mini_Eggshell

MAS T1 | 28.10.2019

Content

Introduction

Eggshell

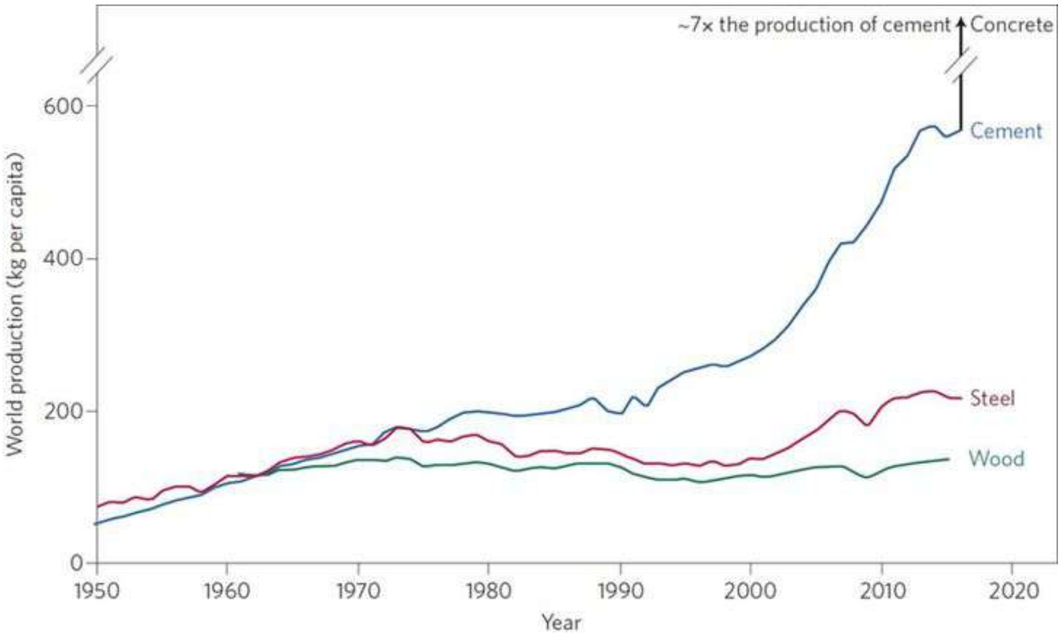
Previous Miniprojects

Assignment

Schedule

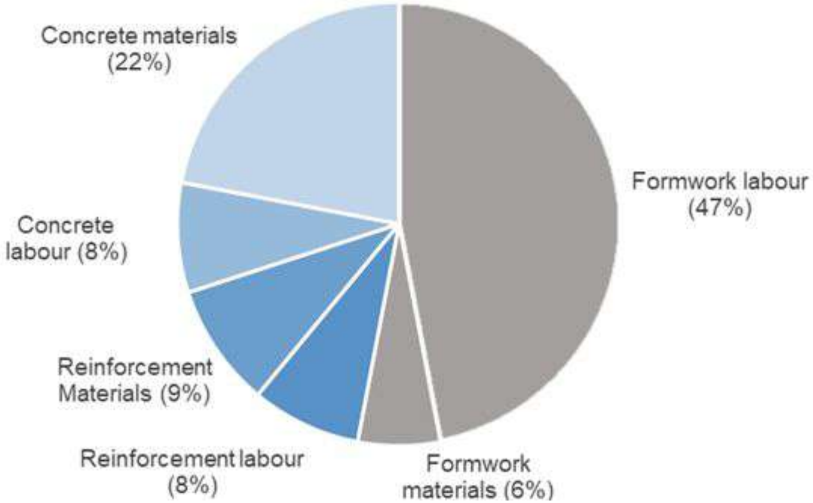
Ur online control

Introduction



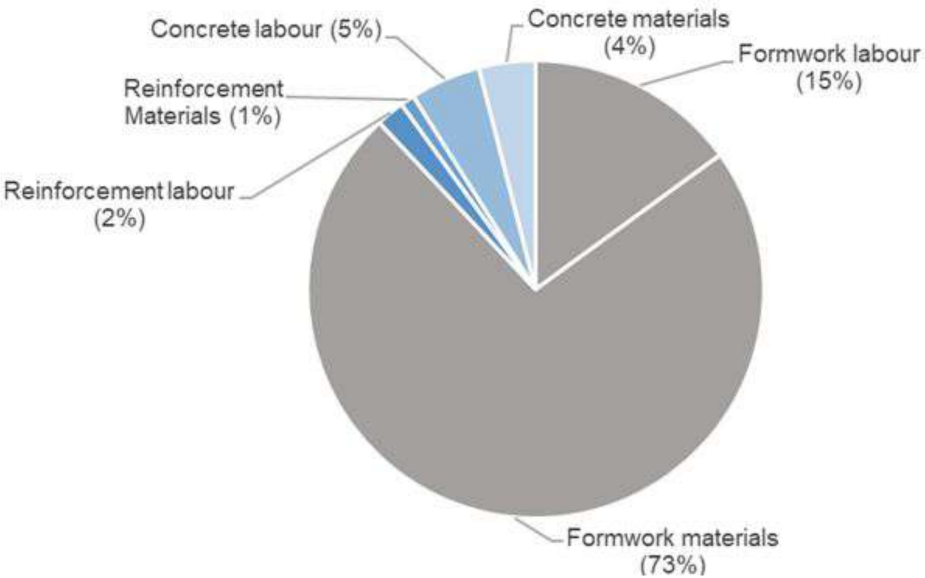
Paulo, Monteiro, Sabbie, Miller and Horvath (2017)

Cost of a standard concrete element

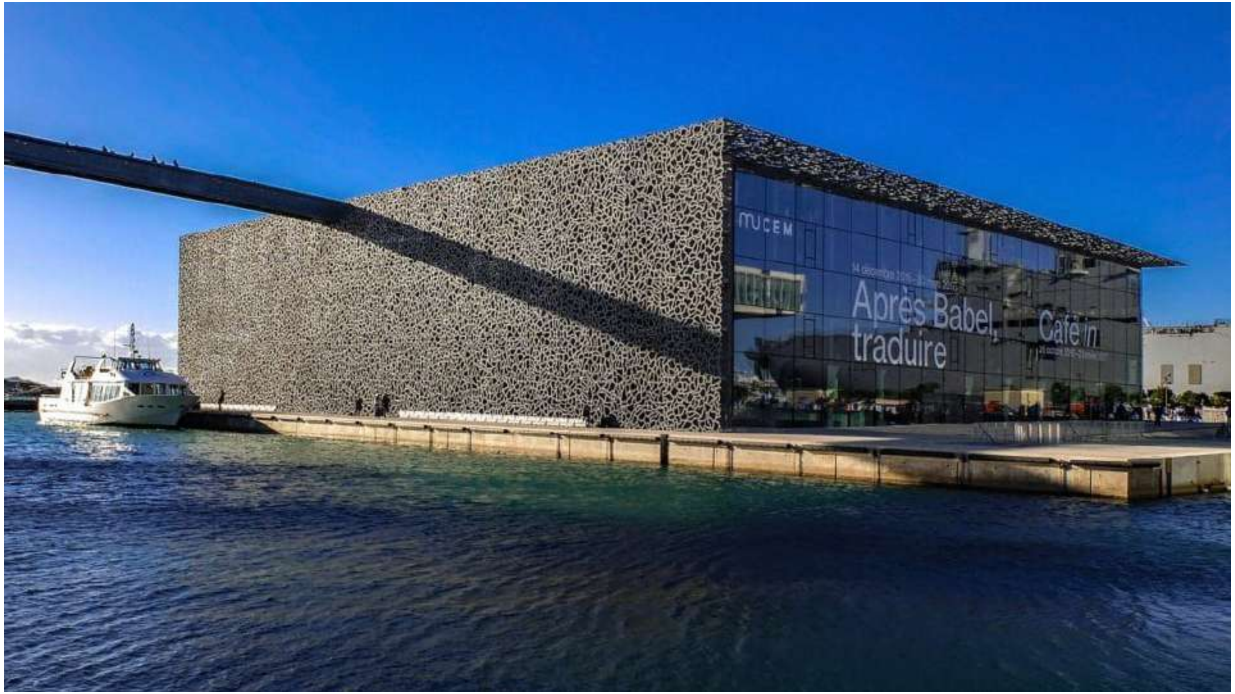


Lab (2007)

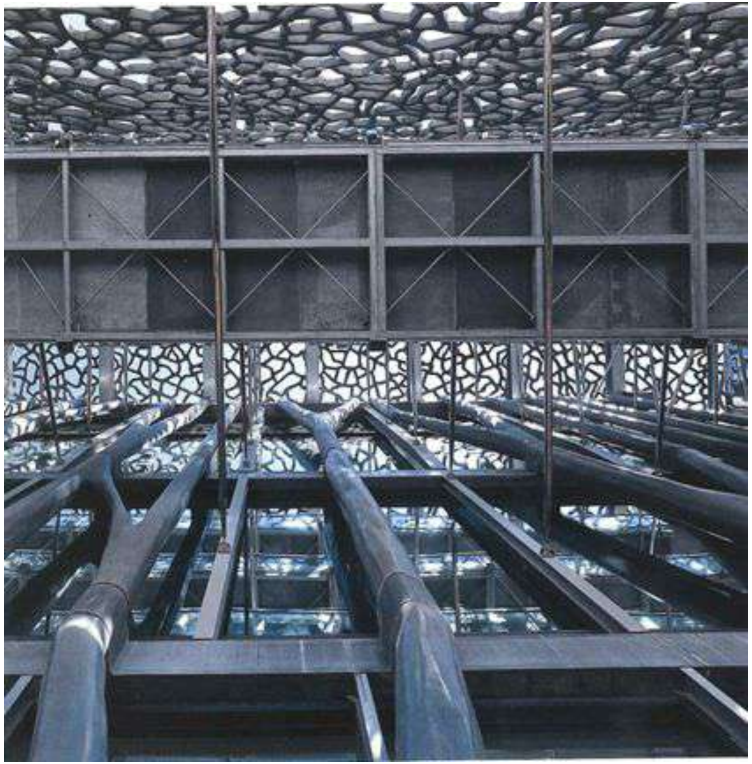
Cost of a non-standard concrete element

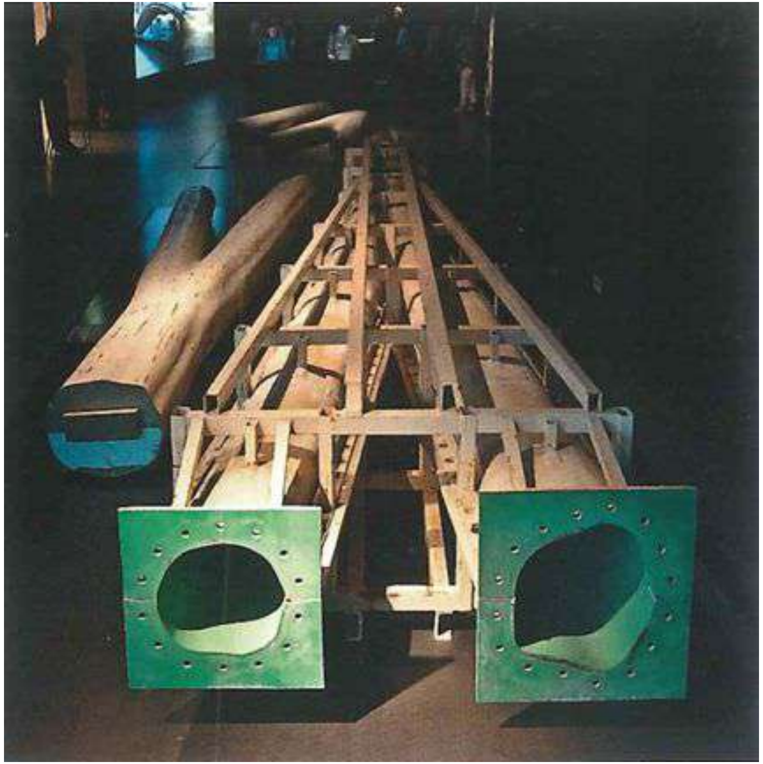


Schipper and Grünewald (2014)



MuCEM Marseille - Rudy Ricciotti (2013)
https://generationvoyage.fr/visiter-mucem/

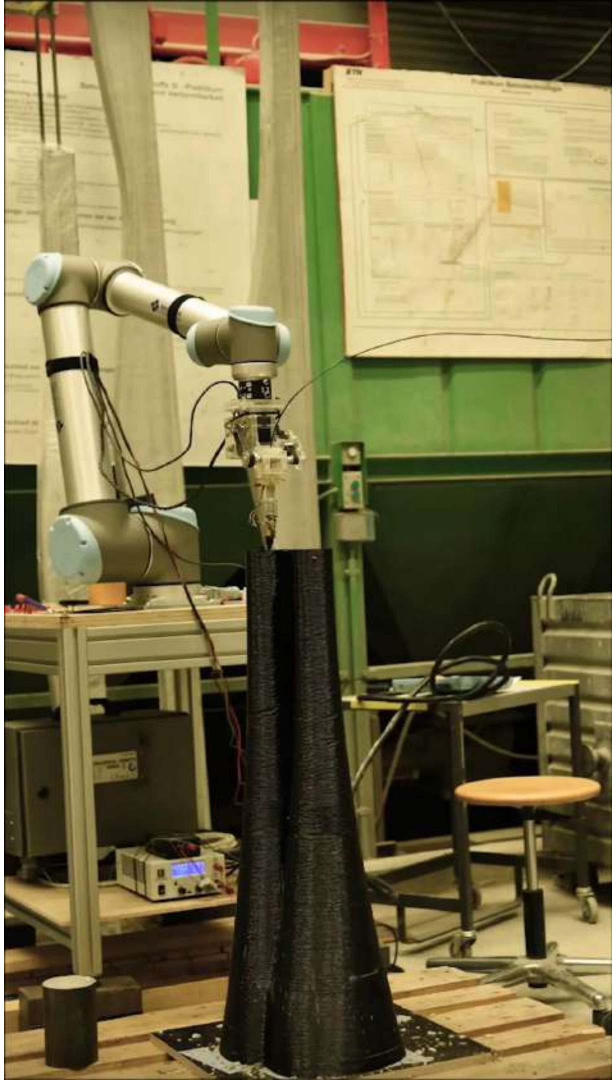




best of DETAIL 8 Beton/Concrete, Christian Schittich (2016)

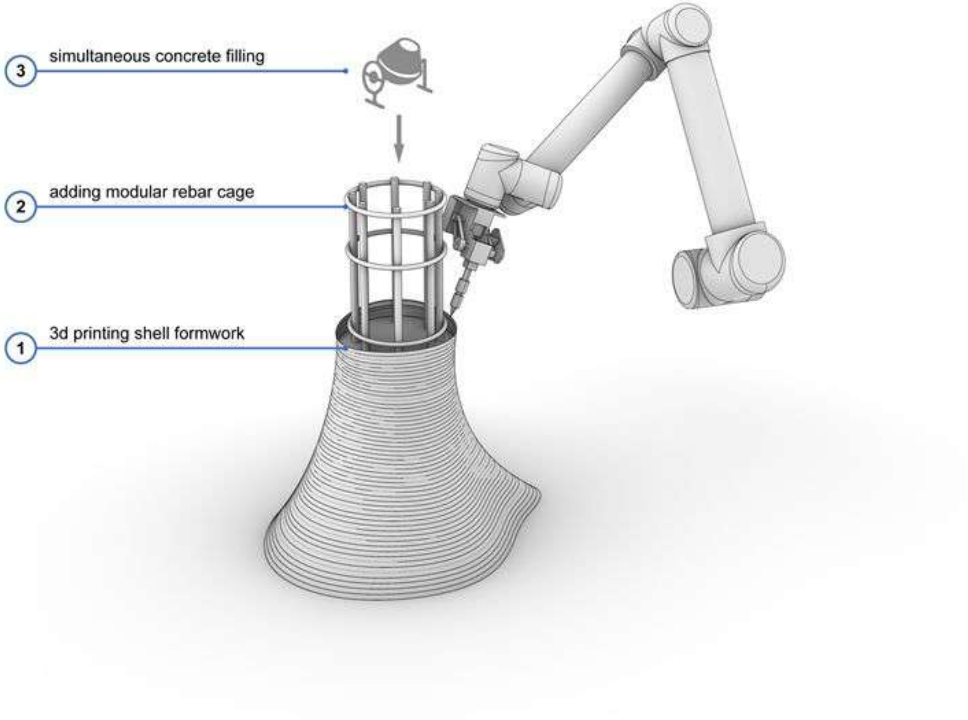
Eggshell concept

Ulrich (2017)



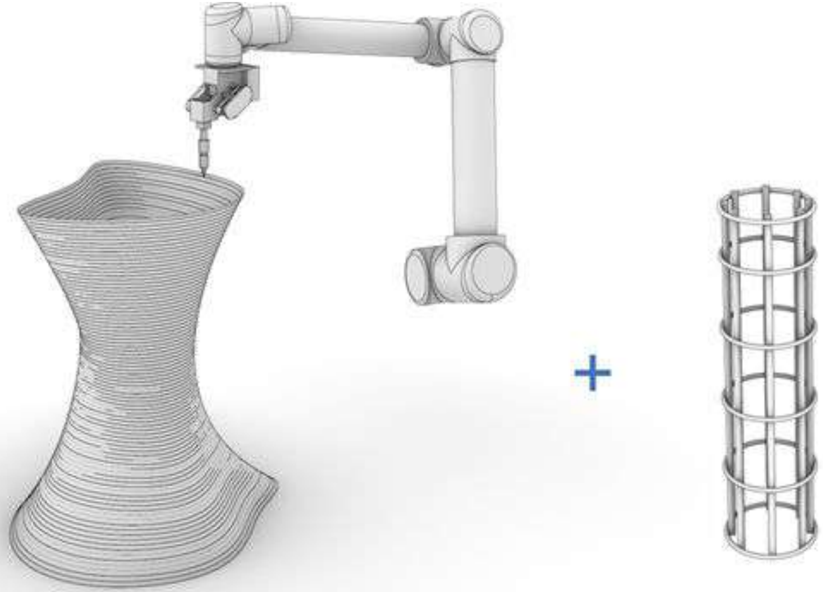
Gramazio Kohler Research
ETH Zurich

Simultaneous printing & filling

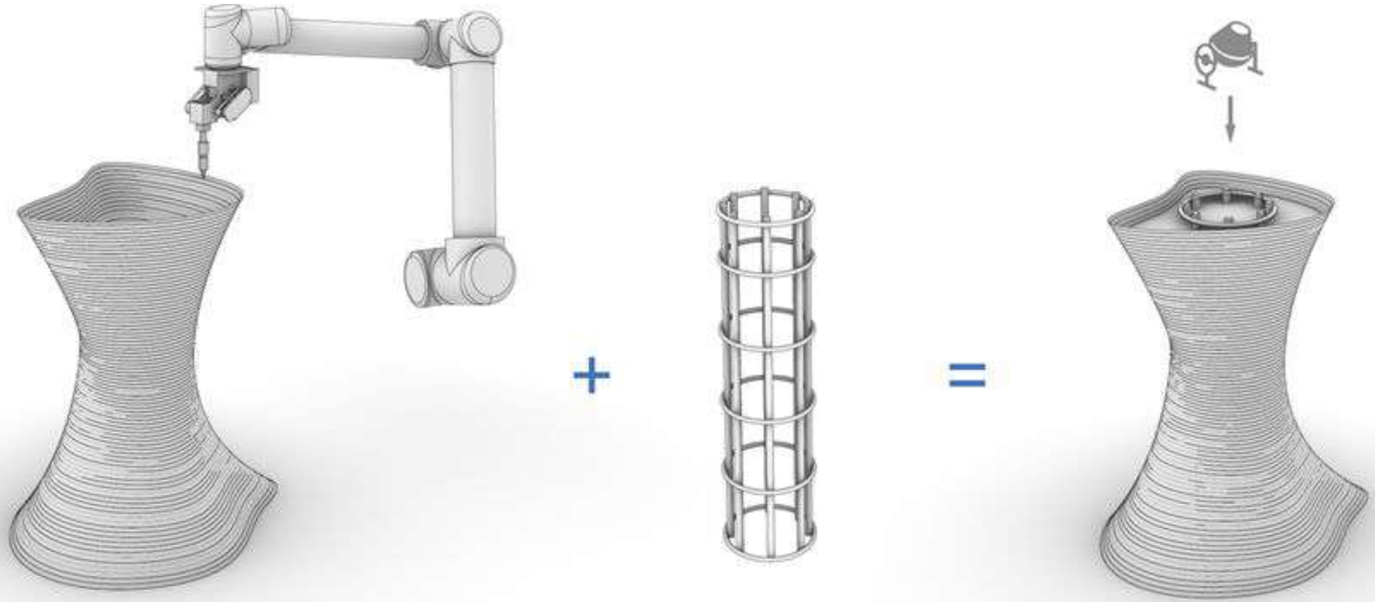

3 simultaneous concrete filling
2 adding modular rebar cage
1 3d printing shell formwork

Consecutive fabrication

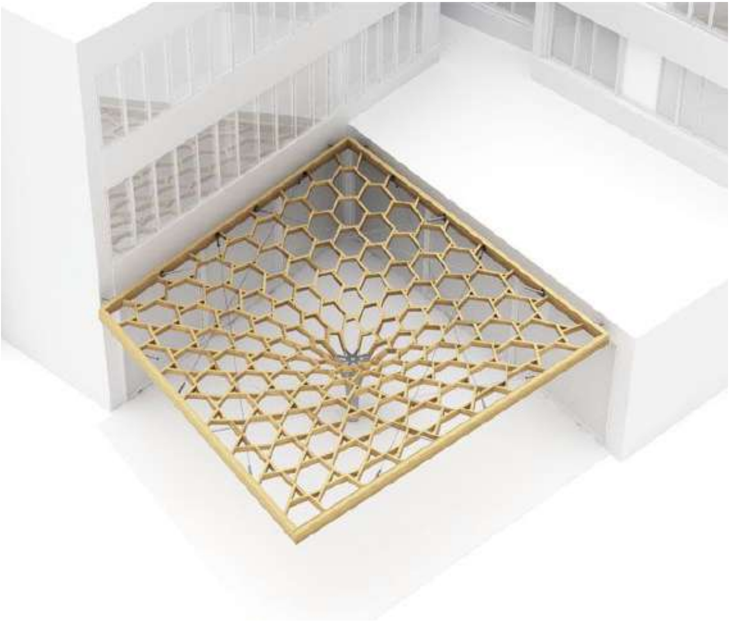
Consecutive fabrication

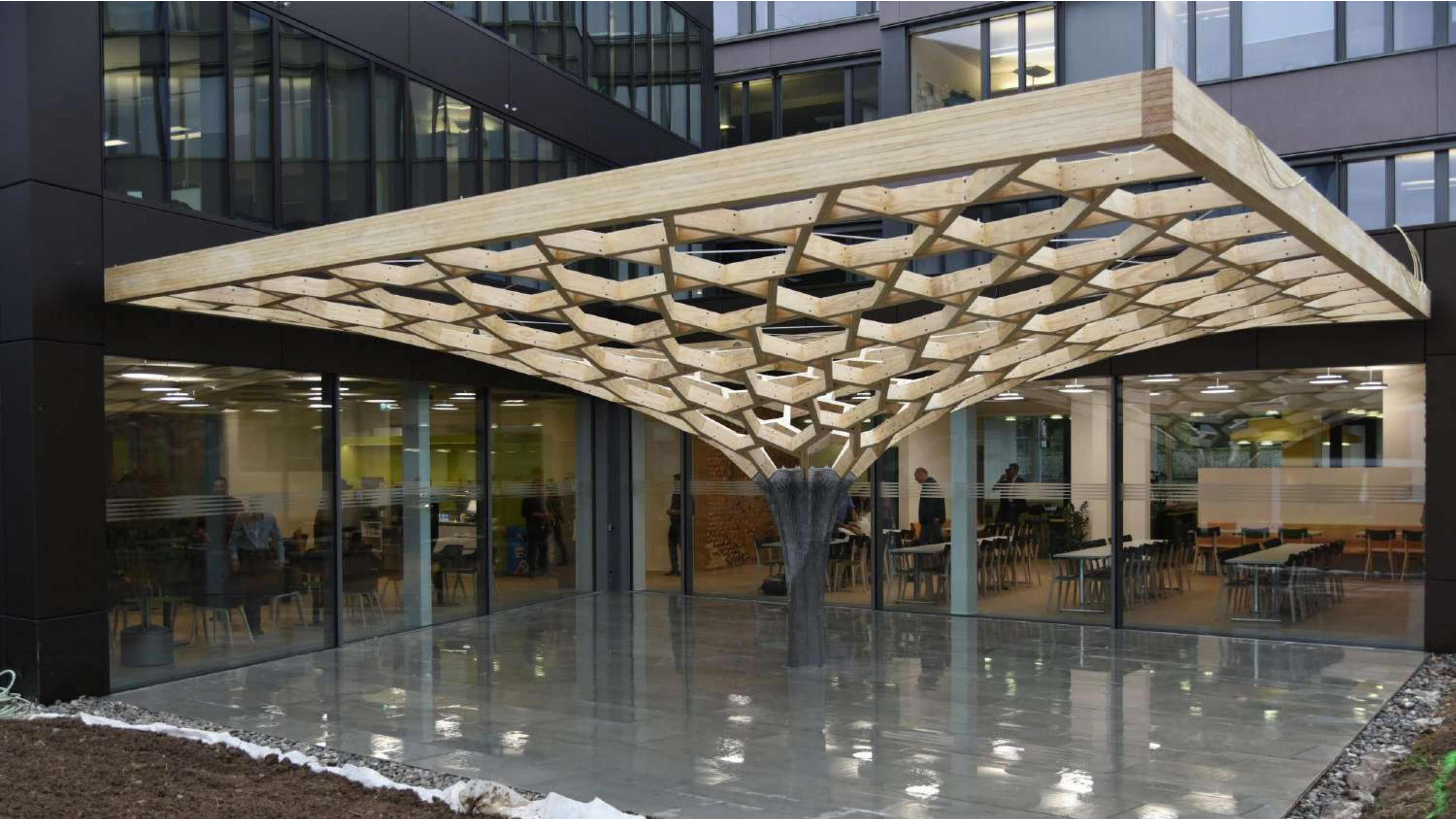


Consecutive fabrication



Basler & Hofmann



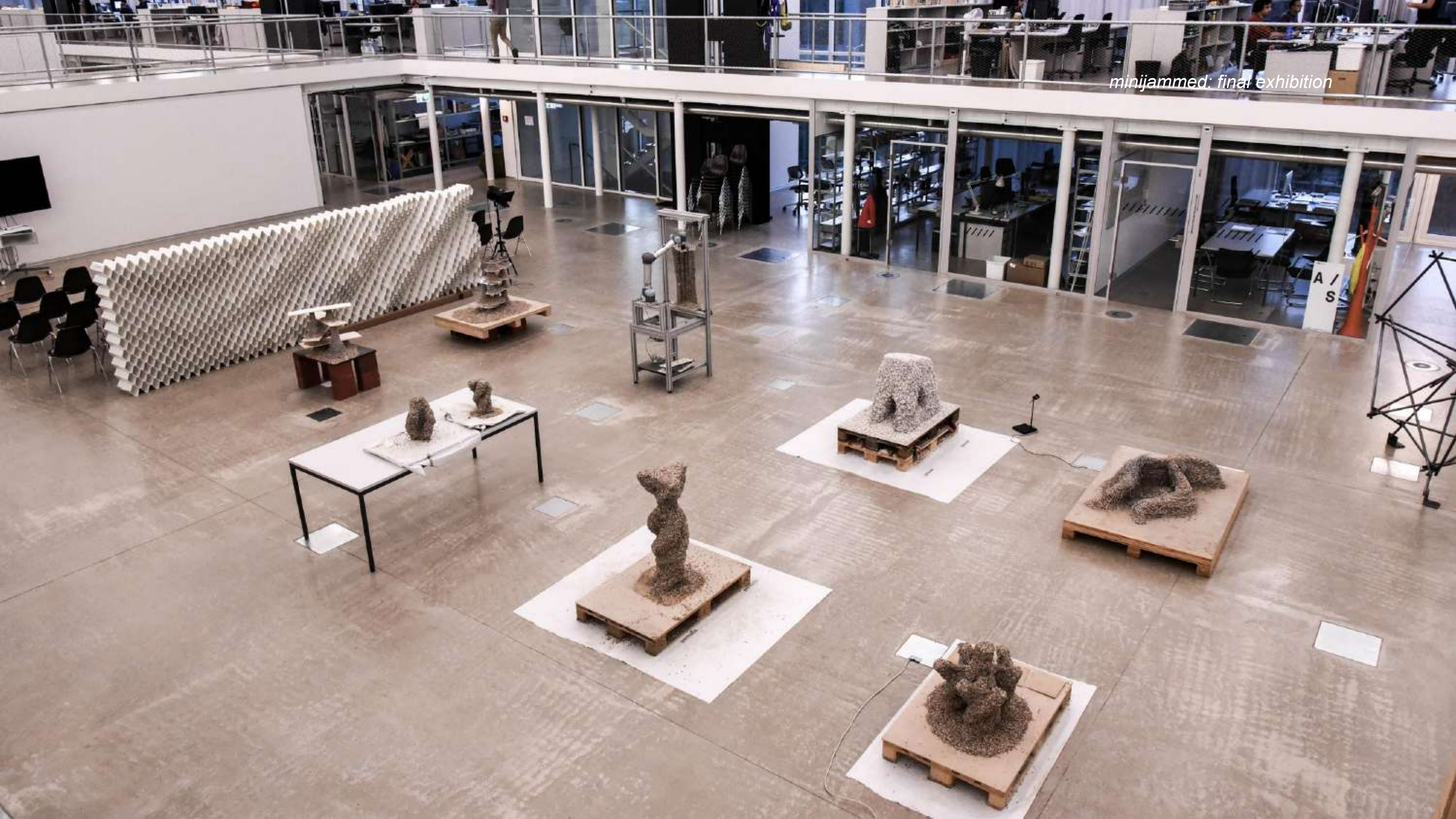


Yang & Barney (2019)

minijammed

minijammed is a three-week design and build assignment based on the ongoing research project Jammed Architectural Structures. The research focuses on the development of design and fabrication techniques to build fully reversible architectural structures by interlacing crushed-rock stones with textile string without any permanent bonding. The potential of the concept has been proven at architectural scale through the realisation of larger prototypes: Rock Print and Rock Print: A Manistone. minijammed invites the students to explore the design possibilities and the rich solution space of the method in a smaller scale.

The short-term project suggests a material-driven design and a robotic fabrication strategy, as only through the execution of multiple physical experiments, both manual and robotic, the designer is able to take informed decisions about the final outcome of the process.



minijammed: final exhibition

Okawa, Ruangjun, Jeong, 2017

Pastrana, Su, Lin, Yoo, 2017

Taha, Mitroupolou, 2017

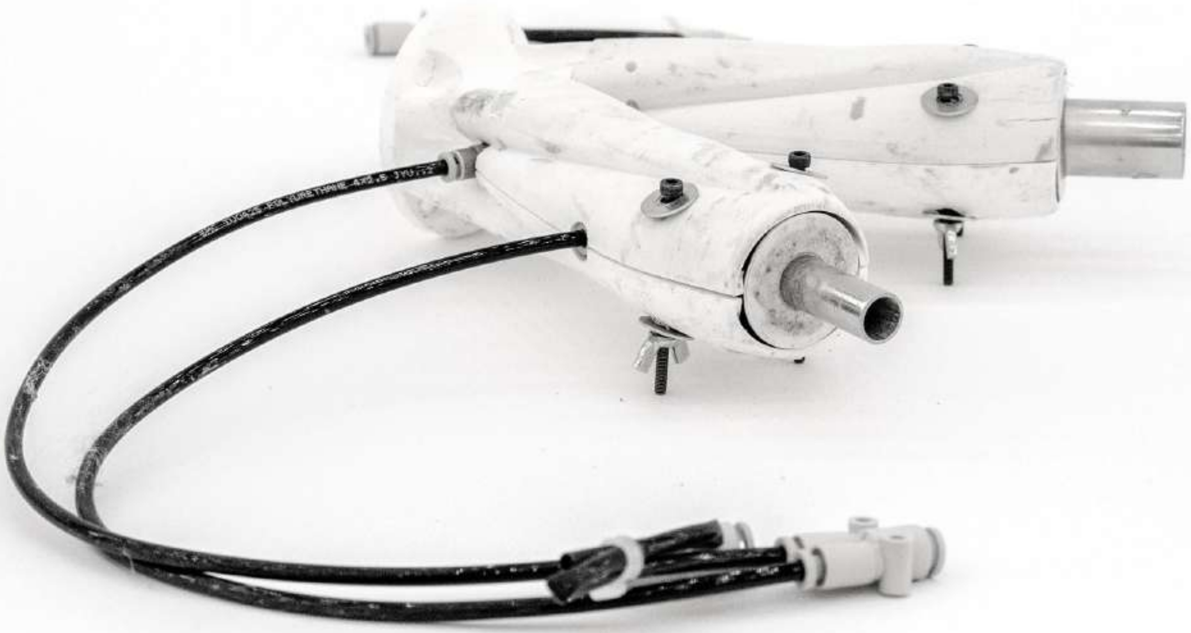
Cena, Chousou, Wang, Wu, 2017

Rapid Clay Formations

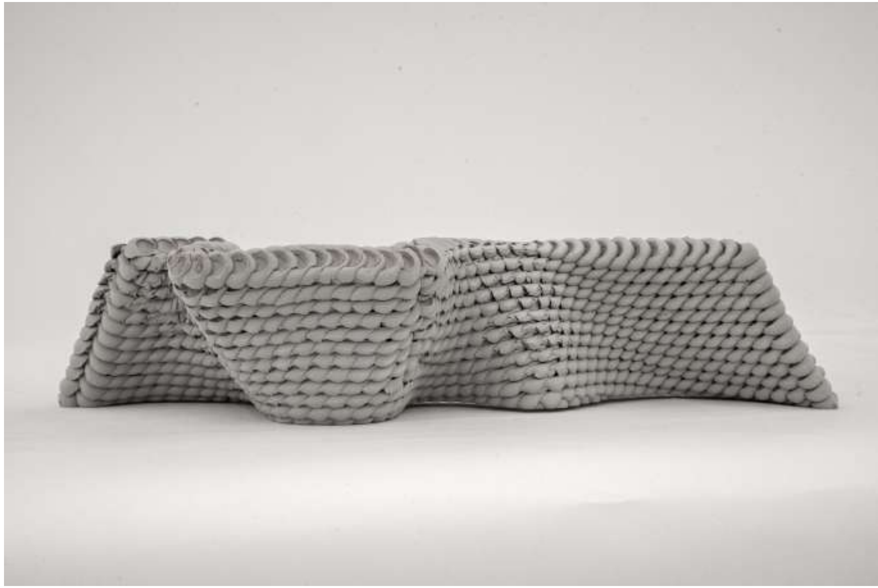
Rapid Clay Formations is a four-week design and build assignment investigating a novel fabrication process for malleable materials. Starting with the Remote Material Deposition installation in 2014, the Chair of Architecture and Digital Fabrication has investigated the idea of robotically positioning material in space from a distance and thereby creating differentiated architectural aggregations that are a direct expression of a dynamic and adaptive fabrication process. Continuing this concept, more recent projects developed within the MAS programme shifted the focus to the study of robotic fabrication processes for malleable materials, where the precise control of forces and positions applied to the material to design and build highly differentiated structures.

Rapid Clay Formations: final presentation

Rapid Clay Formations: double-bullet tool



Akizuki, Barney, Du, 2018



Chen, Skevaki, Yang, 2018

v.d.Bulcke, Girish, Eftekhar, 2018

mini-eggshell: Assignment

Use the Eggshell simultaneous printing & filling process to produce various building components.

Two formwork material options

- Clay - 3 setups
- Thermoplastic - 3 setups

Different building elements

- Beam
- Slab
- Wall
- Interface element (column - beam or wall - slab)
- Column

mini-eggshell: Goals

Focus on:

- Structure (reduce material/ weight, precise guides for reinforcement)
- Function (integrate different functions, lighting , media, electrical or mechanical services)
- Performance (integrate natural lighting and ventilation through transparency, porosity. Acoustics)
- Aesthetics (surface textures and ornamentation, architectural design, details)

3D-printed floor system, Block Research Group

Topology optimisation, Asbjørn Søndergaard / Odico Formwork Robotics

DIGITAL GROTESQUE II, Benjamin Dillenburger and Michael Hansmeyer

Elbphilharmonie, HdM

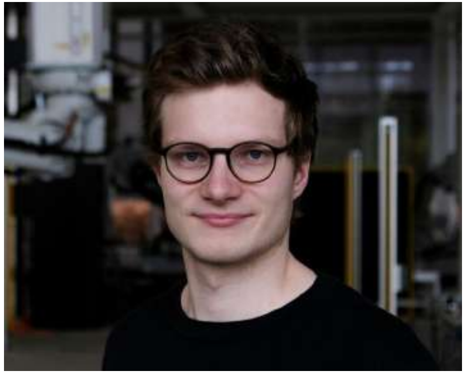
Schedule

https://docs.google.com/spreadsheets/d/181aZo1s0SwAvfoG4fKHYC5BHTWoBZZNCnyvJNnxfnJw/edit?usp=sharing

Guest Lectures

Dr. Lex Reiter (post-doc in PCBM)

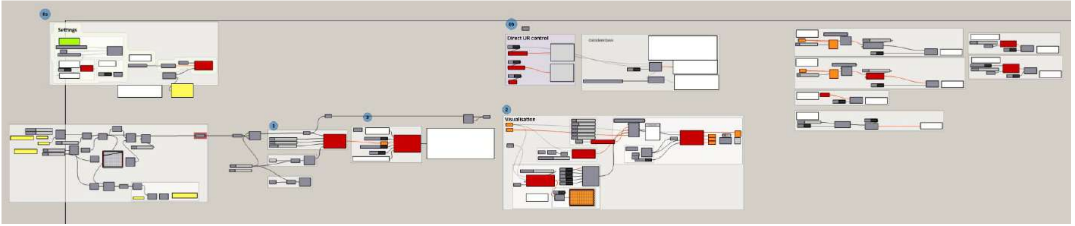
Lukas Gebhard (PhD student with IBK)

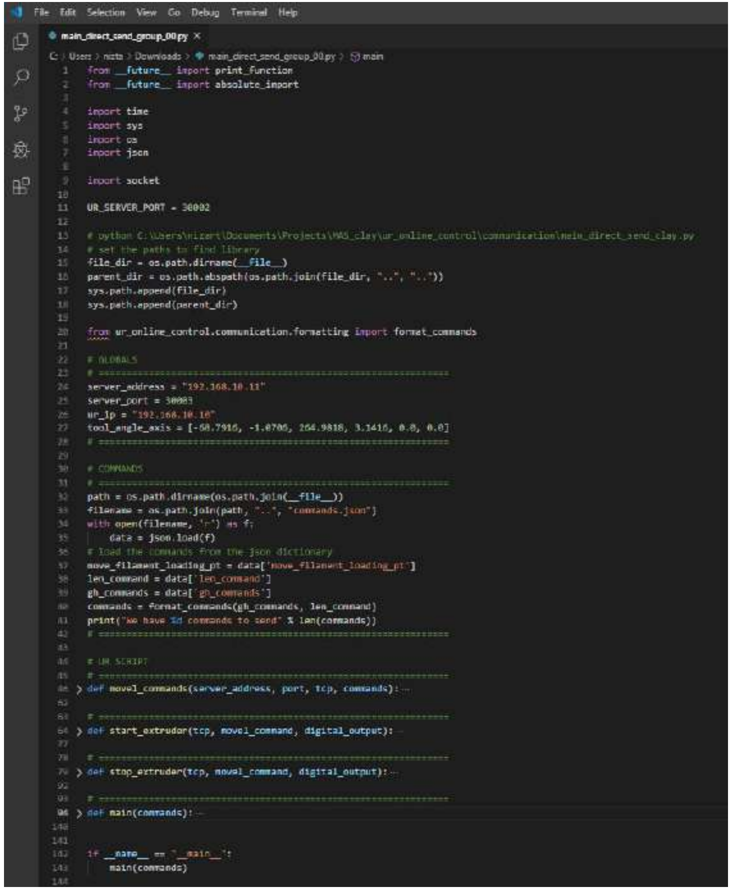


Dr. Thibault Demoulin (post-doc at PCBM and Oxara)

UR Online Control

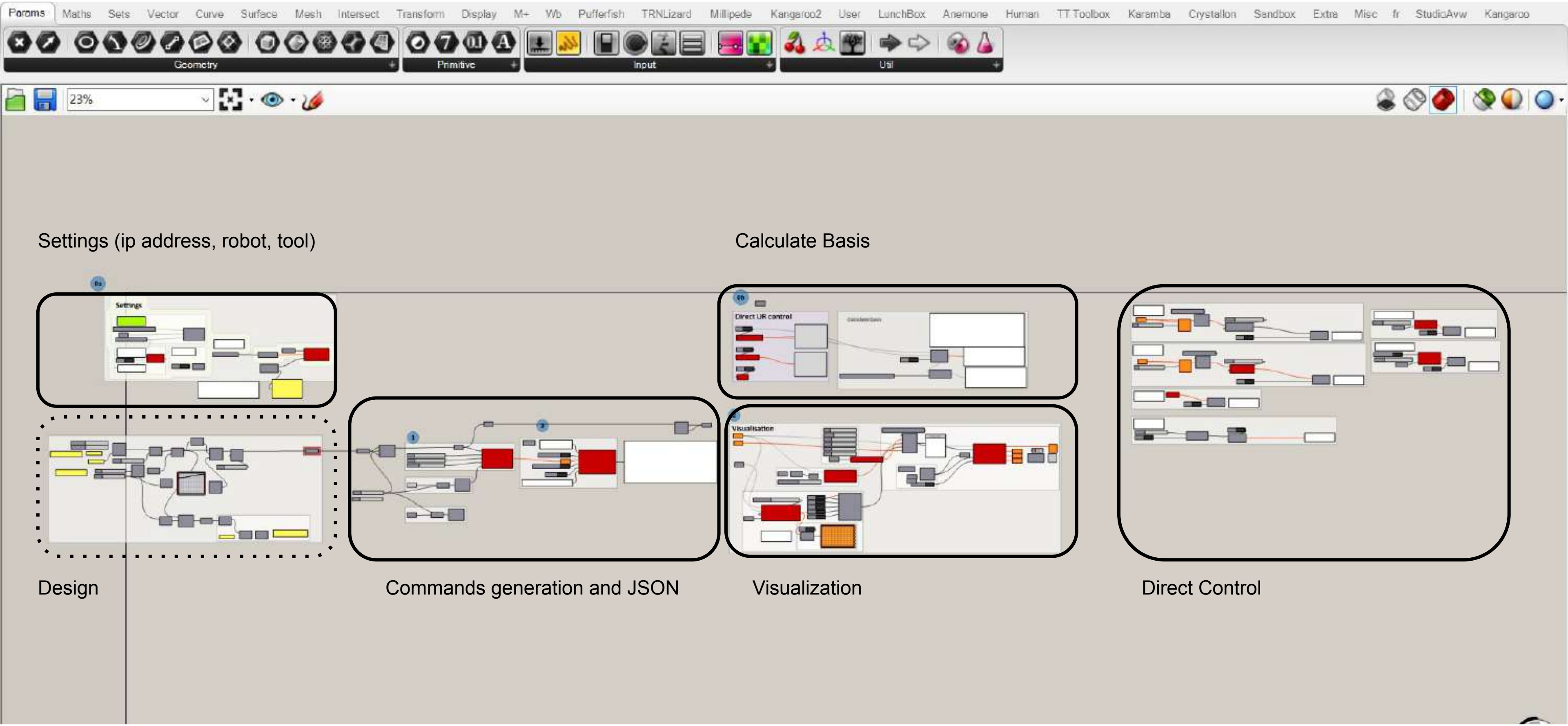
- Grasshopper
- JSON (https://en.wikipedia.org/wiki/JSON)
- Python main file




File Edit Selection View Go Debug Terminal Help
main_direct_send_group_00.py
C: > Users > nizte > Downloads > main_direct_send_group_00.py > main

```python
from __future__ import print_function
from __future__ import absolute_import

import time
import sys
import os
import json

import socket

UR_SERVER_PORT = 30002

# python C:\Users\nizert\Documents\Projects\MAS_clay\ur_online_control\communication\main_direct_send_clay.py
# set the paths to find library
file_dir = os.path.dirname(__file__)
parent_dir = os.path.abspath(os.path.join(file_dir, "..", ".."))
sys.path.append(file_dir)
sys.path.append(parent_dir)

from ur_online_control.communication.formatting import format_commands

# GLOBALS
# ===================================================================
server_address = "192.168.10.11"
server_port = 30003
ur_ip = "192.168.10.10"
tool_angle_axis = [-68.7916, -1.0706, 264.9818, 3.1416, 0.0, 0.0]
# ===================================================================

# COMMANDS
# ===================================================================
path = os.path.dirname(os.path.join(__file__))
filename = os.path.join(path, "..", "commands.json")
with open(filename, 'r') as f:
    data = json.load(f)
# load the commands from the json dictionary
move_filament_loading_pt = data['move_filament_loading_pt']
len_command = data['len_command']
gh_commands = data['gh_commands']
commands = format_commands(gh_commands, len_command)
print("We have %d commands to send" % len(commands))
# ===================================================================

# UR SCRIPT
# ===================================================================
def movel_commands(server_address, port, tcp, commands):...

# ===================================================================
def start_extruder(tcp, movel_command, digital_output):...

# ===================================================================
def stop_extruder(tcp, movel_command, digital_output):...

# ===================================================================
def main(commands):...


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

```

Gramazio Kohler Research
ETH Zurich



Settings (ip address, robot, tool)

Calculate Basis

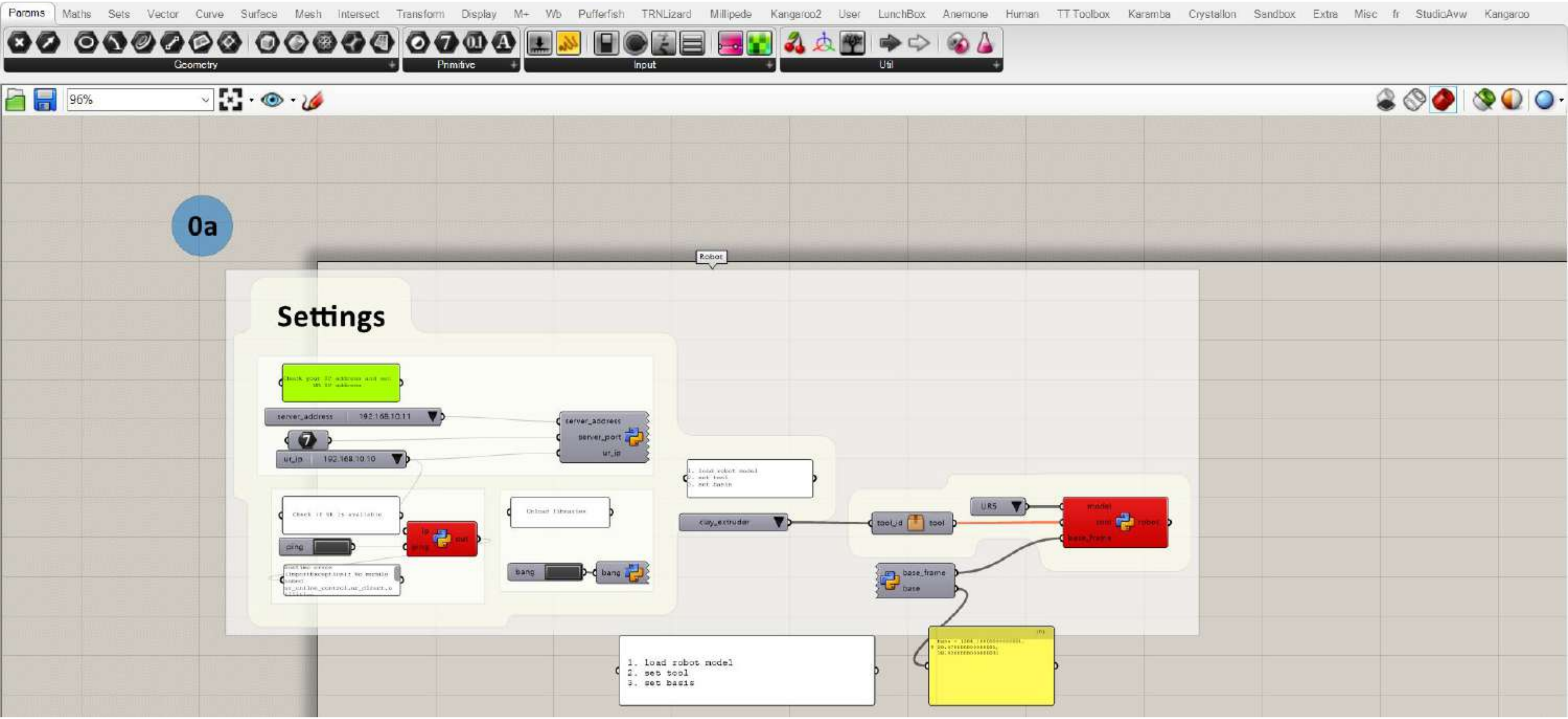
Design

Commands generation and JSON

Visualization

Direct Control

Gramazio Kohler Research
ETH Zurich



server_address 192.168.10.11

7

ur_ip 192.168.10.10

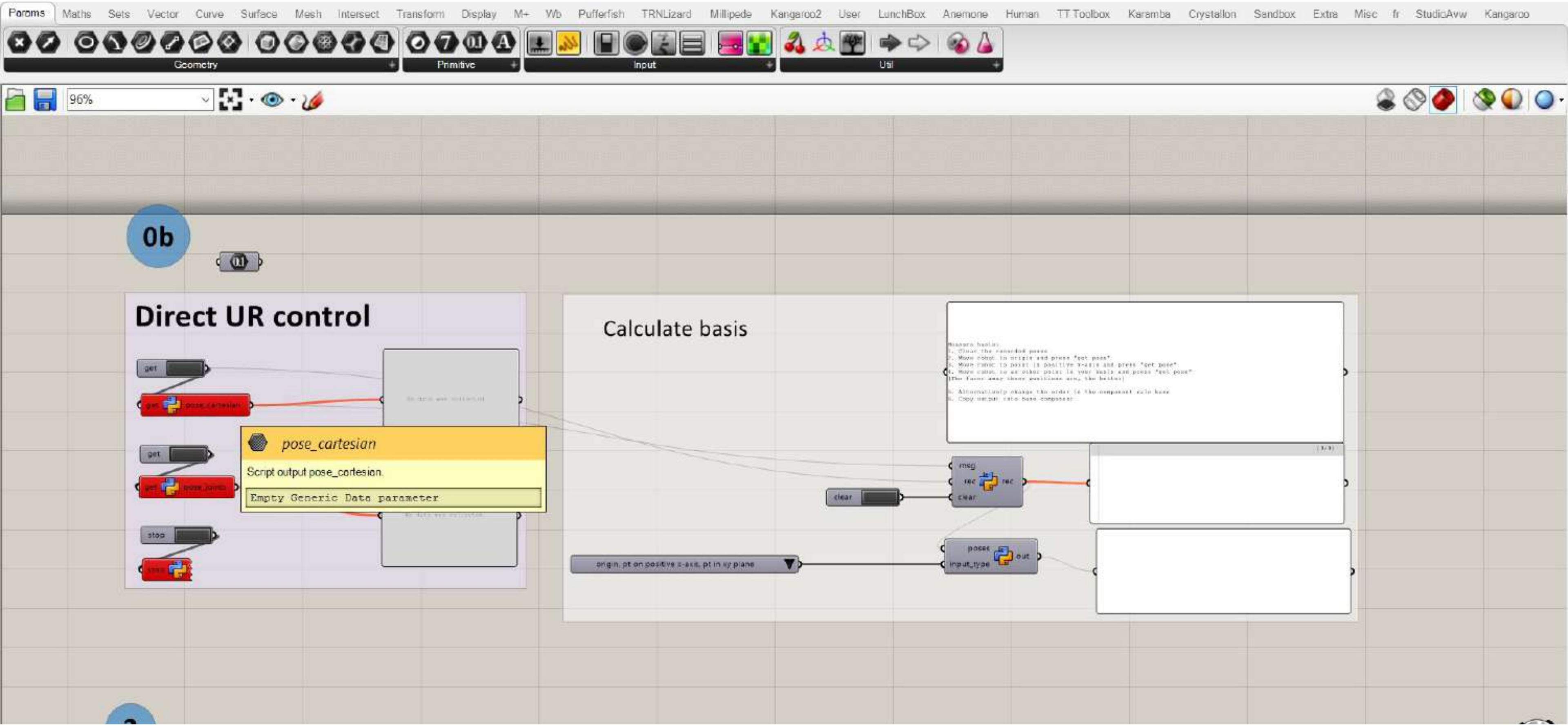
Check if UR is available

Unload libraries

clay_extruder

UR5

```
1. load robot model
2. set tool
3. set basis
```

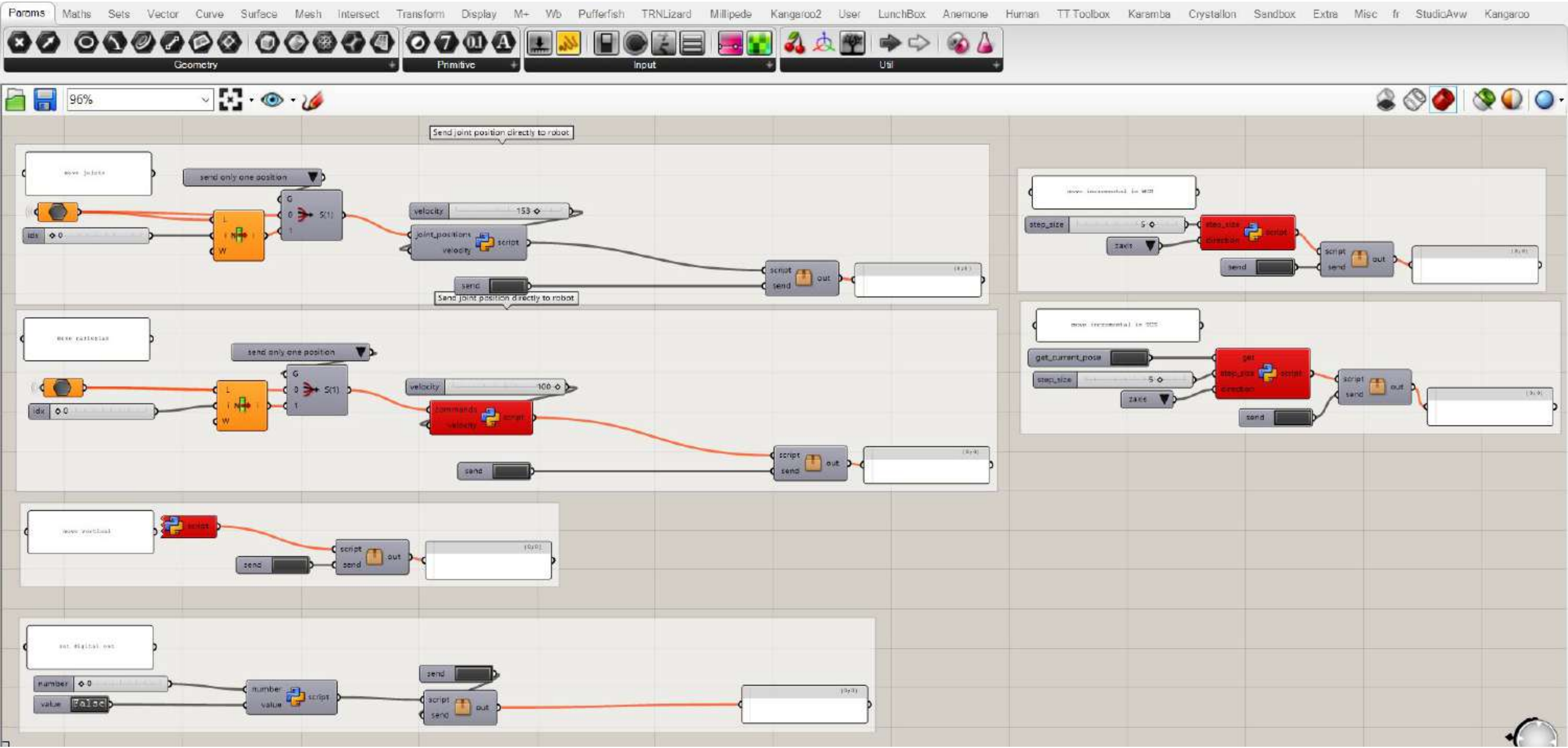


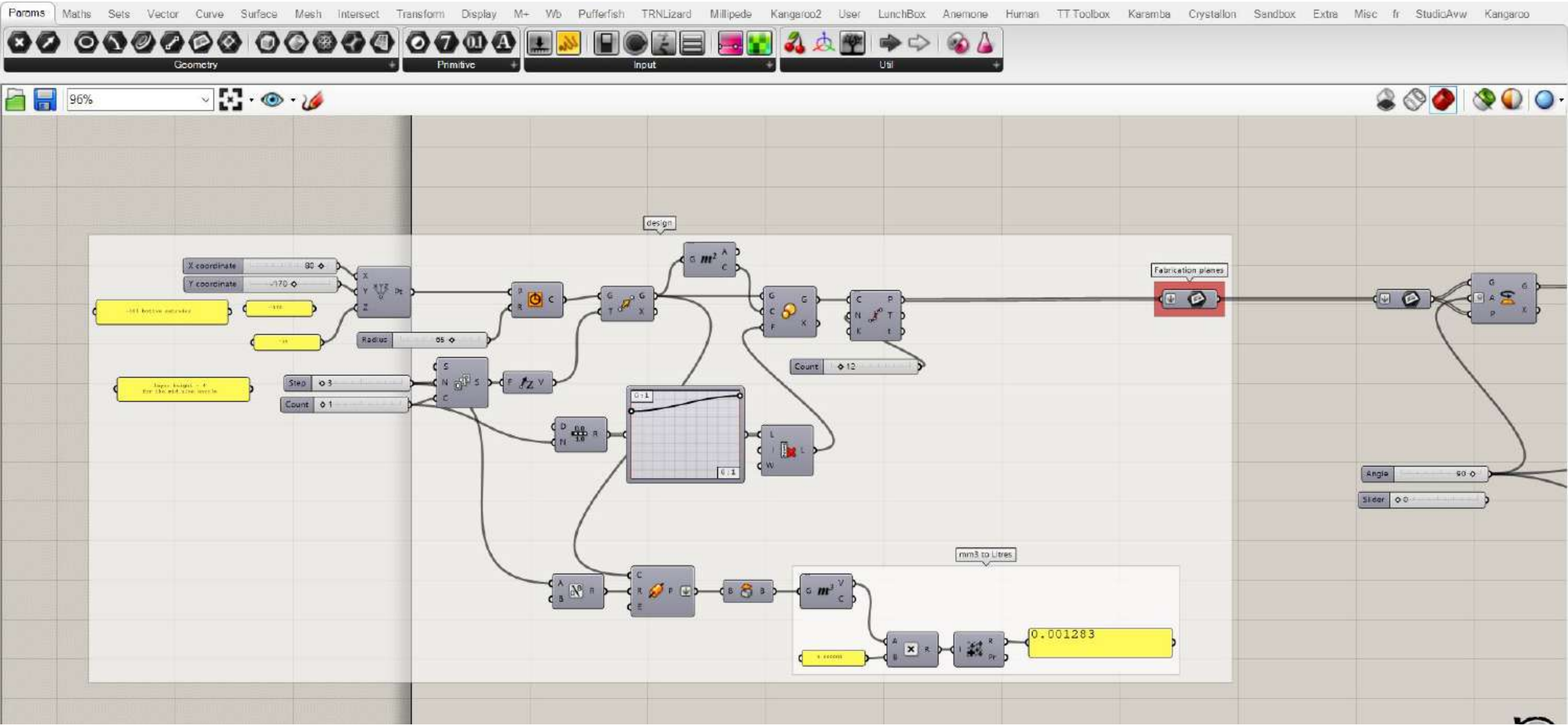
pose_cartesian

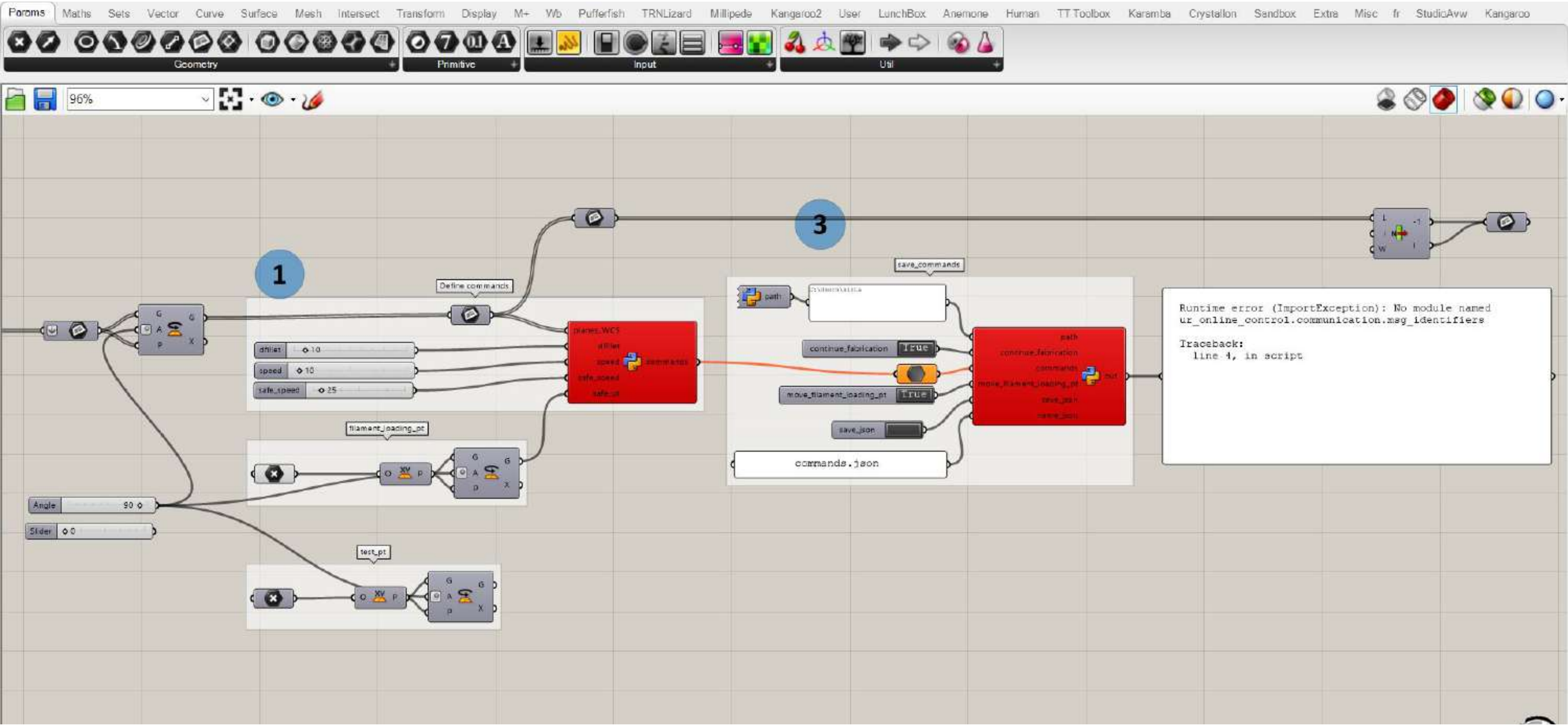
Script output pose_cartesian.

```
Empty Generic Data parameter
```

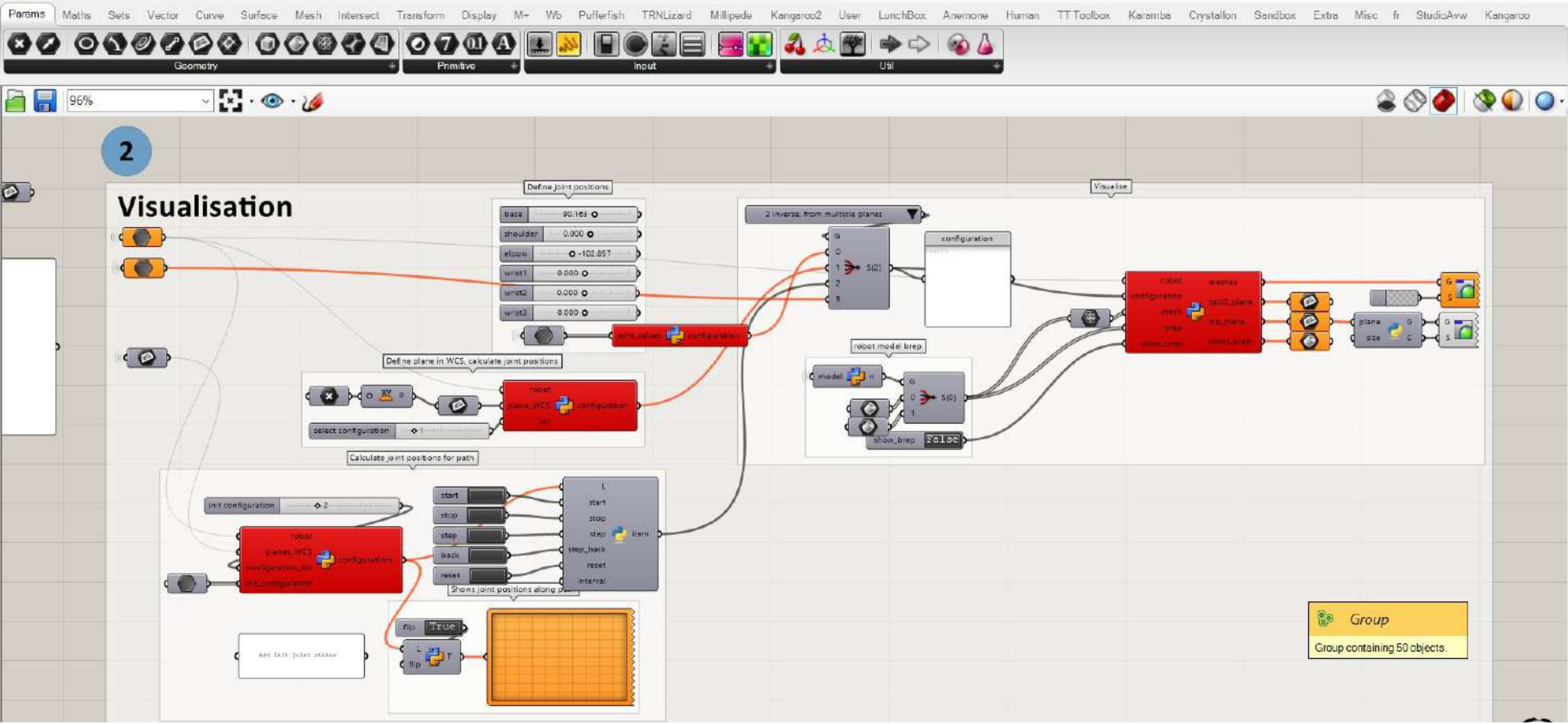
```
Measure basis:
1. Clear the recorded poses
2. Move robot to origin and press "get pose"
3. Move robot to point in positive x-axis and press "get pose"
4. Move robot to an other point in your basis and press "get pose"
(The furer away these positions are, the better)

5. Alternatively change the order in the component xxlo base
6. Copy output into base component
```

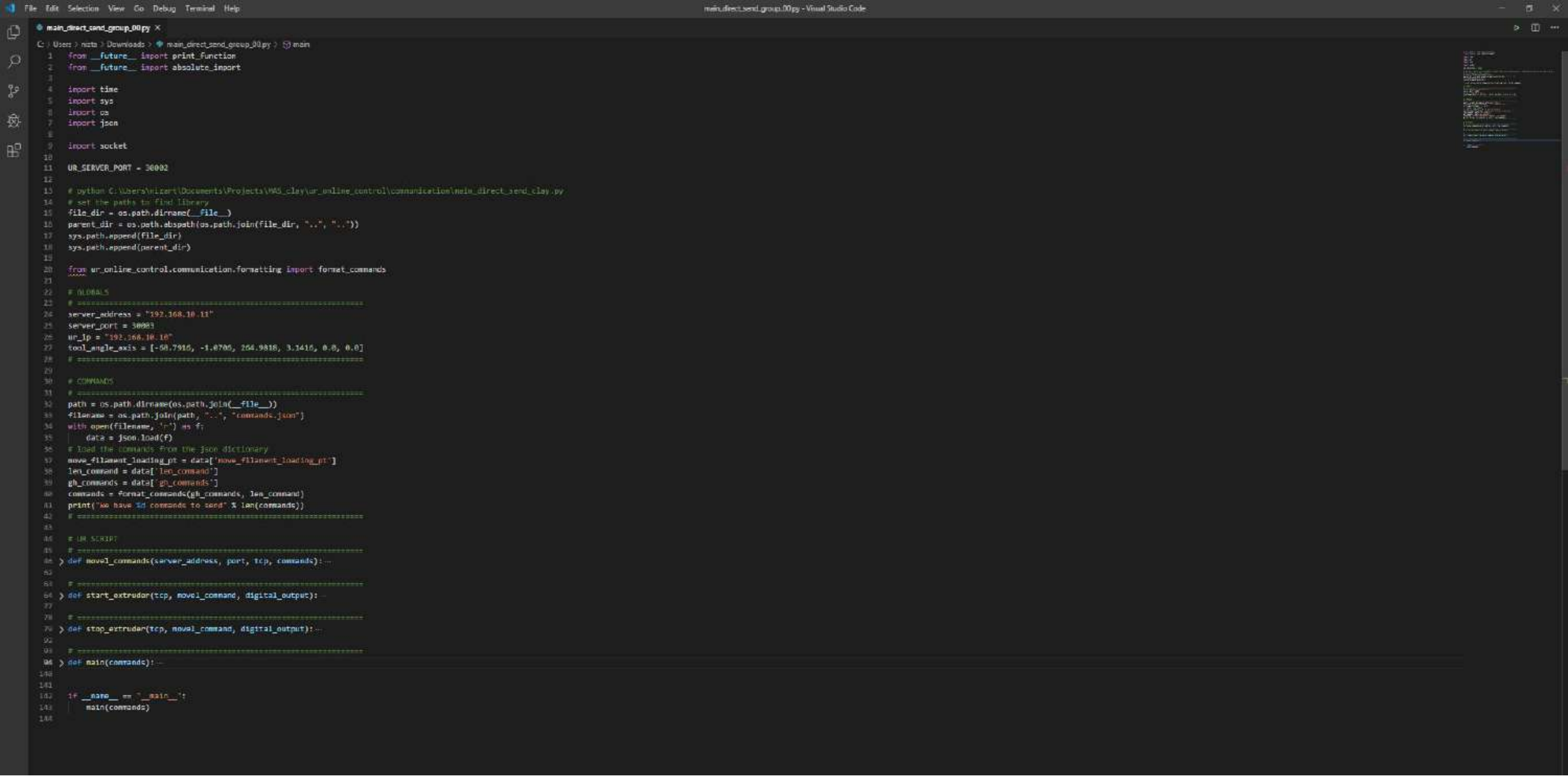

Params
Maths
Sets
Vector
Curve
Surface
Mesh
Intersect
Transform
Display
M+
Wb
Pufferfish
TRNLizard
Millipede
Kangaroo2
User
LunchBox
Anemone
Human
TT Toolbox
Karamba
Crystallon
Sandbox
Extra
Misc
fr
StudioAvw
Kangaroo
Geometry
Primitive
Input
Util
96%
Send joint position directly to robot
move joints
send only one position
idx
velocity
153
joint_positions
velocity
script
send
script
send
out
Send joint position directly to robot
move cartesian
send only one position
idx
velocity
100
commands
velocity
script
send
script
send
out
move vertical
script
script
send
out
send
set digital out
number
0
value
False
number
value
script
send
script
send
out
move incremental in WCS
step_size
5
axis
step_size
direction
script
script
send
out
send
move incremental in TCS
get_current_pose
step_size
5
axis
get
step_size
direction
script
script
send
out
send


Params
Maths
Sets
Vector
Curve
Surface
Mesh
Intersect
Transform
Display
M+
Wb
Pufferfish
TRNLizard
Millipede
Kangaroo2
User
LunchBox
Anemone
Human
TT Toolbox
Karamba
Crystallon
Sandbox
Extra
Misc
fr
StudioAvw
Kangaroo
Geometry
Primitive
Input
Util
96%
design
X coordinate
80
Y coordinate
-170
Radius
65
Step
3
Count
1
Count
12
Fabrication planes
mm3 to Litres
0.001283
Angle
90
Slider
0

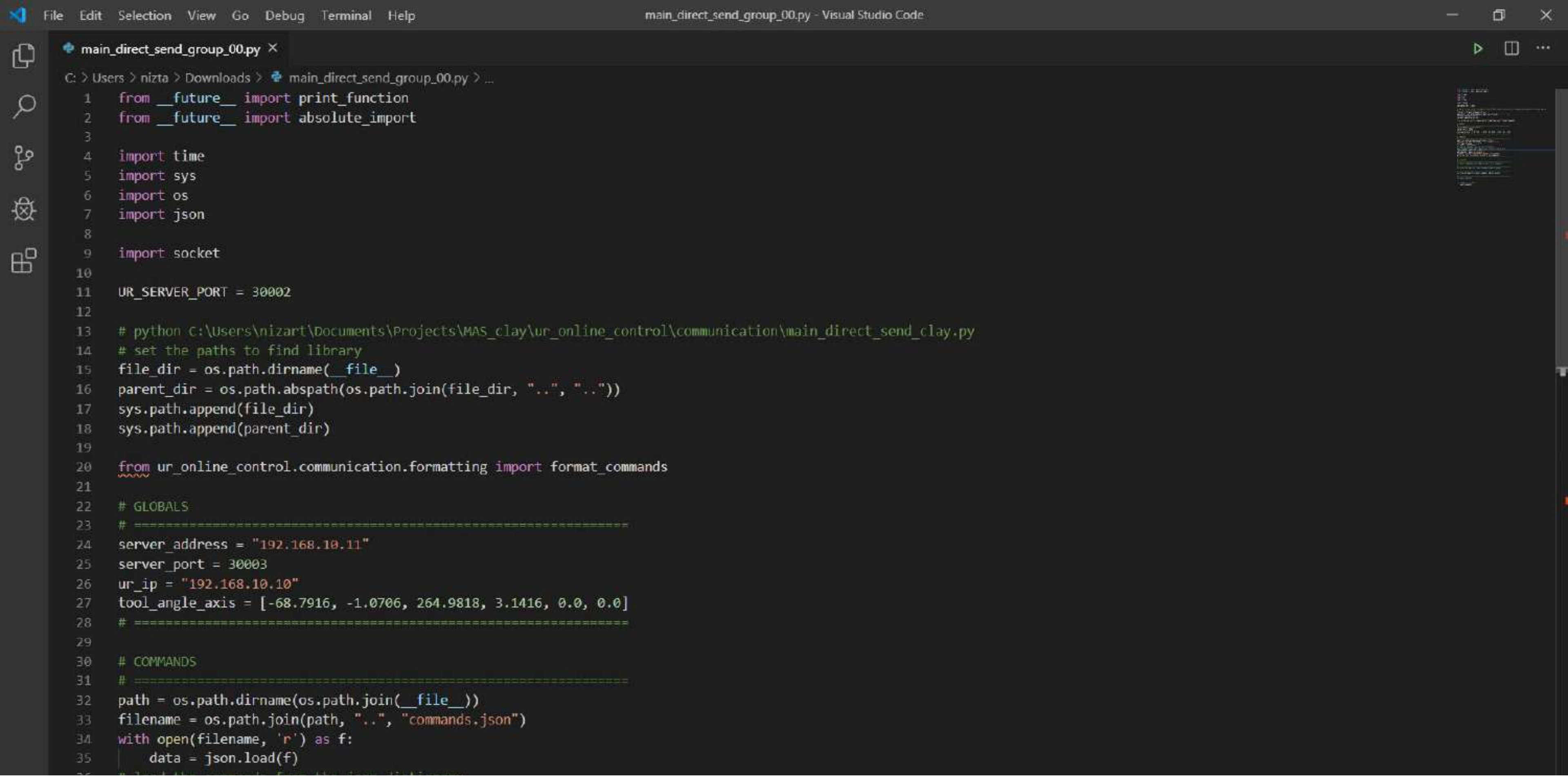


```
Runtime error (ImportException): No module named
ur_online_control.communication.msg_identifiers

Traceback:
  line 4, in script
```

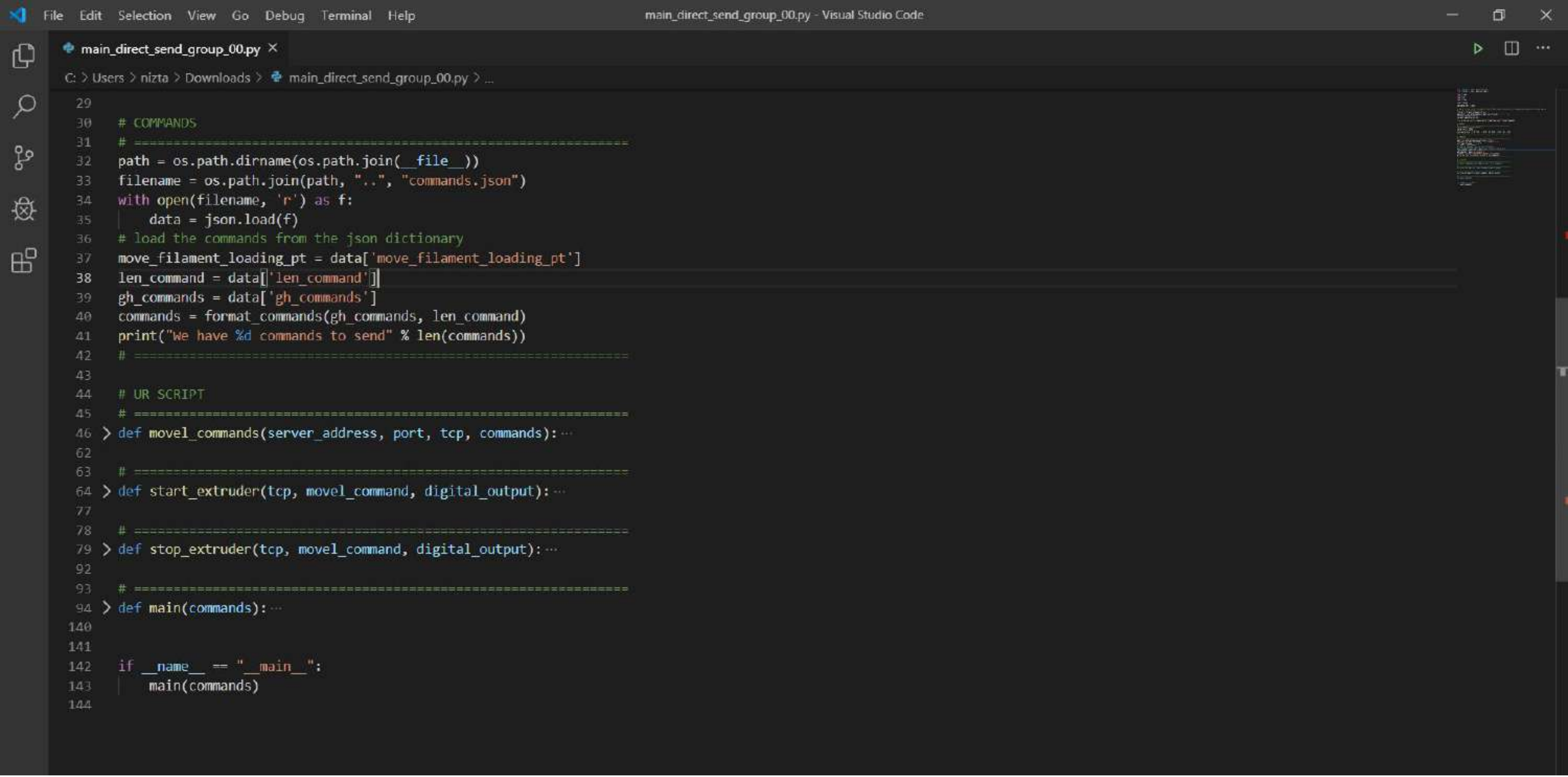

Params
Maths
Sets
Vector
Curve
Surface
Mesh
Intersect
Transform
Display
M+
Wb
Pufferfish
TRNLizard
Millipede
Kangaroo2
User
LunchBox
Anemone
Human
TT Toolbox
Karamba
Crystallon
Sandbox
Extra
Misc
fr
StudioAvw
Kangaroo
Geometry
Primitive
Input
Util
96%
2
Visualisation
Define joint positions
base 90.163
shoulder 0.000
elbow -102.857
wrist1 0.000
wrist2 0.000
wrist3 0.000
joint_values configuration
Define plane in WCS, calculate joint positions
robot
plane_WCS
configuration
select configuration
Calculate joint positions for path
init configuration 2
robot
planes_WCS
configuration_ids
init_configuration
configurations
start
stop
step
back
reset
L
start
stop
step
step_back
reset
interval
item
Shows joint positions along path
flip True
L
flip
T
Add init joint states
Visualise
2 inverse, from multiple planes
configuration
robot model brep
model
show_brep False
robot
configuration
mesh
step
show_brep
meshes
tool0_plane
tcp_plane
robot_brep
plane
size
G
S
Group
Group containing 50 objects.


File Edit Selection View Go Debug Terminal Help
main_direct_send_group_00.py - Visual Studio Code
main_direct_send_group_00.py
C: › Users › nizta › Downloads › main_direct_send_group_00.py › main

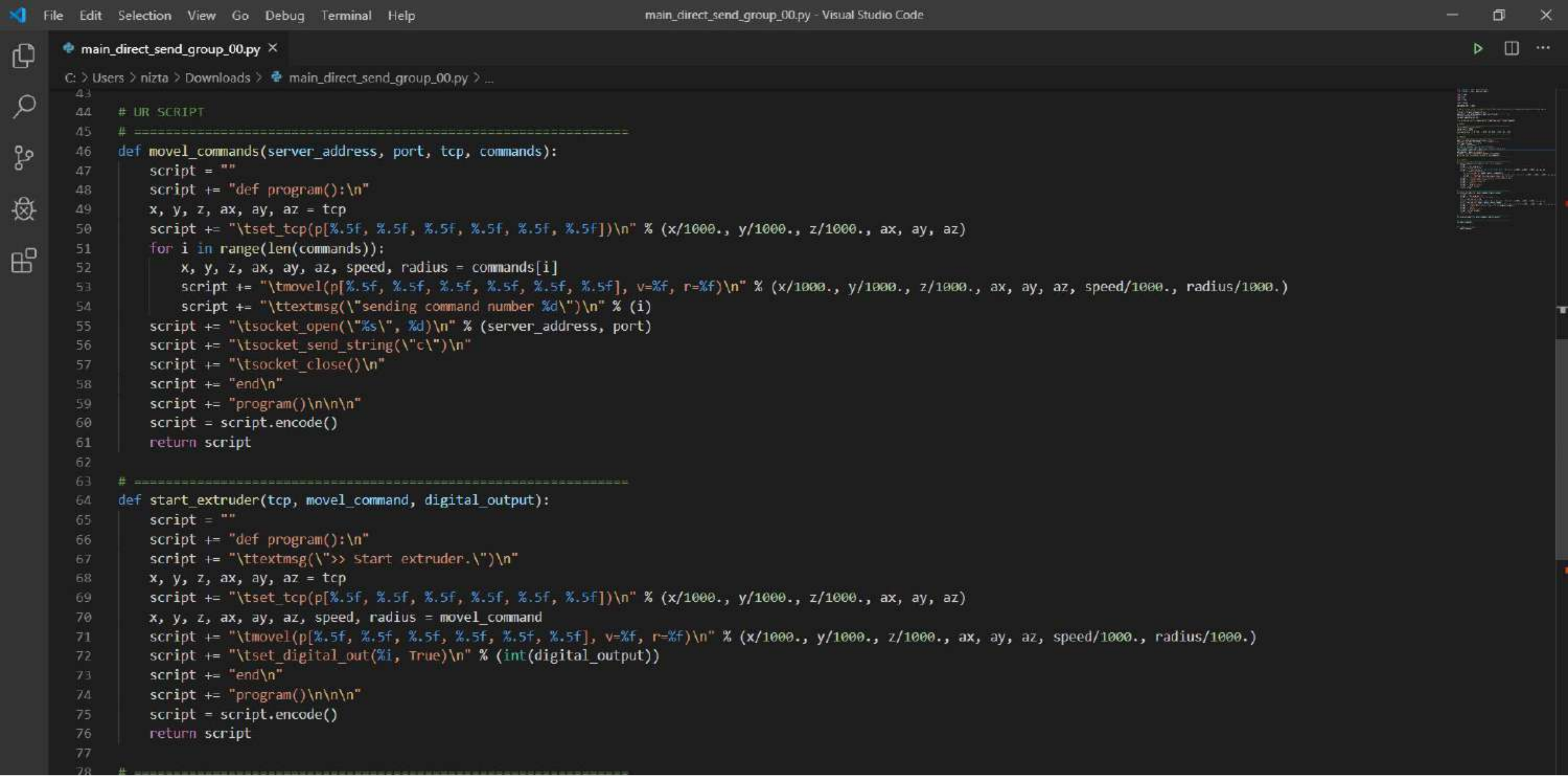
```python
from __future__ import print_function
from __future__ import absolute_import

import time
import sys
import os
import json

import socket

UR_SERVER_PORT = 30002

# python C:\Users\nizart\Documents\Projects\MAS_clay\ur_online_control\communication\main_direct_send_clay.py
# set the paths to find library
file_dir = os.path.dirname(__file__)
parent_dir = os.path.abspath(os.path.join(file_dir, "..", ".."))
sys.path.append(file_dir)
sys.path.append(parent_dir)

from ur_online_control.communication.formatting import format_commands

# GLOBALS
# ===============================================================
server_address = "192.168.10.11"
server_port = 30003
ur_ip = "192.168.10.10"
tool_angle_axis = [-68.7916, -1.0706, 264.9818, 3.1416, 0.0, 0.0]
# ===============================================================

# COMMANDS
# ===============================================================
path = os.path.dirname(os.path.join(__file__))
filename = os.path.join(path, "..", "commands.json")
with open(filename, 'r') as f:
    data = json.load(f)
# load the commands from the json dictionary
move_filament_loading_pt = data['move_filament_loading_pt']
len_command = data['len_command']
gh_commands = data['gh_commands']
commands = format_commands(gh_commands, len_command)
print("We have %d commands to send" % len(commands))
# ===============================================================

# UR SCRIPT
# ===============================================================
def movel_commands(server_address, port, tcp, commands):...

# ===============================================================
def start_extruder(tcp, movel_command, digital_output):...

# ===============================================================
def stop_extruder(tcp, movel_command, digital_output):...

# ===============================================================
def main(commands):...


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


File
Edit
Selection
View
Go
Debug
Terminal
Help
main_direct_send_group_00.py - Visual Studio Code
main_direct_send_group_00.py
C: > Users > nizta > Downloads > main_direct_send_group_00.py > ...

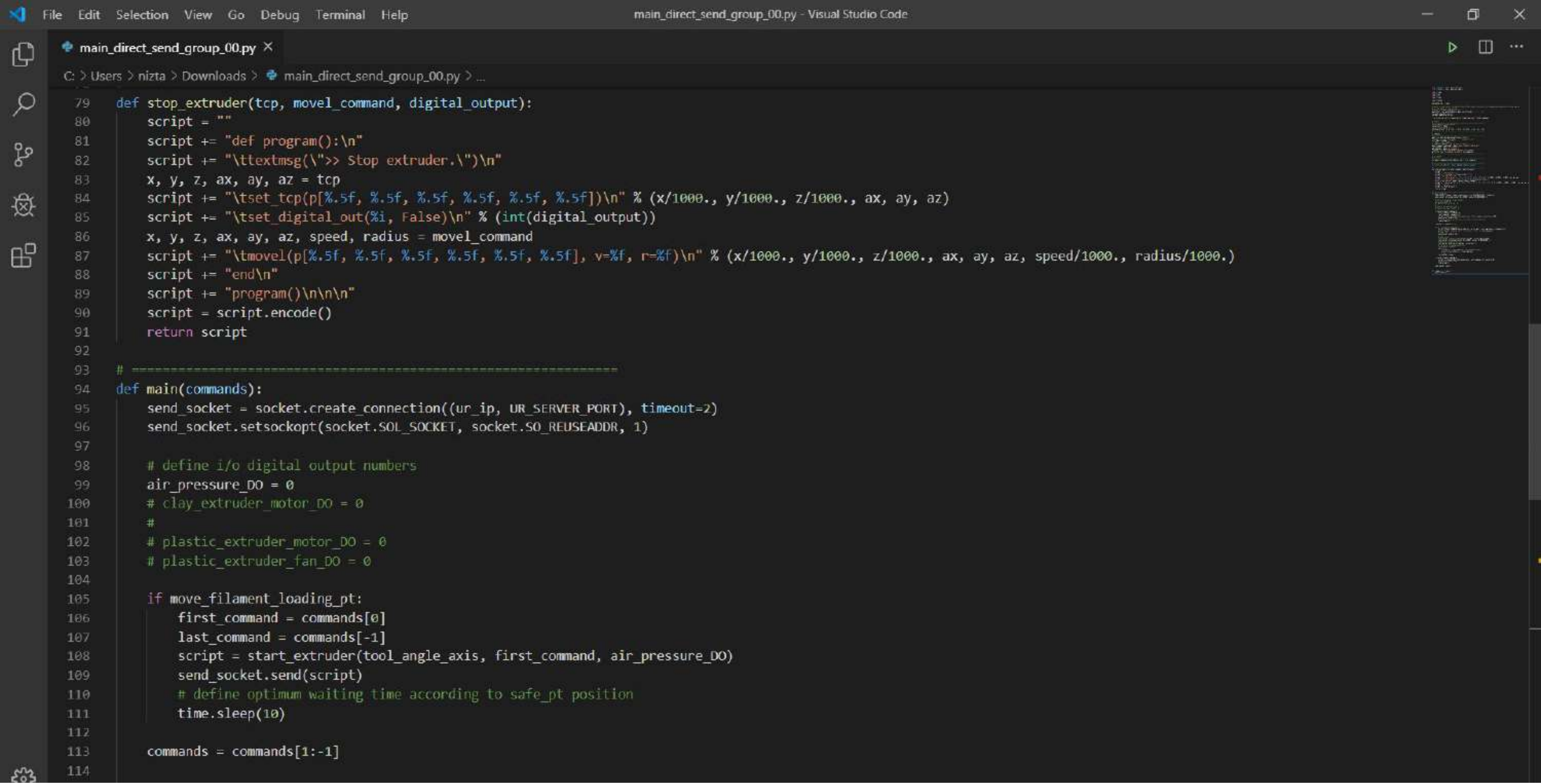
```python
from __future__ import print_function
from __future__ import absolute_import

import time
import sys
import os
import json

import socket

UR_SERVER_PORT = 30002

# python C:\Users\nizart\Documents\Projects\MAS_clay\ur_online_control\communication\main_direct_send_clay.py
# set the paths to find library
file_dir = os.path.dirname(__file__)
parent_dir = os.path.abspath(os.path.join(file_dir, "..", ".."))
sys.path.append(file_dir)
sys.path.append(parent_dir)

from ur_online_control.communication.formatting import format_commands

# GLOBALS
# ===================================================================
server_address = "192.168.10.11"
server_port = 30003
ur_ip = "192.168.10.10"
tool_angle_axis = [-68.7916, -1.0706, 264.9818, 3.1416, 0.0, 0.0]
# ===================================================================

# COMMANDS
# ===================================================================
path = os.path.dirname(os.path.join(__file__))
filename = os.path.join(path, "..", "commands.json")
with open(filename, 'r') as f:
    data = json.load(f)
```


File Edit Selection View Go Debug Terminal Help
main_direct_send_group_00.py - Visual Studio Code
main_direct_send_group_00.py
C: > Users > nizta > Downloads > main_direct_send_group_00.py > ...

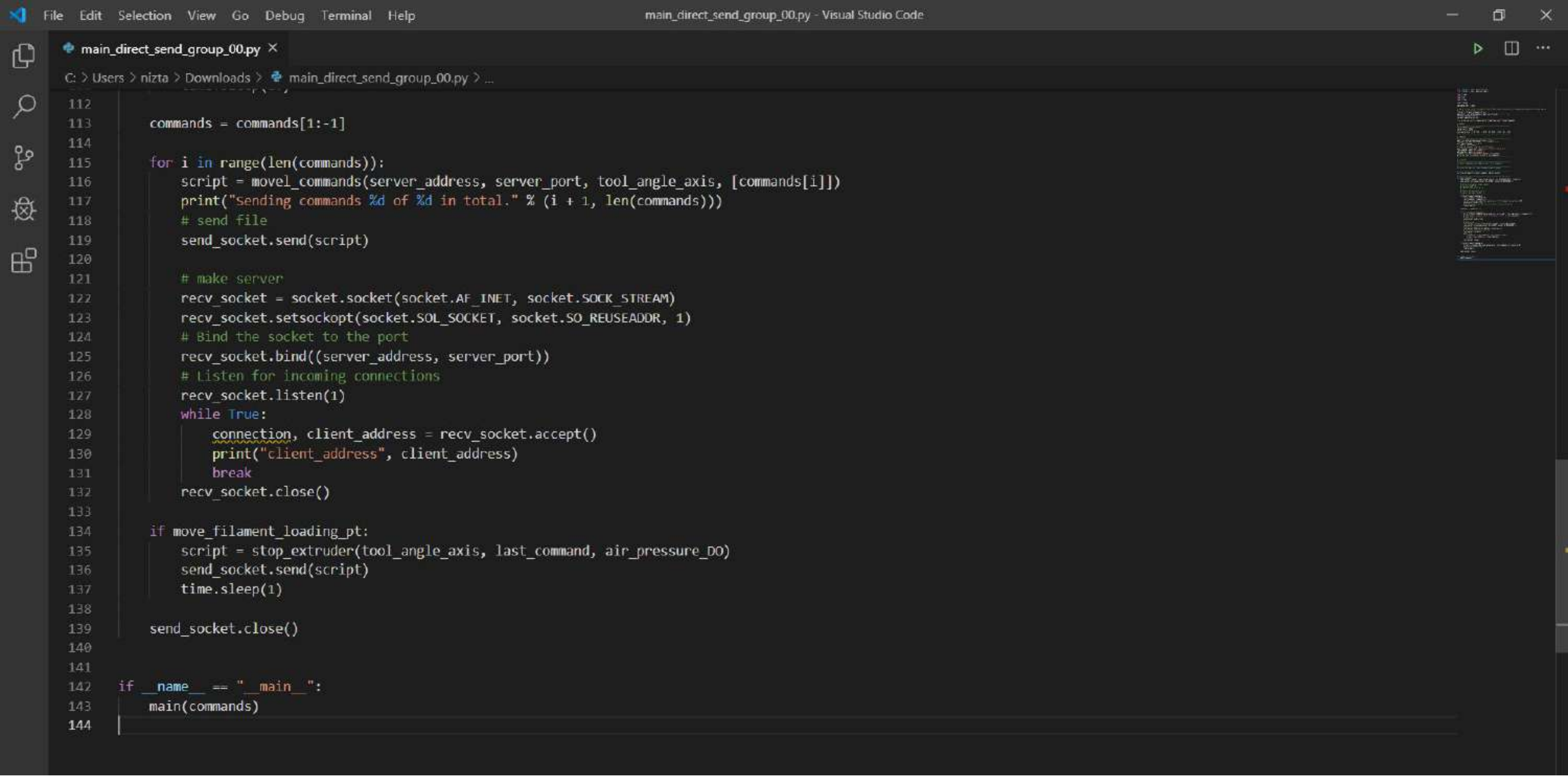
```python

# COMMANDS
# ================================================================
path = os.path.dirname(os.path.join(__file__))
filename = os.path.join(path, "..", "commands.json")
with open(filename, 'r') as f:
    data = json.load(f)
# load the commands from the json dictionary
move_filament_loading_pt = data['move_filament_loading_pt']
len_command = data['len_command']
gh_commands = data['gh_commands']
commands = format_commands(gh_commands, len_command)
print("We have %d commands to send" % len(commands))
# ================================================================

# UR SCRIPT
# ================================================================
def movel_commands(server_address, port, tcp, commands):···

# ================================================================
def start_extruder(tcp, movel_command, digital_output):···

# ================================================================
def stop_extruder(tcp, movel_command, digital_output):···

# ================================================================
def main(commands):···


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

```


File
Edit
Selection
View
Go
Debug
Terminal
Help
main_direct_send_group_00.py - Visual Studio Code
main_direct_send_group_00.py
C: > Users > nizta > Downloads > main_direct_send_group_00.py > ...

```python
# UR SCRIPT
# ============================================================
def movel_commands(server_address, port, tcp, commands):
    script = ""
    script += "def program():\n"
    x, y, z, ax, ay, az = tcp
    script += "\tset_tcp(p[%.5f, %.5f, %.5f, %.5f, %.5f, %.5f])\n" % (x/1000., y/1000., z/1000., ax, ay, az)
    for i in range(len(commands)):
        x, y, z, ax, ay, az, speed, radius = commands[i]
        script += "\tmovel(p[%.5f, %.5f, %.5f, %.5f, %.5f, %.5f], v=%f, r=%f)\n" % (x/1000., y/1000., z/1000., ax, ay, az, speed/1000., radius/1000.)
        script += "\ttextmsg(\"sending command number %d\")\n" % (i)
    script += "\tsocket_open(\"%s\", %d)\n" % (server_address, port)
    script += "\tsocket_send_string(\"c\")\n"
    script += "\tsocket_close()\n"
    script += "end\n"
    script += "program()\n\n\n"
    script = script.encode()
    return script

# ============================================================
def start_extruder(tcp, movel_command, digital_output):
    script = ""
    script += "def program():\n"
    script += "\ttextmsg(\">> Start extruder.\")\n"
    x, y, z, ax, ay, az = tcp
    script += "\tset_tcp(p[%.5f, %.5f, %.5f, %.5f, %.5f, %.5f])\n" % (x/1000., y/1000., z/1000., ax, ay, az)
    x, y, z, ax, ay, az, speed, radius = movel_command
    script += "\tmovel(p[%.5f, %.5f, %.5f, %.5f, %.5f, %.5f], v=%f, r=%f)\n" % (x/1000., y/1000., z/1000., ax, ay, az, speed/1000., radius/1000.)
    script += "\tset_digital_out(%i, True)\n" % (int(digital_output))
    script += "end\n"
    script += "program()\n\n\n"
    script = script.encode()
    return script

# ============================================================
```


File
Edit
Selection
View
Go
Debug
Terminal
Help
main_direct_send_group_00.py - Visual Studio Code
main_direct_send_group_00.py
C: > Users > nizta > Downloads > main_direct_send_group_00.py > ...

```python
def stop_extruder(tcp, movel_command, digital_output):
    script = ""
    script += "def program():\n"
    script += "\ttextmsg(\">> Stop extruder.\")\n"
    x, y, z, ax, ay, az = tcp
    script += "\tset_tcp(p[%.5f, %.5f, %.5f, %.5f, %.5f, %.5f])\n" % (x/1000., y/1000., z/1000., ax, ay, az)
    script += "\tset_digital_out(%i, False)\n" % (int(digital_output))
    x, y, z, ax, ay, az, speed, radius = movel_command
    script += "\tmovel(p[%.5f, %.5f, %.5f, %.5f, %.5f, %.5f], v=%f, r=%f)\n" % (x/1000., y/1000., z/1000., ax, ay, az, speed/1000., radius/1000.)
    script += "end\n"
    script += "program()\n\n\n"
    script = script.encode()
    return script

# ===============================================================
def main(commands):
    send_socket = socket.create_connection((ur_ip, UR_SERVER_PORT), timeout=2)
    send_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

    # define i/o digital output numbers
    air_pressure_DO = 0
    # clay_extruder_motor_DO = 0
    #
    # plastic_extruder_motor_DO = 0
    # plastic_extruder_fan_DO = 0

    if move_filament_loading_pt:
        first_command = commands[0]
        last_command = commands[-1]
        script = start_extruder(tool_angle_axis, first_command, air_pressure_DO)
        send_socket.send(script)
        # define optimum waiting time according to safe_pt position
        time.sleep(10)

    commands = commands[1:-1]

```


File
Edit
Selection
View
Go
Debug
Terminal
Help
main_direct_send_group_00.py - Visual Studio Code
main_direct_send_group_00.py
C: > Users > nizta > Downloads > main_direct_send_group_00.py > ...

```python

    commands = commands[1:-1]

    for i in range(len(commands)):
        script = movel_commands(server_address, server_port, tool_angle_axis, [commands[i]])
        print("Sending commands %d of %d in total." % (i + 1, len(commands)))
        # send file
        send_socket.send(script)

        # make server
        recv_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        recv_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        # Bind the socket to the port
        recv_socket.bind((server_address, server_port))
        # Listen for incoming connections
        recv_socket.listen(1)
        while True:
            connection, client_address = recv_socket.accept()
            print("client_address", client_address)
            break
        recv_socket.close()

    if move_filament_loading_pt:
        script = stop_extruder(tool_angle_axis, last_command, air_pressure_DO)
        send_socket.send(script)
        time.sleep(1)

    send_socket.close()


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

```