This manual explains briefly how to conduct the experiments with the experimental setup.

⇒Hardware
⇒Software Launching, MIOS & Interface, Experiments)
⇒List of commands

Prepare the Experiments | Hardware

⇒Robot

Turn on Robot (switch on the back of the controller)
Make sure it is connected to the network via the switch

⇒NUC

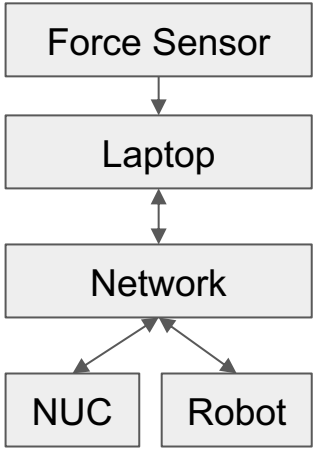
Turn on NUC (if power button is blue -> ON)
Make sure it is connected to the network via the switch

⇒Force Sensor

Turn on GSV8 (front, leftmost button, blue light -> ON)

⇒Laptop

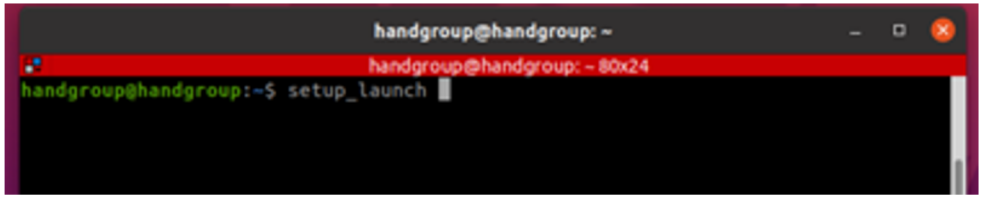
Connect GSV8 via USB
Make sure it is connected to the network via the switch



Prepare the Experiments | Software

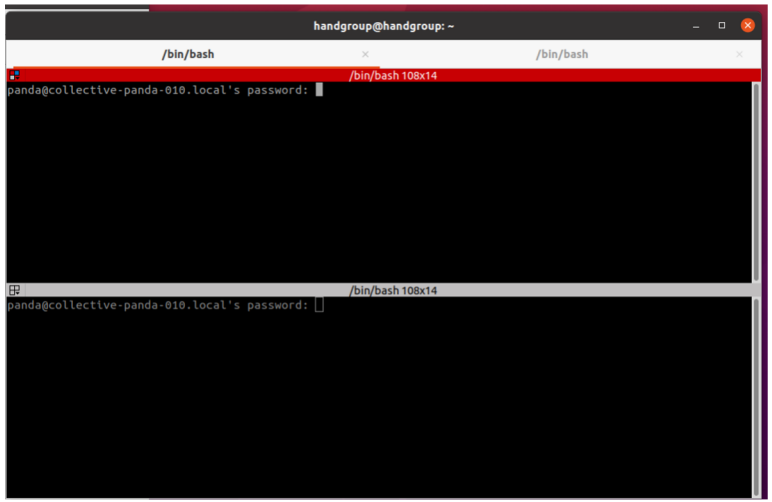
We first must connect to the NUC that will run MIOS and the Interface

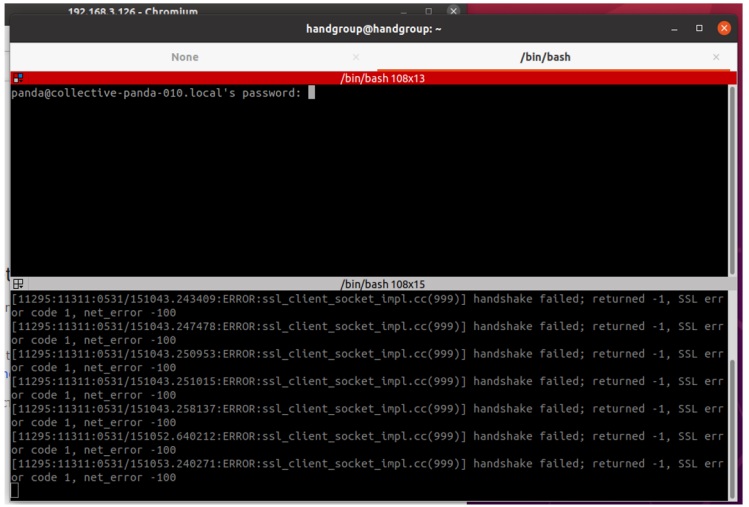
Open Terminal and execute
$ setup_launch
⇒ New window with 2 tabs opens



TAB1	TAB2
SSH_NUC	Port Forward
SSH_NUC	Chromium

SSH_NUC: To launch mios and the sensor experiment (requires password)
Port Forward: To access the robot (requires password)
Chromium: To access DESK





Experiment Manual | Software - Prepare the Robot

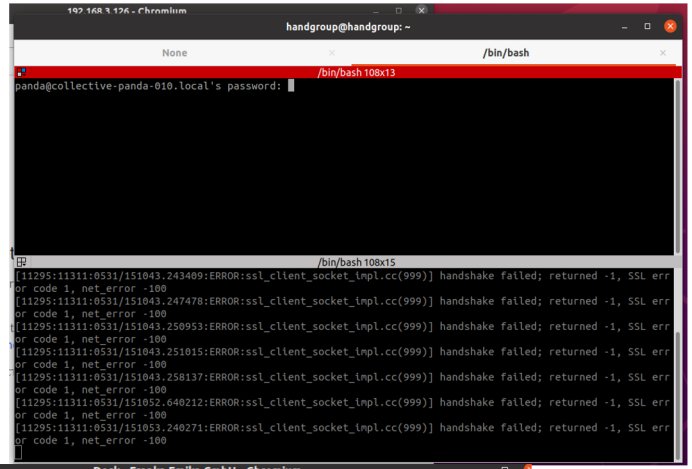
TAB2 - For using Desk

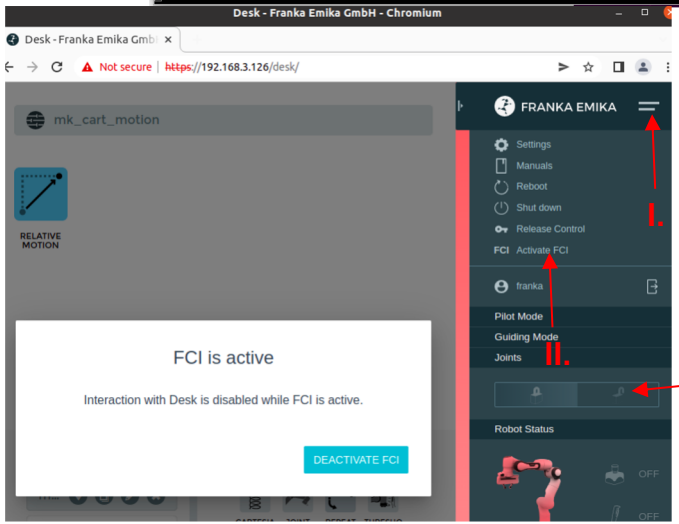
1. Enter password to robot via ssh
⇒ Port forwarding starts. Nothing else happens in that window
⇒ DESK should appear in the
chromium window that opened.
If it does not work reload and ignore (not secure), turn wifi off
2. Unlock the robot joints by clicking on the unlock button and confirming
3. Release safety switch on robot so that the robot light turns blue
4. Activate FCI by following steps I. & II.
⇒ FCI is activated, nothing else needs to be done here
⇒ Will only be needed for shutdown

LIFT ROBOT ARM BEFORE SHUTDOWN to avoid damaging the tactile sensors

(robots tilts downwards once the joints lock)

⇒ Only minimize window, don't close it

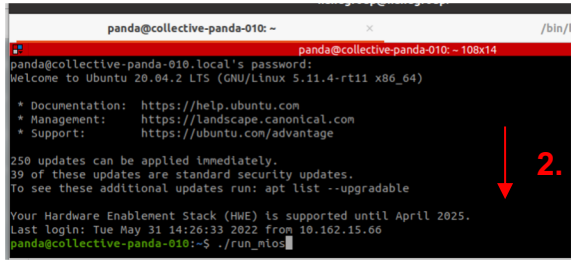


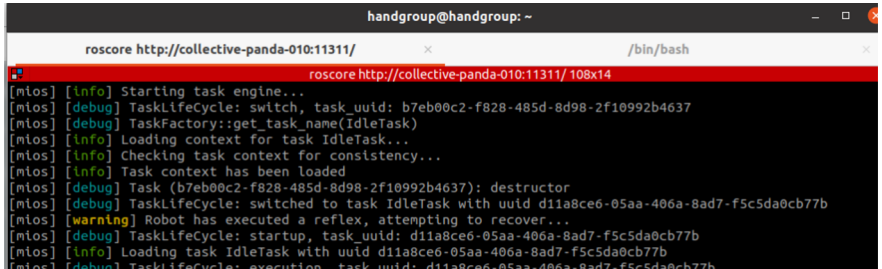


Experiment Manual | Software - MIOS

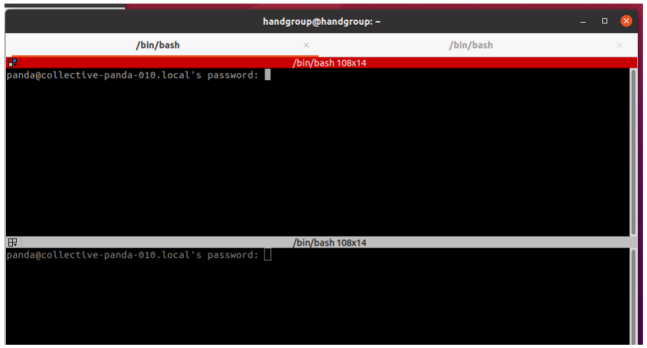
1. Login via ssh.
⇒ NUC can be used

2. Start MIOS in upper window with
$ ./run_mios.sh
⇒ Wait until “Executing task Idletask…”
(In case of an error, cancel command with STR+C and execute command again)
3. run $./send_sensor_data.sh (10000s)
4. in new terminal: $ ssh panda IP (panda pw)
5. run $./sensor_experiment.sh (carefull, robot moves!!)
(then ready for python scripts)

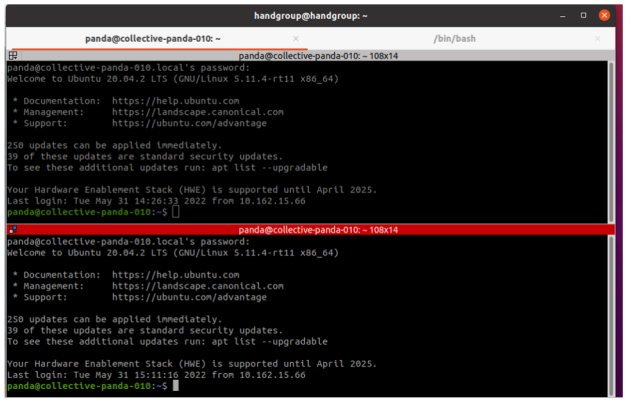




TAB1 - For conducting the experiments



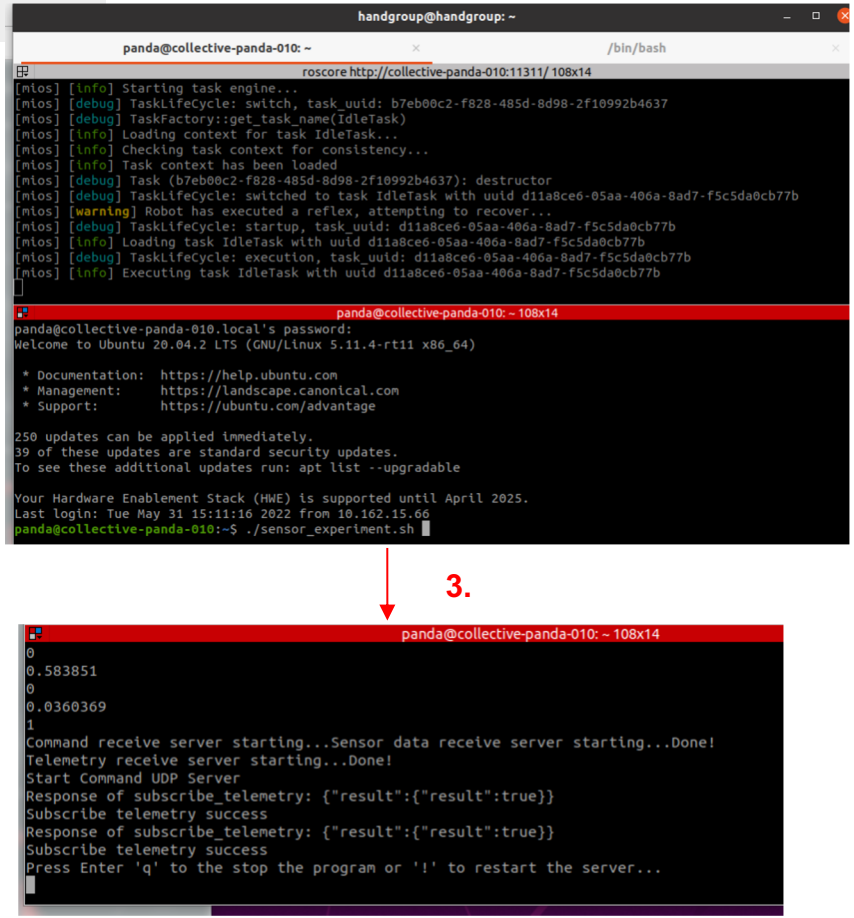
1.



Experiment Manual | Software - Interface

3. Launch the Interface in the lower window with
$ ./sensor_experiment
ROBOT WILL MOVE TO INITIAL POSITION! Careful!

($ ./sensor_experiment 1
for sliding experiment with vertical mount)
⇒ Wait until “Press Enter ‘q’...
⇒ Interface is ready for commands
(In case of an error, cancel command with STR+C and execute command again)



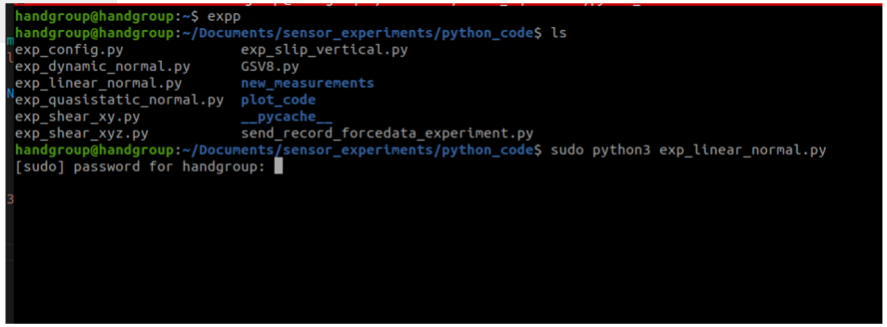
Run the Experiments | Software

4. Open new terminal window (STR+ALT+T / Menu)
5. Navigate to *~/Documents/sensor_experiments/python_code*

$ cd *~/Documents/sensor_experiments/python_code*

6. Execute the script corresponding to the experiment with

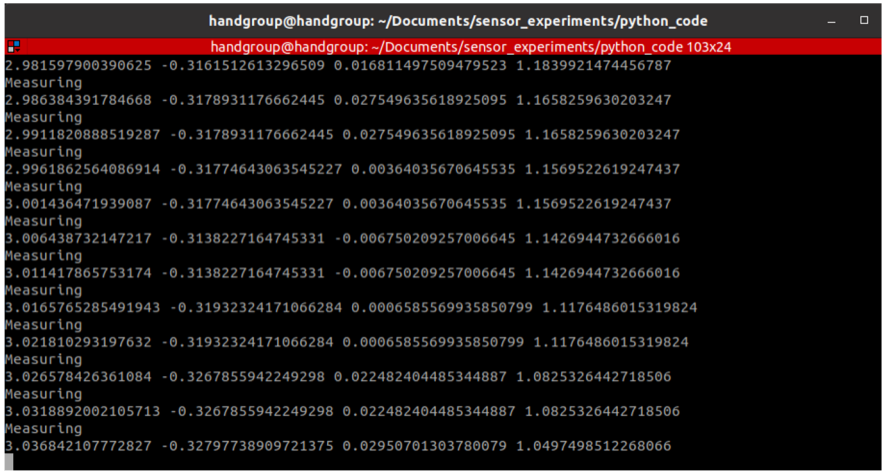
$ sudo python3 SCRIPTNAME.py



example:

$ sudo python3 exp_linear_normal.py

⇒ Script starts (after entering system password “handgroup2022”)
⇒ Timer F_x F_y F_z are printed in console
⇒ Starts recording once a certain force threshold is reached ⇒ “Measuring” is shown

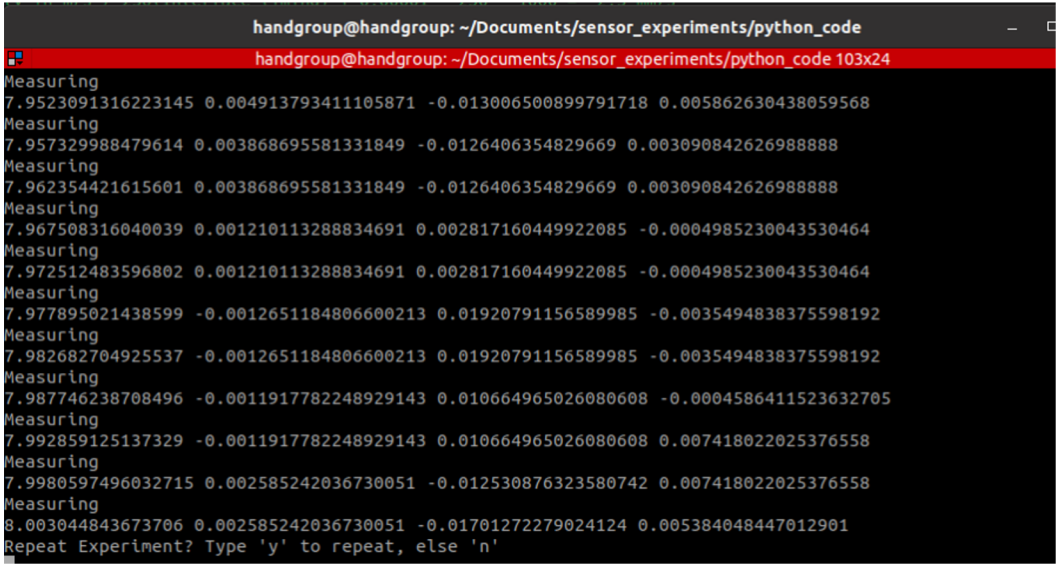


7. “Repeat Experiment…” is shown
8. Write “y” if experiment should be repeated with same parameters
9. Write anything else if not.

⇒ Script ends

10. You can now execute a different experiment script

In case it is not working, restart the interface in the corresponding window ⇒ slide 6



Inside the Code | Software

Two common files (used for generic parameters and data recording):

exp_config.py: Generic Parameters like IP address of the laptop in the network, NUC IP , UDP Ports…

```
#Network Parameters
HOST_IP = "10.162.15.200"    #IP of the computer running this script
NUC_IP = "10.162.15.46"      #IP of NUC that runs MIOS-Interface

UDP_PORT_RECEIVE_OK = 6665   #Port where script receives answer if robot has finished
UDP_PORT_RECEIVE_TEL = 6666  #Port where script receives telemetry

UDP_PORT_SEND_CMD = 3332     #Port where MIOS-Interface listens for commands
UDP_PORT_SEND_FORCE = 3333   #Port where MIOS-Interface listens for force data
```

HOST_IP has to be adapted when switching control computer
NUC_IP has to be adapted if anything changes with the NUC setup

send_record_forcedata_experiment.py: Handles data recording, UDP sending (UDP Network Protocol)

```
FORCE_THRESHHOLD = 0.02
```

Determines the force threshold in Newton

```
BASE_DIR = "./new_measurements/"
```

Determines the directory for saving measurement data

You can change the experiment parameters at the beginning of each python experiment file.

exp_linear_normal.py: Example of parameters

```
13   #Experiment Parameters
14   REPETITIONS = 1          #Amount of Cycles
15   FORCE_VAL = 5            #Target Force in Newton
16   FEED_RATE = 0.00001      #Target Velocity in m/s / 250(Interface timing) | 0.00001 * 250 * 1000 =  2.5 mm/s
17   INITIAL_WAIT = 0         #Waiting Time in Seconds
18   DATA_RECORD_TIME = 8     #Duration for Data Record Loop in Seconds (Better too high than too low)
```

FORCE_VAL should be between 1N and 15N
FEED_RATE should be between 0.0001 and 0.000001 (faster than 0.0001 is fast and might damage sensor)

Note that DATA_RECORD_TIME defines how long the data is recorded after the force threshold is reached and STOPS sending data afterwards.
Therefore it might happen that the robot stops receiving force data even though it is still moving if this time is too chosen too small.

⇒ Set it to estimated experiment duration + 10% as buffer

```
time_short = time.strftime("%b%d", time.localtime())
dataName = f"{time_short}_linearNormal_{FORCE_VAL}N_FR_{FEED_RATE}_{DATA_RECORD_TIME}s"
```

dataName defines how the measurement data names are composed for each experiment

Experiment Manual | Software - Change Parameters

Main loop is similar for every experiment script.
Commented code for clarification.
Further description is found in the *exp_linear_normal.py* file

```
if __name__ == '__main__':


    while True:

        record_timer = time.time()
        p = subprocess.Popen(f"python3 ./send_record_forcedata_experiment.py {DATA_RECORD_TIME} {dataName}", shell=True)    #Execute scri

        print('Initial wait...')
        time.sleep(INITIAL_WAIT)


        for val in forceValues:

            cmd = f"F {val} R {FEED_RATE} Back"

            SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))      #Send the command string to the Realtime-PC
            waitForResponse(SOCK_RECEIVE)                                    #Wait for Robot to finish task

        p.wait()                                                             #Wait for data recording to end

        rep = repeatExp()                                                    #Ask if experiment should be repeated

        if not rep:
            break                                                                                    #End loop and terminate script


    print("----------")
    print("Now terminating.")
```

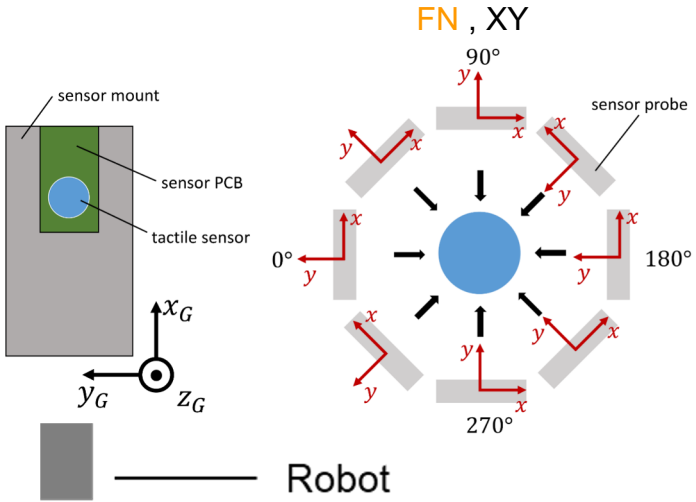
Commands that can be sent from python to the interface via UDP

Normal Force & Sliding Commands	Description
F 5 Back	Normal Force to 5N, return to start
F 5 Stay	Normal Force to 5N, stay
F 5 R 0.00001 Back	Normal Force to 5N, with feed rate 0.00001, return to start
F 5 R 0.00001 Stay	Normal Force to 5N, with feed rate 0.00001, stay
F 5 R 0.00001 C Back	Normal Force to 5N, with feed rate 0.00001, cyclic test allows fast consecutive repetitions (For dynamic cyclic experiment)
S 5 Stay	Normal Force to 5N, stay (Has to be executed before “S 0.001 Slip)
S 0.0001 Slip	Start slip motion with a feed rate of 0.0001

⇒ Forces can be adapted from 1 to about 15N
⇒ Rate can be anything between 0.001 (should not go higher, as sensor might get damaged) and 0.000001 (very slow)

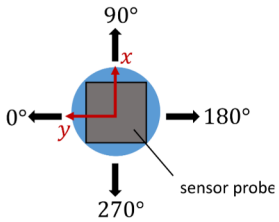
⇒ Feed rate of 0.00001 ⇒ Should result in a speed of 0.00001m/s * 1000 * 250 = 2.5 mm/s (Is not always reached, slower)
“1000” to convert from m to mm, “250” due to the interface timing
(0.0001 is actually the increment with which the position is increased and ~250 results from the timer in the interface loop)

Shear Force Commands	Description
F 5 Angle 45 Back	Apply 5 N tangential on sensor surface, 45° angle, return
F 5 Angle 45 Stay	Apply 5 N tangential on sensor surface, 45° angle, stay
FN 5 R 0.0001 Angle 45 Back	Apply 5N to side of sensor, rate of 0.0001, angle of 45°, return
FN 5 R 0.0001 Angle 45 Stay	Apply 5N to side of sensor, rate of 0.0001, angle of 45°, stay



FN , XY

F , XYZ



Commands for XYZ experiments have to be run after a “normal force stay” command
⇒ Example in python script for XYZ experiments

You can use the previous commands to create the experiments

exp_linear_normal.py: Example of code

```
cmd = f"F {val} R {FEED_RATE} Back"

SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))
waitForResponse(SOCK_RECEIVE)
```

In this case only one command is sent and the script awaits feedback from the robot that it has executed the command

```
while True:

    for val in forceValues:

        cmd = f"F {val} R {FEED_RATE} Back"

        SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))
        waitForResponse(SOCK_RECEIVE)

    p.wait()

    rep = repeatExp()

    if not rep:
        break
```

The loop is repeated if the repeat question is answered with “y” in the terminal

Experiment Manual | Software - Use commands

You can chain commands together to create the experiments

exp_quasistatic_normal.py: Example of code

```
cmd = f"F {step} R {FEED_RATE} Stay"

SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))
waitForResponse(SOCK_RECEIVE)

print(f"Holding {step} N for {STEP_WAIT} seconds...")
time.sleep(STEP_WAIT)

SOCK_SEND.sendto(f"F {step} R {FEED_RATE} Back".encode(), (NUC_IP, UDP_PORT_SEND_CMD))
waitForResponse(SOCK_RECEIVE)
```

1. Send STAY so that force is held
2. Wait for response that command is executed
3. Sleep for defined dwell time
4. Send BACK command to return to initial position

You can chain commands together to create the experiments

exp_shear_xyz.py: Example of code

```
cmd = f"F {FORCE_VAL_Z} R {FEED_RATE} Stay"

SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))
waitForResponse(SOCK_RECEIVE)


cmd = f"F {FORCE_VAL_XY} Angle {ANGLE} Back"

SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))
waitForResponse(SOCK_RECEIVE)


cmd = f"F {FORCE_VAL_Z} R {FEED_RATE} Back"

SOCK_SEND.sendto(cmd.encode(), (NUC_IP, UDP_PORT_SEND_CMD))
waitForResponse(SOCK_RECEIVE)
```

1. Send STAY so that normal force is held
2. Wait for response that command is executed
3. Send BACK to apply tangential force and return (back to position on the sensor)
4. Wait for response that command is executed
5. Send BACK to deload normal force and return (back to initial starting position above sensor)

Useful Information | Extra

Experiment Manual | Recalibrate starting position

This step is about acquiring the current T matrix and joint configuration so that they may be adjusted on the NUC.

Open Visual Studio
Find file *my_print_T.py* in */Documents/sensor_experiments/python_code/ folder*

Move robot manually into desired calibration position
Hit safety switch so that the robot light is blue
Execute the *my_pirint_T.py* script in VSC (not working atm)
Console shows current **T-matrix** and **joint configuration**
Both are required for the calibration position
Proceed on next slide

Experiment Manual | Recalibrate starting position

This step is about editing the calibration position on the nuc

Connect to NUC via ssh
Navigate to configs folder
$ cd ~/Documents/sensor_experiment/Config/
Show all files
$ ls
You see 5 Points

Calibration Point above the sensor, starting point for normal and shear force experiments
T-Matrix of the point -> Robot_Calibration_Point_1.txt
Joint configuration of the point -> Robot_Calibration_Point_1_Joint.txt

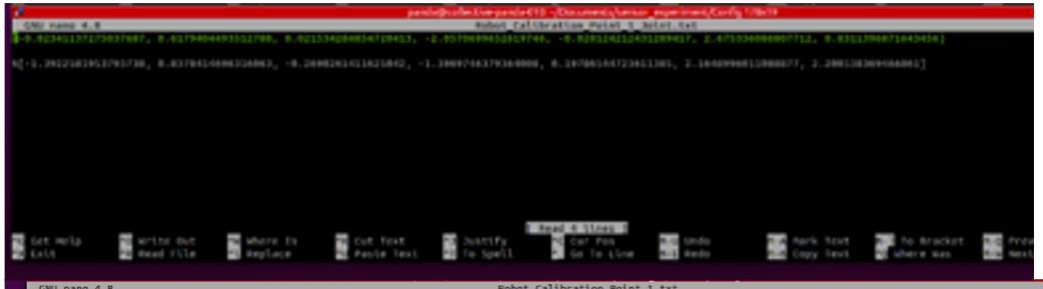
Calibration Point in slide configuration “above” the sensor
T-Matrix of the point -> Robot_Calibration_Point_2.txt
Joint configuration of the point -> Robot_Calibration_Point_2_Joint.txt

Calibration Point (Tool Center Point) of the point on the PCB surface (center of the tactile sensor)
Joint configuration of the point -> Robot_Calibration_TCP.txt

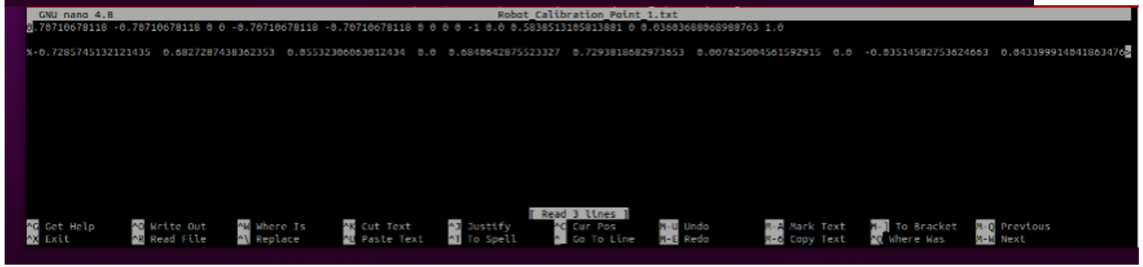
Proceed on next slide

Experiment Manual | Recalibrate starting position

Open desired file with
$ sudo nano FILENAME
You see the current configuration



Joint config



T-Matrix

The new values have to be in the SAME FORMAT as the current configuration

Paste corresponding array into file
Make sure you changed ***both joint and T-Matrix*** if you change either Point1 or Point2
Restart MIOS and the interface

setup_launch

./run_mios

./sensor_experiment

cd *~/Documents/sensor_experiments/python_code*

sudo python3

sudo python3 exp_linear_normal.py