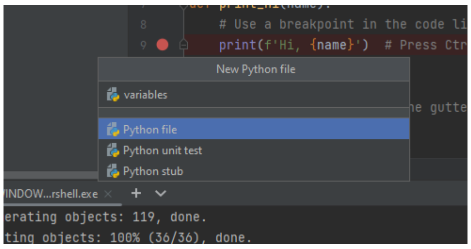
```
rating objects: 119, done.
ing objects: 100% (36/36), done.
essing objects: 100% (30/30), done.
```

(iii) Right click on the directory LAB1 and select new -> Python file



```
# This is a sample Python script.
```

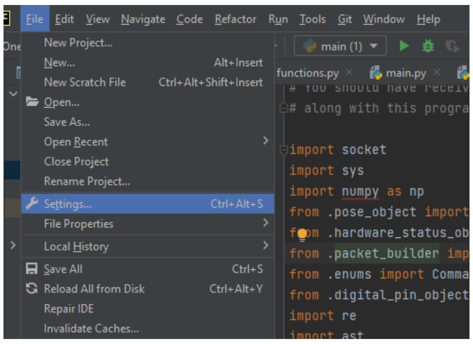
(iv) Name this file variables



```
    # Use a breakpoint in the code li
    print(f'Hi, {name}')  # Press Ctr
```

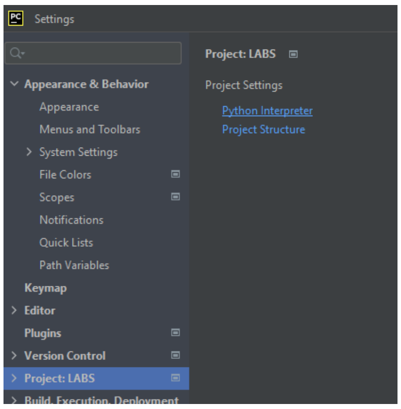
```
erating objects: 119, done.
ting objects: 100% (36/36), done.
```

(v) Follow the same procedure for creating the following files “functions” and “main”
(vi) Add the corresponding code below to each file.
(vii) Click on File -> Settings

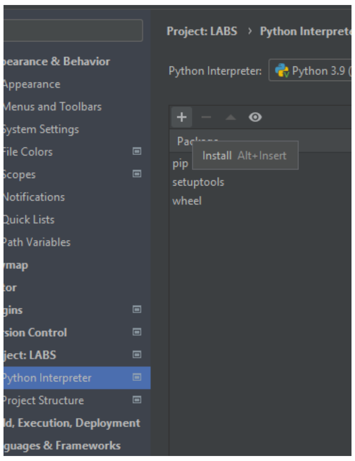


```
# along with this progra

import socket
import sys
import numpy as np
from .pose_object import
from .hardware_status_ob
from .packet_builder imp
from .enums import Comma
from .digital_pin_object
import re
```

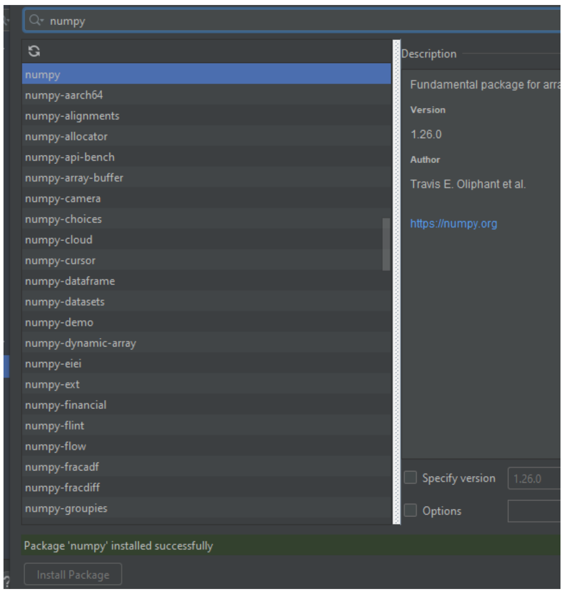
(viii) Select Project LABS -> Interpreter



(ix) Select the plus sign



(x) Install the NumPy library. After successful installation a green message appears. Close the window and click on okay to close the settings window.



Package 'numpy' installed successfully

4.2. Project Structure

- `functions.py`

- `connection to the network`: Contains the `connectRobot` function to establish a connection between the robot and the router.
- `movement`: Contains the `calibrate` and `moveToPoint` functions for robot calibration and movement.

- `variables.py`: Contains the variables for gripper speed, gripper tool, and sleep joints.

- `main.py`: The main program that initializes the client, performs calibration, moves the robot, and disconnects.

4.3. Project python code

## place in the `variables.py` file

```python
from niryo_one_tcp_client import *
# Set gripper speed and tool for picking
gripper_speed = 1000
gripper_used = RobotTool.GRIPPER_1  # Tool used for picking

# Set sleep joints for returning to a specific position
sleep_joints = [0.0, 0.55, -1.2, 0.0, 0.0, 0.0]
```

## place in the `functions.py` file

```python
from niryo_one_tcp_client import *
import variables as v

def connectRobot(niryo_one_client):
    # Connect to the robot using the provided IP address
    niryo_one_client.connect("10.10.10.10")



def calibrate(niryo_one_client):
    # Calibrate the robot in auto mode
    status, data = niryo_one_client.calibrate(CalibrateMode.AUTO)
    print('Calibration Status:', status)
    print('Calibration Data:', data)

def moveToPoint(niryo_one_client):
    # Move the robot to a specific set of joint angles (0 degrees for all
joints)
    niryo_one_client.move_joints(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)

    # Change the tool to the specified gripper tool
    niryo_one_client.change_tool(v.gripper_used)

    # Close the gripper with specified speed
    niryo_one_client.close_gripper(v.gripper_used, v.gripper_speed)

    # Open the gripper with specified speed
    niryo_one_client.open_gripper(v.gripper_used, v.gripper_speed)

    # Move the robot to sleep position (specified joint angles)
    niryo_one_client.move_joints(*v.sleep_joints)

    # Set the robot to learning mode
    niryo_one_client.set_learning_mode(True)

    # Quit and release connection
    niryo_one_client.quit()
```

## place in the `main.py` file

```
from niryo_one_tcp_client import *
from functions import connection, movement
import variables as v

if __name__ == '__main__':
    # Initialize NiryoOneClient
    niryo_one_client = NiryoOneClient()

    # Connect to the robot
    connectRobot(niryo_one_client)
    print('Connected to Niryo One robot.')

    # Calibrate the robot
    calibrate(niryo_one_client)
    print('Calibration completed.')

    # Move the robot to a specific point
    moveToPoint(niryo_one_client)
    print('Movement completed.')

    # Disconnect from the robot
    niryo_one_client.quit()
    print('Disconnected from Niryo One robot.')
```

4.4. Project Description

This structured project separates the code into different folders and files for better organization. The `**variables.py**` file contains the variables for gripper speed, gripper tool, and sleep joints. The `**connection.py**` and `**movement.py**` files within the `**functions/**` folder contain functions for connecting to the robot and handling movement actions, respectively. Finally, the `**main.py**` file ties everything together, initializing the client, performing calibration, moving to the target point, and disconnecting from the robot.

Remember to replace `"10.10.10.10"` with the actual IP address of your robot and ensure that you have the required libraries/modules and variables set up in your project.

4.5. Task

(a) Create a Python function to move the robot to a different coordinate location on the J1 axis.

(b) Try incrementing other axes J2, J3, J4, J5, J6 in increments of 0.1+-.Do not try higher values as you can damage the robot.

(c) . Record your observations.

5. Lab Exercise 2: Niryo One Robot Movement in Joint Mode

In this lab exercise, we will organize the code into separate files and folders to demonstrate the Niryo One robot's movement in joint mode.

Follow the procedure in section 3.4 for creating a new lab file.

5.1. Project Structure

- **`functions.py`**
  - `connection`: Contains the `connectRobot` function to establish a connection with the robot.
  - `movement`: Contains the `calibrate`, `jointsPosition`, and `moveToPoint` functions for robot calibration and joint movement.

- **`variables.py`:** Contains the variables for gripper speed, gripper tool, sleep joints, and joint positions.

- **`main.py`:** The main program that initializes the client, connects to the robot, calibrates it, and performs joint movements.

`variables.py`

```
# user-defined joint positions
from niryo_one_tcp_client import *
counter = 0
J1 = -1.5
J2 = 0.0
J3 = 0.0
J4 = 0.0
J5 = 0.0
J6 = 0.0

# Set gripper speed and tool for picking
gripper_speed = 1000
gripper_used = RobotTool.GRIPPER_1  # Tool used for picking

# Set sleep joints for returning to a specific position
sleep_joints = [0.0, 0.55, -1.2, 0.0, 0.0, 0.0]
```

`functions.py`

```
from niryo_one_tcp_client import *
import variables as v

def connectRobot(niryo_one_client):
    niryo_one_client.connect("10.10.10.10")  # connect to the Robot's
Wireless Hotspot


def calibrate(niryo_one_client):
    status, data = niryo_one_client.calibrate(CalibrateMode.AUTO)  # Robot
Calibration Function

def jointsPosition():
    # increment Joint 1 (Waist) by 0.1 radians
    v.J1 = v.J1 + 0.1

def moveToPoint(niryo_one_client):
    positionsNumber = 3  # Number of times to increment joint values

    while v.counter <= positionsNumber:
        # command to move the Robot in Joint Mode
        niryo_one_client.move_joints(v.J1, v.J2, v.J3, v.J4, v.J5, v.J6)
        v.counter = v.counter + 1  # increment the counter
        jointsPosition()
        niryo_one_client.wait(1)  # delay for 1 second

    niryo_one_client.move_joints(*v.sleep_joints)  # move the Robot to the
Sleep Joint Position
    niryo_one_client.set_learning_mode(True)  # Enable Learning Mode =
Release breaks of the Robot joint
    niryo_one_client.quit()  # disconnect from Network
```

`main.py`

```
from niryo_one_tcp_client import *
from functions import *
import variables as v

if __name__ == '__main__':
    niryo_one_client = NiryoOneClient()

    # Connect to the robot
    connectRobot(niryo_one_client)
    print('Connected to Niryo One robot.')

    # Calibrate the robot
    calibrate(niryo_one_client)
    print('Calibration completed.')

    # Move the robot through joint positions
    moveToPoint(niryo_one_client)
    print('Movement completed.')
    # Disconnect from the robot
    niryo_one_client.quit()
    print('Disconnected from Niryo One robot.')
```

5.2. Project Description

This structured project separates the code into different folders and files for better organization. The `**variables.py**` file contains the variables for joint positions, gripper speed, gripper tool, and sleep joints. The `**connection.py**` and `**movement.py**` files within the `**functions/**` folder contain functions for connecting to the robot and handling movement actions in joint mode. Finally, the `**main.py**` file ties everything together, initializing the client, connecting to the robot, calibrating it, moving the robot through joint positions, and disconnecting from the robot.

Remember to replace `"10.10.10.10"` with the actual IP address of your robot and ensure that you have the required libraries/modules and variables set up in your project.

5.3. Task

(a) Comment on the outcome of the exercise in comparison to that of lab exercise section 4.2

(b) Change the direction of movement so the robot goes back to the starting position. (Hint: deduct the values from the counter so that the robot changes direction)

6. Lab Exercise 3: Niryo One Robot Path Planning and Execution

In this lab exercise, we will organize the code into separate files and folders to demonstrate the Niryo One robot's path planning and execution using learning/teach mode and repeat mode.

Follow the same procedure you used to install the NumPy module in Section 4.1 (vii-x) for installing the keyboard module.

6.1. Project Structure

- **`functions.py`**
  - `connection`: Contains the `connectRobot` function to establish a connection with the robot.
  - `movement`: Contains the `calibrate`, `learningMode`, and `repeatMode` functions for robot calibration, learning/teach mode, and repeat mode.

- **`variables.py`:** Contains the variables for gripper speed, gripper tool, sleep joints, and joint positions.

- **`main.py`:** The main program that initializes the client, connects to the robot, calibrates it, and provides menu options for learning/teach mode or repeat mode.

`variables.py`

```
# Variables for gripper, joint positions, and other settings
import numpy as np
from niryo_one_tcp_client import *
gripper_used = RobotTool.GRIPPER_1  # SERVO based Gripper
gripper_speed = 1000  # max speed 1000
sleep_joints = [0.0, 0.55, -1.2, 0.0, 0.0, 0.0]  # joint positions when
Robot is calibrated
counter = 0

# Default SAFE joint positions
J1 = -1.5
J2 = 0.0
J3 = 0.0
J4 = 0.0
J5 = 0.0
J6 = 0.0
i = 0

# Menu option
option = ''

# Learning/Teach mode option
keyPressed = ""
```

```
# Maximum number of positions to Teach/Learn
number_Of_positions = 6

# Array of taught positions
np.joints = []

# Delay in seconds at each position reached
position_delay = 2
```

`functions.py`

```
from niryo_one_tcp_client import *
import variables as v
import numpy as np

def connectRobot(n1):

    status=n1.connect("10.10.10.10")
    n1.change_tool(v.gripper_used)
    n1.open_gripper(v.gripper_used, v.gripper_speed)
    print("Connection to robot successful\nGripper attached ")

def calibrate(n1):

       status, data = n1.calibrate(CalibrateMode.AUTO)
       n1.move_joints(*v.sleep_joints)
       n1.set_learning_mode(True)
       print("Robot calibrated and in initial position")

def learningMode(n1):
#Begining of Learning/Teach Mode

    status, data1 = n1.set_learning_mode(True)
    # Ensure the robot joint coordinate arrays are deleted from the list
    # CAUTION !! every time you run the learning mode the previous taught
#poistions are deleted
    del np.joints[:]
    print("-------Learning Mode----------- ")
    # In this mode pressing s - stores the position with the gripper
#remaining OPEN
    # pressing g - stores the positions with the gripper CLOSED

    # This while loop ends either by pressing 'q' or when all points are
#taught
    # defined by the maximum value of number_Of_positions (declared in
#variables.py)
    while v.keyPressed != 'q':

        v.keyPressed = input(" Press s+ENTER for open gripper position"
                             "\n Press g+ENTER for close gripper position"
                             "\n Press q+ENTER to quit\n  ")

        if v.keyPressed in  ['s','g']:
            if v.i < v.number_Of_positions:
                # get the joint coordinates of the Robot and store them in
#data2 array
                status, data2 = n1.get_joints()
                # depending on Gripper status insert 0 for OPEN and 1 for
#CLOSE Gripper in
```

```
                # array position 6
                data2.insert(6,0)
                if v.keyPressed == 'g':
                    data2[6]= 1
                # point to the next joint array position by appending data
#to it
                np.joints.append(data2)
                v.i = v.i + 1
                print("Positions in radians:
[j1,j2,j3,j4,j5,j6,gripper],[j1,j2....second position],[...]etc ")
                print(np.joints)
                print("Number of positions %d " % len(np.joints))

                # check the total number of positions are taught, then stop
                if v.i == v.number_Of_positions:
                   print("Taught points maximum value is reached")
                   break # break out of the while loop if all points are
#Taught
# end of the WHILE LOOP

def repeatMode(n1,joints):
# extract the joints one point at a time from the stored list and move to
#the coordinates
        for i in range (len(joints)):
            print("[j1,           j2,              j3,            j4,
j5,              j6,          gripper 0-Open 1-Closed]")
            print(joints[i][0], joints[i][1], joints[i][2], joints[i][3],
joints[i][4], joints[i][5], joints[i][6])
            n1.move_joints(joints[i][0], joints[i][1], joints[i][2],
joints[i][3], joints[i][4], joints[i][5])

            if joints[i][6] == 0:
            # check the array's 6th position to detect the Gripper FLAG
            # if the flga is 1 CLOSE gripper of 0 then OPEN Gripper
                  n1.open_gripper(v.gripper_used, v.gripper_speed)
            else:
                   n1.close_gripper(v.gripper_used, v.gripper_speed)
            # wait for 2 senconds
            n1.wait(v.position_delay)


        n1.move_joints(*v.sleep_joints)
        n1.set_learning_mode(True)
        print("Repeat mode ended")
```

`main.py`

```
from niryo_one_tcp_client import *
from functions import *
import variables as v
import numpy as np


def main():
    print("Niryo 1 Robot Control Program")
    n1 = NiryoOneClient()
    connectRobot(n1)
    calibrate(n1)
    while v.option != '3':
        print("1. Learning/Teach Mode"
              "\n2. Repeat Mode"
              "\n3. Exit Program")
        v.option = input("Enter Choice: ")

        if v.option == '1':
            print('Learning/Teach Mode')
            learningMode(n1)

        elif v.option == '2':
            print('Repeat Mode')
            repeatMode(n1, np.joints)

        elif v.option == '3':
            print("Exited Program")
            exit()


if __name__ == '__main__':
    main()
```

6.2. Project Description

This structured project separates the code into different folders and files for better organization. The `**variables.py**` file contains variables for gripper settings, joint positions, menu options, and other parameters. The `**connection.py**` and `**movement.py**` files within the `**functions/**` folder contain functions for connecting to the robot, handling calibration, learning/teach mode, and repeat mode. Finally, the `**main.py**` file ties everything together, initializing the client, connecting to the robot, calibrating it, and providing menu options for learning/teach mode or repeat mode.

6.3. Task

(a) Use teach mode to pick a block from one position and move it to a different position.

(b) Pick up two blocks from their locations and stack them on top of one another at a different location.

7. Lab Exercise 4: Niryo One Robot Position Control

In this lab exercise, we will organize the code into a single file called main.py to demonstrate the Niryo One robot's position control using keyboard inputs.

Follow the procedure in section 3.4 for creating a new lab file.

7.1. Calibration

Due to mechanical assembly each robot has different starting angles for each of the 6 DoF joints. Therefore, the calibration for utilising the move position commands requires each robot to have different values for (x,y,z,roll,yaw,pitch)

A main.py file has been provided in the link below. The name of the file matches the IP address of each of the 6 robots available in the lab. Make sure to select the correct file to avoid robot malfunction or unpredicted robot behaviour.

```
https://github.com/LSBU-Electronics-
Lab/NiryoOneExperiments/tree/main/Niryo%20Robots%20mainPY%20Calibration%20d
ata
```

7.2. Project Structure

- `main.py`: The main program that initializes the client, connects to the robot, calibrates it, and controls the robot's position using keyboard inputs.

`main.py` **(Do not copy the code below it is just for reference use the file referred to in section 7.1)**

Note: The lines marked in red differ from each robot therefore the use of the calibration files.

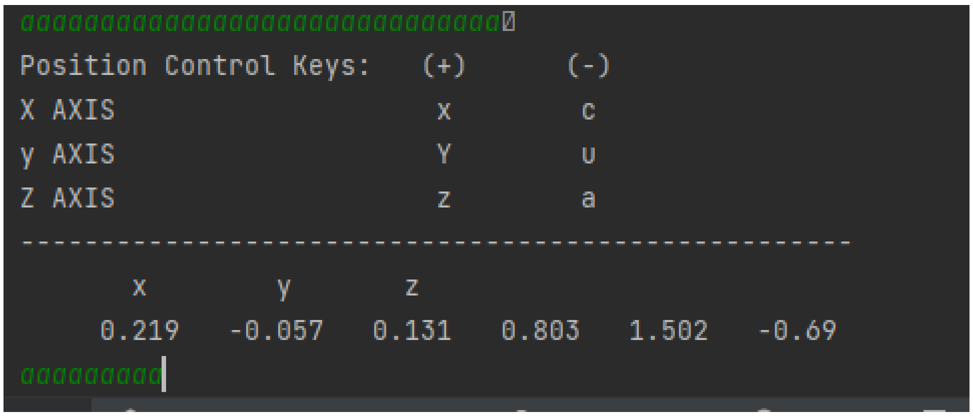
```
import time
from niryo_one_tcp_client import *
import keyboard

import os

x=0.22
y=-0.061
z=0.133
r=0.803
```

```
p=1.502
ya=-0.69

key=' '
incr=0.001 # increment value for all axes; May be changed to
           # different values for each axis
def main():

    global x,y,z,r,p,ya,incr

    print(" Niryo 1 Robot Control Program")

    # Connect to the Robot Wireless Hotspot
    n1 = NiryoOneClient()
    status = n1.connect("10.10.10.10")

    position_keys()

    n1.change_tool(RobotTool.GRIPPER_1)
    n1.open_gripper(RobotTool.GRIPPER_1, 1000)
    #n1.__init__()
    status, data = n1.calibrate(CalibrateMode.AUTO)
    # activate learning mode
    n1.set_learning_mode(True)
    # go to initial position defined above
    n1.move_pose(x, y, z, r, p, ya)
    # stay in the while loop below
    while True:


        if keyboard.read_key()=="x":
           x=x+incr
        elif keyboard.read_key()=="c":
            x=x-incr
        elif keyboard.read_key()=="y":
            y=y+incr
        elif keyboard.read_key()=="u":
            y=y-incr
        elif keyboard.read_key()=="z":
            z=z+incr
        elif keyboard.read_key()=="a":
            z=z-incr

        elif keyboard.read_key()=="-":
            n1.open_gripper(RobotTool.GRIPPER_1, 1000)
        elif keyboard.read_key()=="+":
            n1.close_gripper(RobotTool.GRIPPER_1, 1000)

        elif keyboard.read_key()=="q":

        # when 'q' is pressed , break out of the loop
             break
        #Clears the screen preventing it from being populated continuously
with data
        clear_screen()
        # function that print Labels for position keys x,y,z axis
        position_keys()
        # ---------------------------------------------------------
        print("----------------------------------------------------")
        print("       x        y        z  ")
        # print the pose /world coordinatesxxx
        print(key,"  ",round(x,7)," ",round(y,7)," ",round(z,7)," ",r,"
",p," ",ya)

        #move the Robot
        n1.move_pose(x, y, z, r, p, ya)
# END of WHILE: remain in the while loop until 'q' is pressed
    # go to the sleep position

    n1.move_joints(0.0, 0.55, -1.2, 0.0, 0.0, 0.0)
    n1.set_learning_mode(True)

def clear_screen():
    os.system('cls')  # on Windows System
def position_keys():

    print("\nPosition Control Keys:   (+)       (-)\n"
          "X AXIS                     x         c\n"
          "y AXIS                     Y         u\n"
          "Z AXIS                     z         a")

if __name__ == '__main__':
    main()
```

7.3. Project Description

The `**main.py**` file ties everything together, initializing the client, connecting to the robot, and controlling the robot's position using keyboard inputs.

Place the cursor on the terminal window and click in order to be able to control the robot using the keyboard.



```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Position Control Keys:   (+)      (-)
X AXIS                    x        c
y AXIS                    Y        u
Z AXIS                    z        a
----------------------------------------------------
       x        y        z
     0.219    -0.057    0.131    0.803    1.502    -0.69
aaaaaaaaa
```

7.4. Taking measurements

For this task we will take coordinate measurements using Lab 4 main.py. The robot positioning is very important for the gripper to successfully grab an object. The two images below show how the measurement should be taken for block A. The same method should be applied when taking measurements for blocks B and C.

7.4. Correct measurement examples

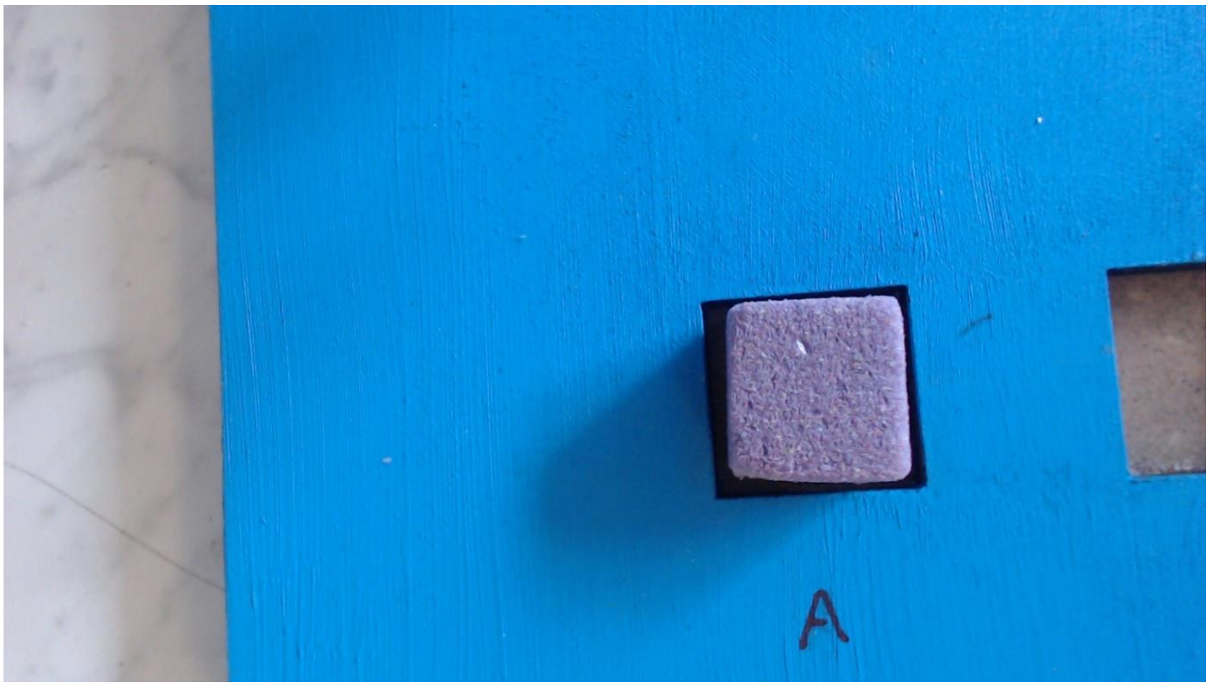
If the gripper is too close to the blue platform, the robot will crash while trying to pick up an object. Therefore, a 5 mm distance above the platform should be sufficient for the robot to grab the object and not crash during the operation.

The robot gripper should be correctly positioned from the side and aligned with the block. Verify these two views are covered correctly before taking your measurement. In addition, notice how the gripper is perpendicular to the platform in both cases. This is very important to prevent the other stacked blocks from falling when trying to move them to other positions.

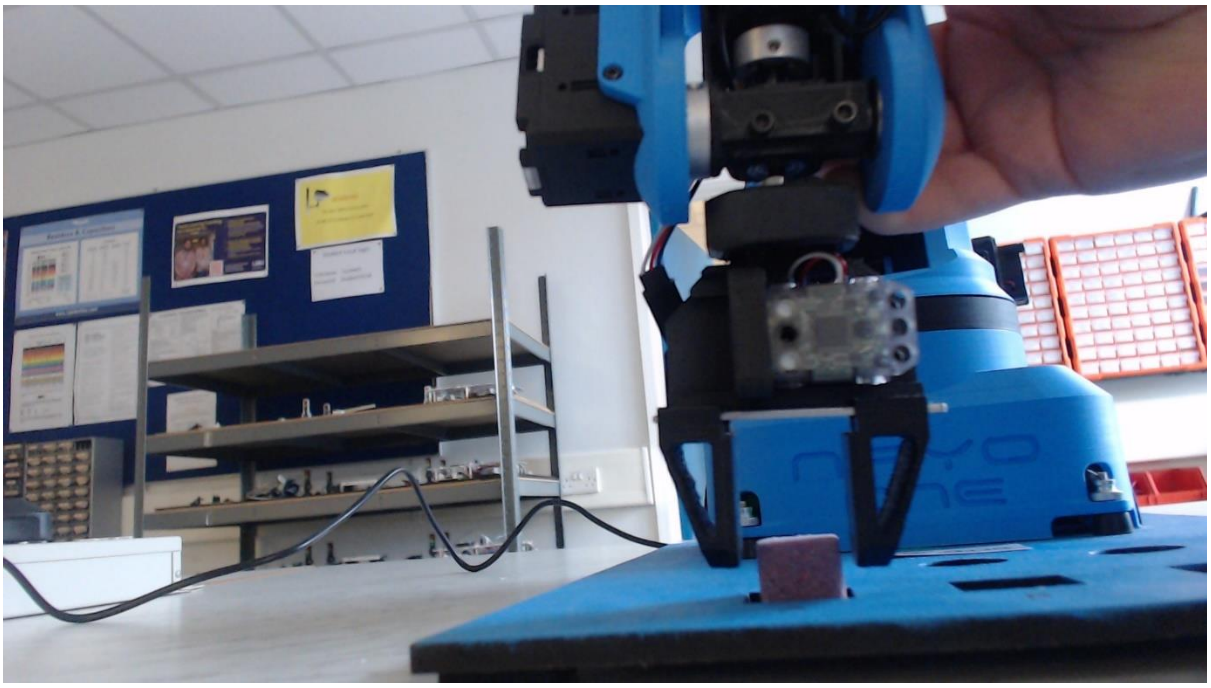
The current gripper position is 0. Avoid turning it multiple times.

The block should be aligned to the platform square as there gripper to prevent one side of the gripper touching the block first causing the block to collapse to one side.

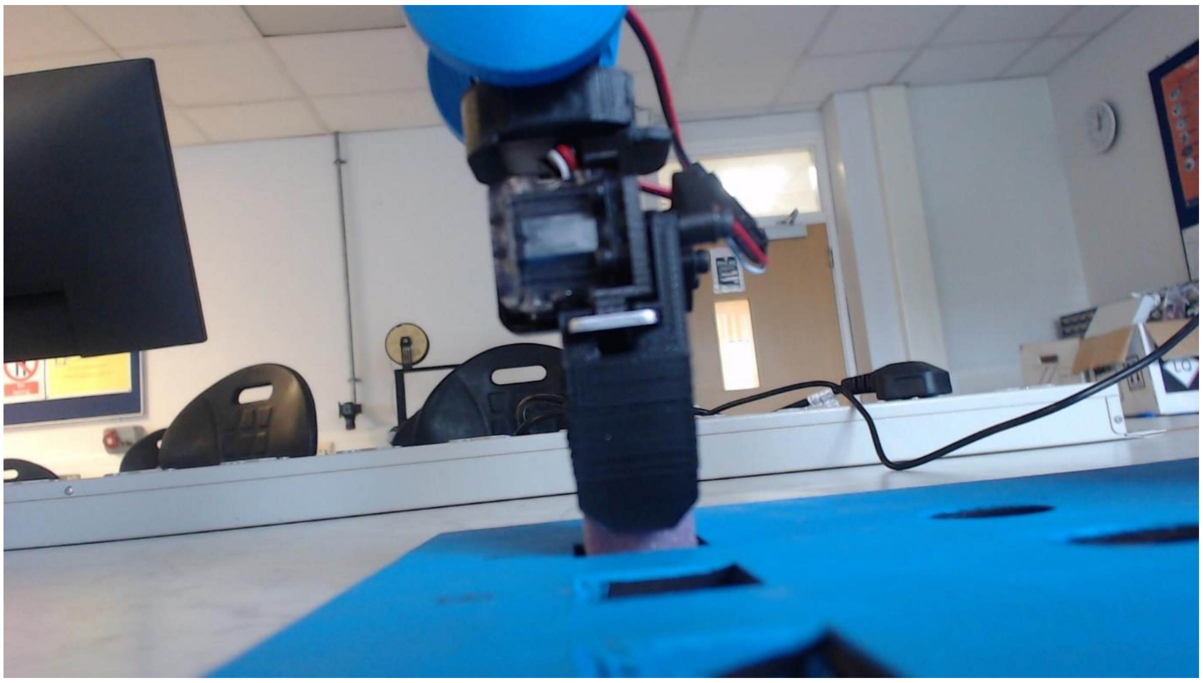
Block aligned to the centre of the platform square on Position A



Correct measurement front view



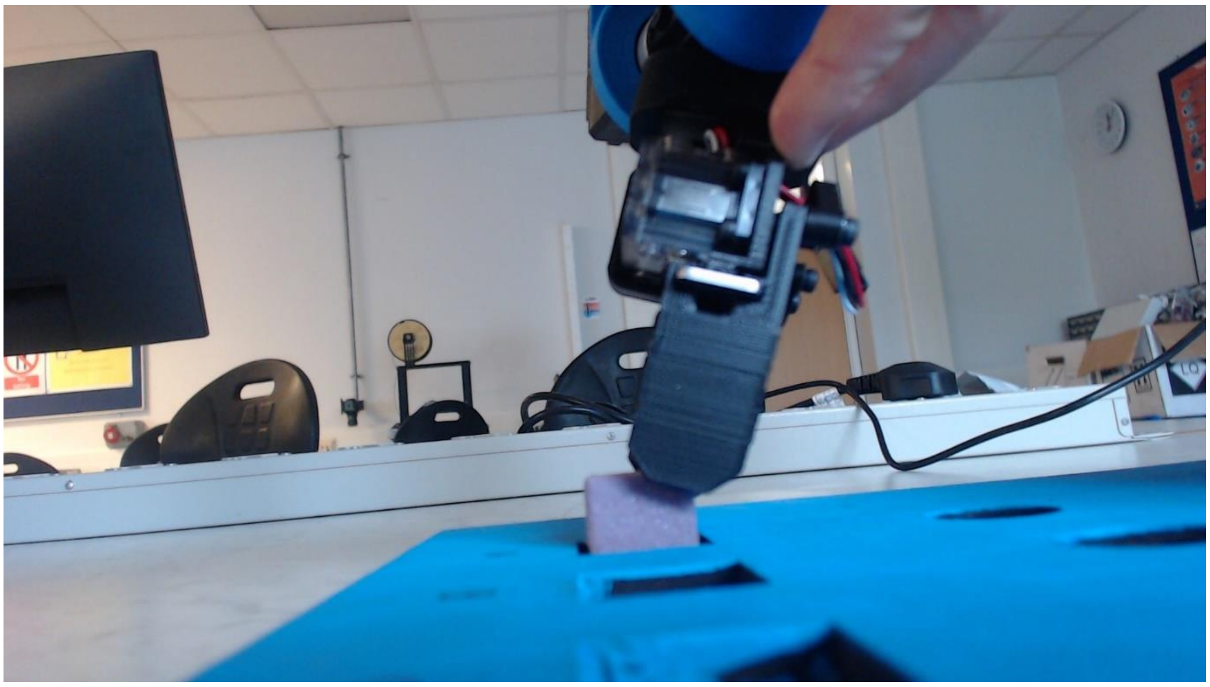
Correct measurement side view



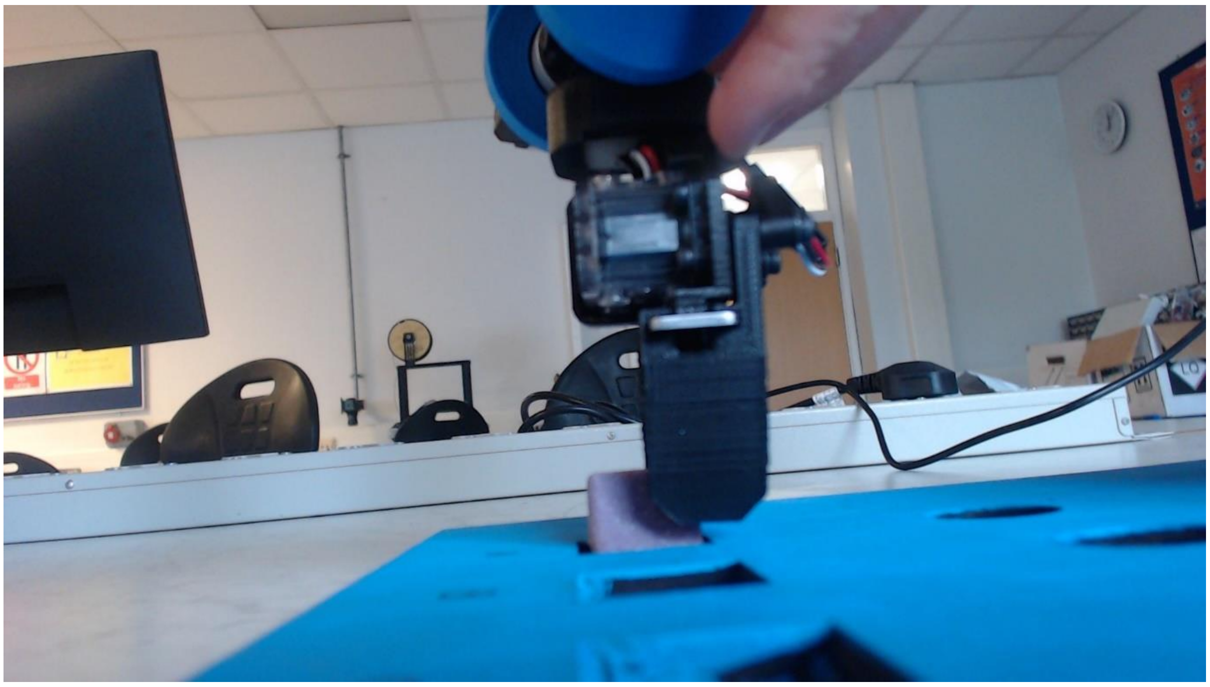
7.4. Incorrect measurement examples

Incorrect measurements

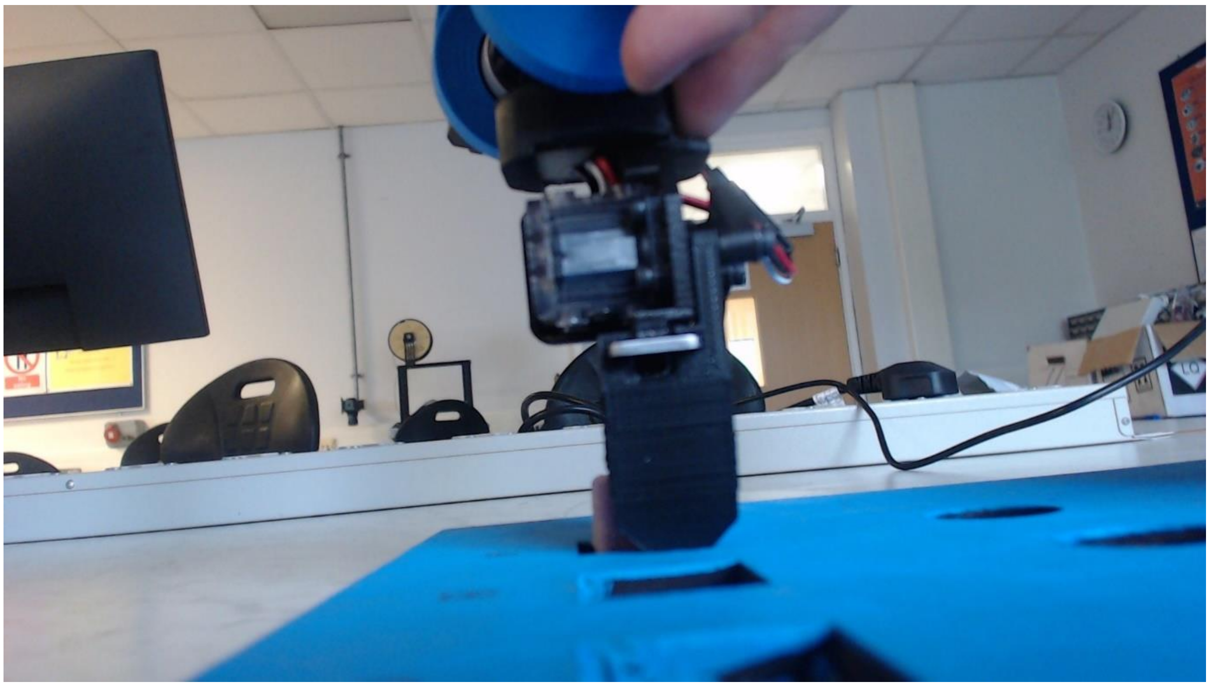
Gripper at an angle



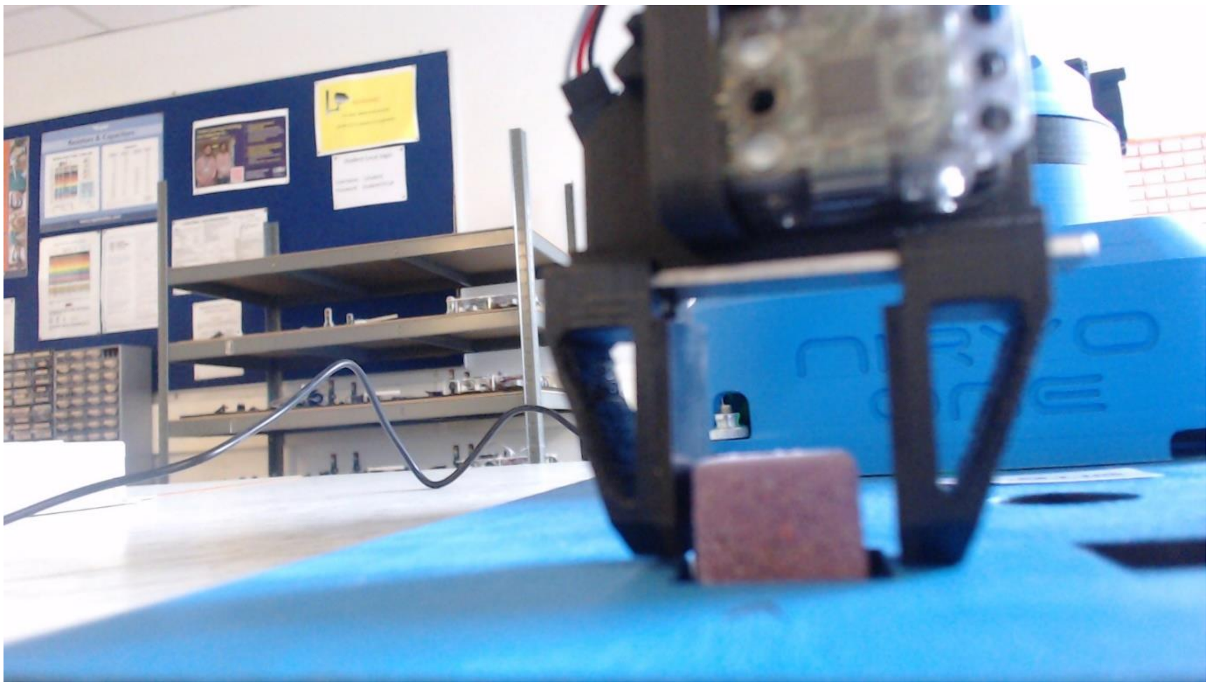
Gripper not aligned with the center of the object



Gripper touching the platform without a 5mm safe distance

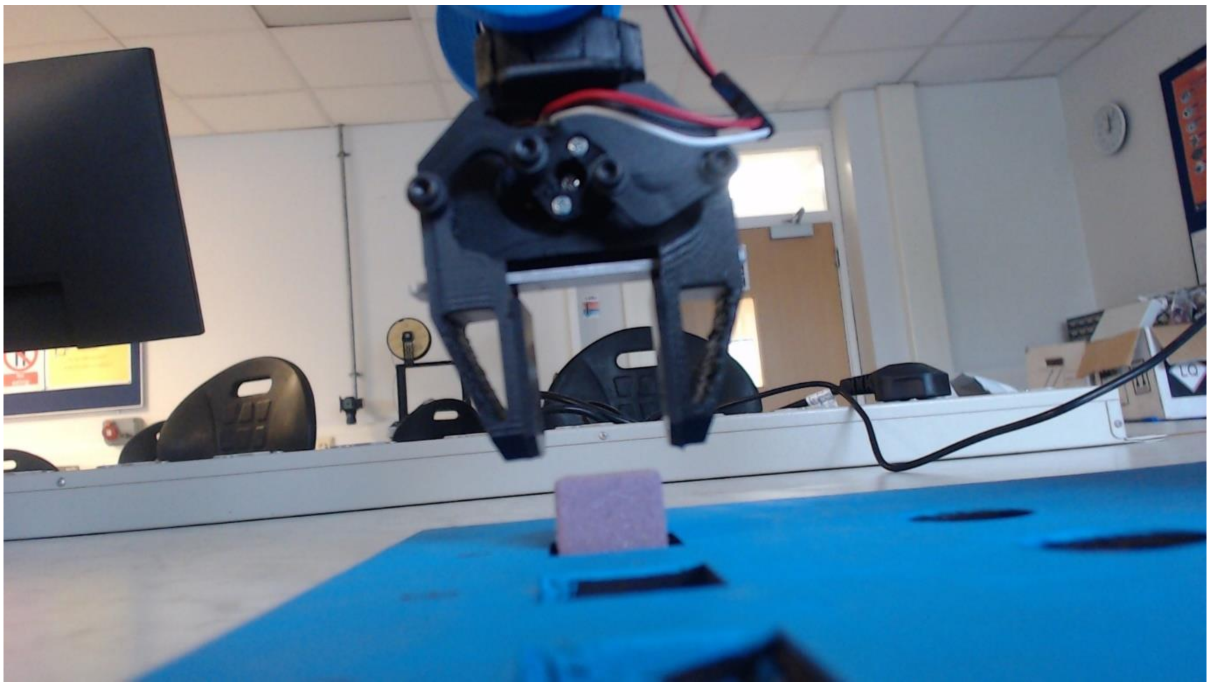


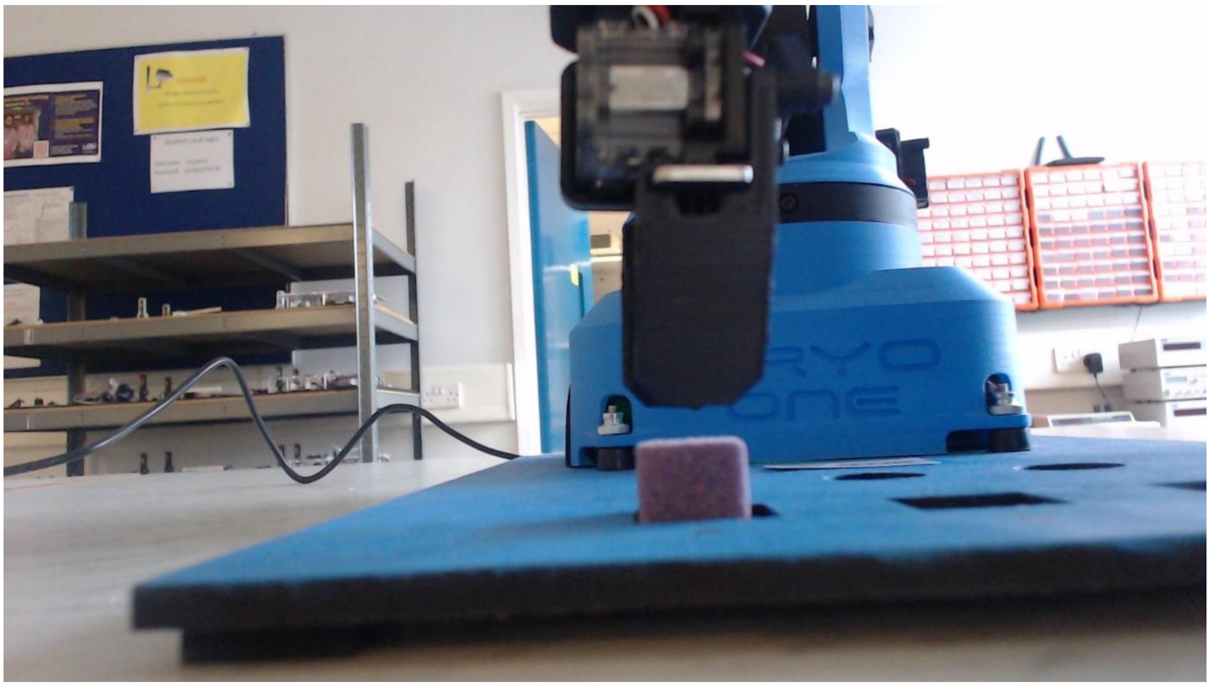
One side of the gripper to close to the block. (Refer to image 1 of section 7.4)



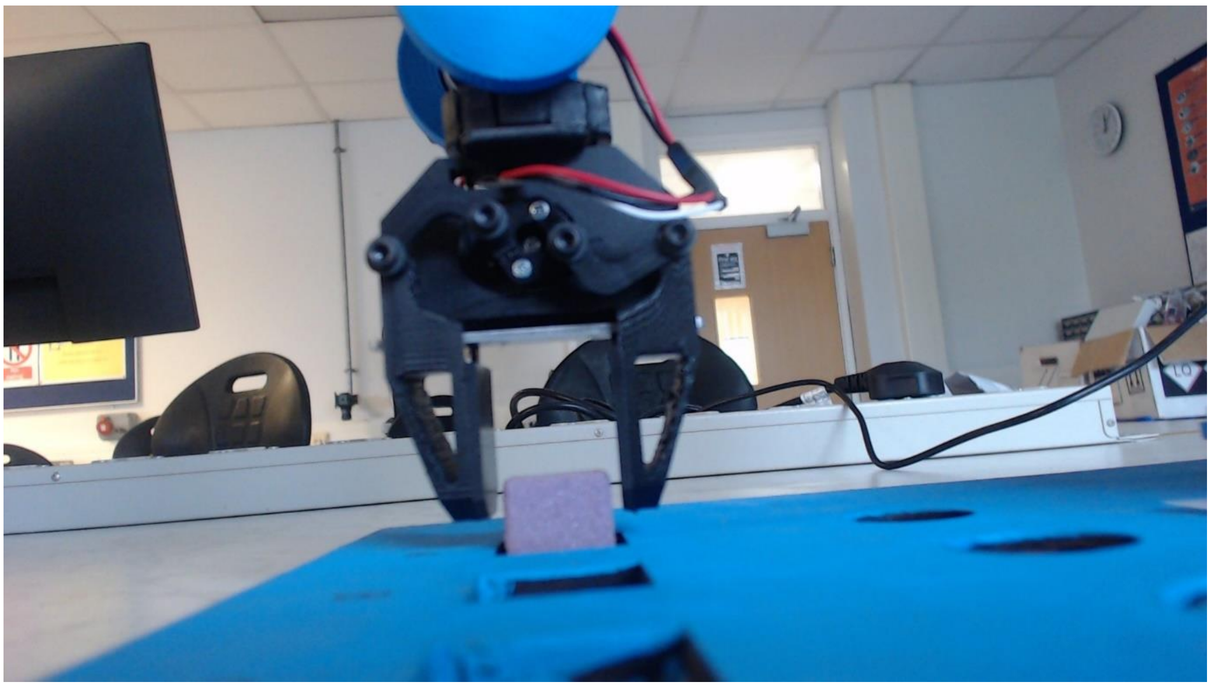
7.5. Task

(a) Place a block in pre pick up position for block A about 5mm above the block. This allow a perpendicular path for the gripper when going down to the block as shown in the image below. Record the x,y,z,r,y,p coordinates in a spreadsheet, The gripper position should be as in this picture with the servo motor facing left.

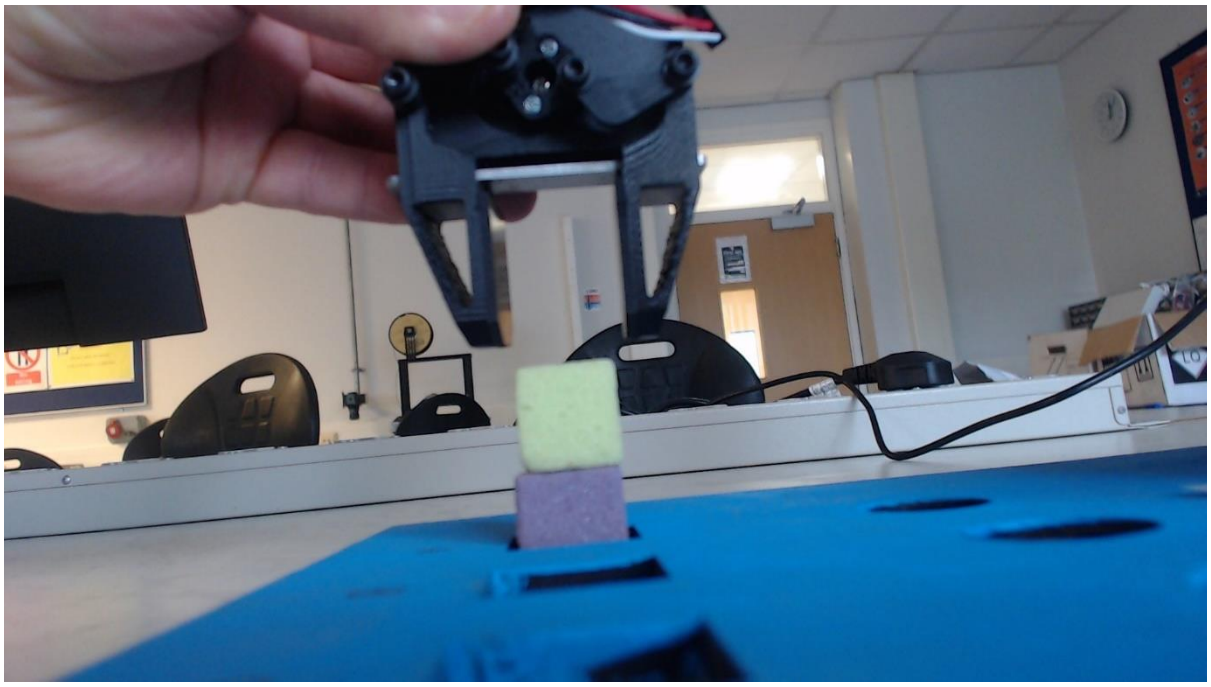




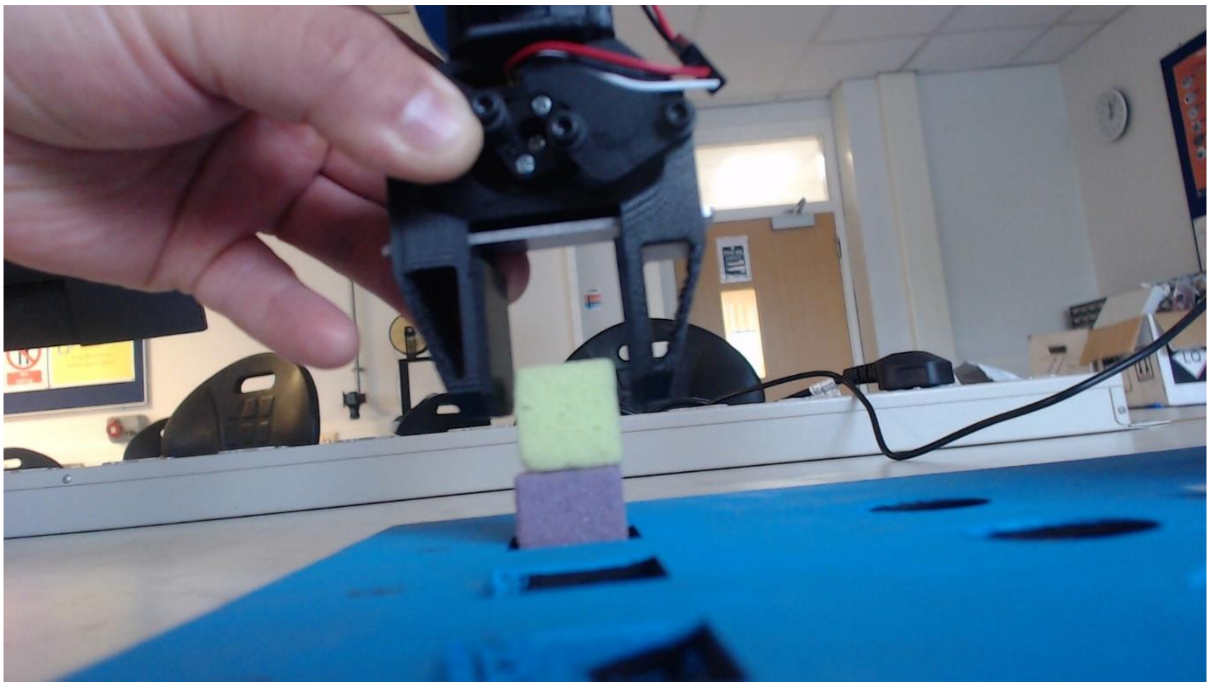
(b) Take a measurement for pickup position block A and record the coordinates in your table.



(c) Place a block on top of block A and take a pre pickup position coordinate and record the values



(d) Record a pickup position coordinate for the second block as below, at about the middle of the block.



(e) Record a pre pick position coordinate for the third block in position A and a pickup coordinate and record it in your table.

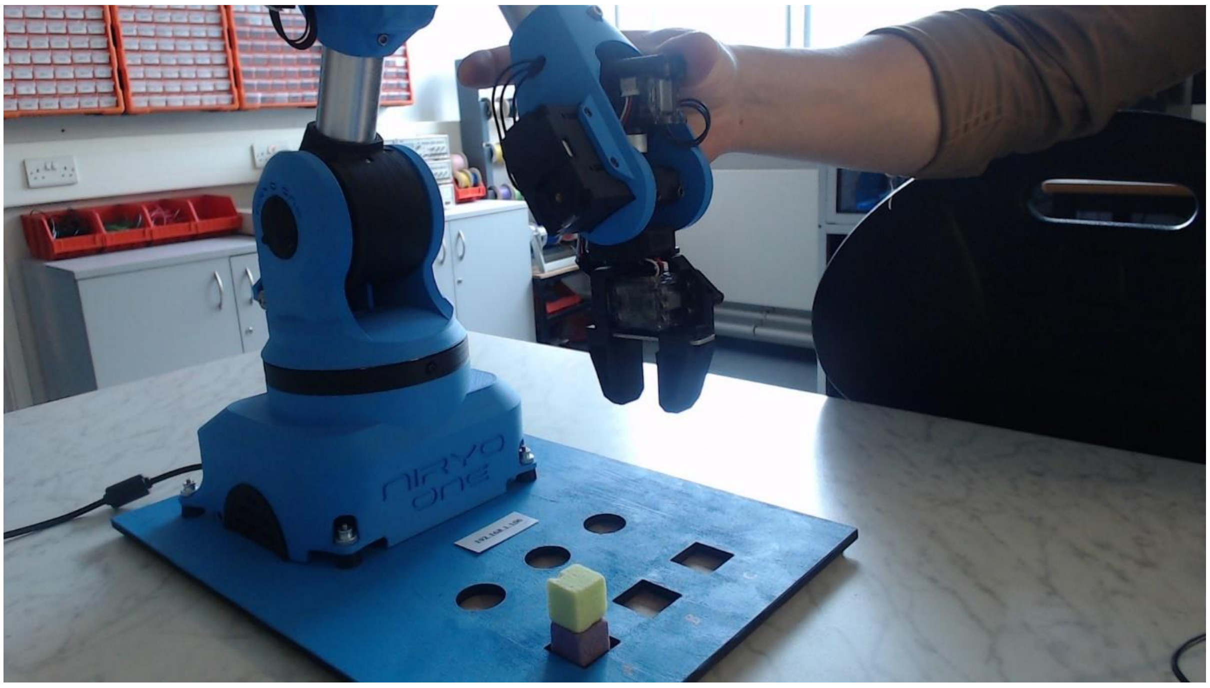
(f) Repeat taking pre pickup and pickup measurements for position B by using the same 3 blocks. Start with one block as in position A and add the other blocks after recording the coordinates.

(g) Repeat taking pre pickup and pickup measurements for position C by using the same 3 blocks. Start with one block as in position A and add the other blocks after recording the coordinates.

(h) Create a program to automate block positioning and stacking, keeping track of the blocks in each position. Use the coordinates recorded in your table to position the robot to pick up blocks. The program must follow the basic steps detailed below to ensure the blocks are grabbed without damaging the robot gripper.

i) The robot should remain ready at the Initial position which is perpendicularly above positions A,B,C at a distance which does not interfere with the blocks as shown in the image below. This starting position should also be recorded in your table with the LAB 4 program so it can be implemented at the start of the block stacking program.

Initial position



ii) One block should rest in each position A, B , C on start

iii) An input should be read, and the robot arm should be positioned in pre pickup position

iv) The robot should go down and pick up the block

v) (Important) The robot should move up perpendicular to the block positionfor about 2cm

vi) Then go back to the Initial position holding the block

vii) An input should be read from the keyboard to select the new position on top of another block

viii) The program should keep track of the number of blocks stack at each position

ix) The program should be able to stack 3 blocks at any position A, B or C

7.6. Lab 5 Demo video

Note the starting position of the robot before and after taking the block your program should follow a similar approach.