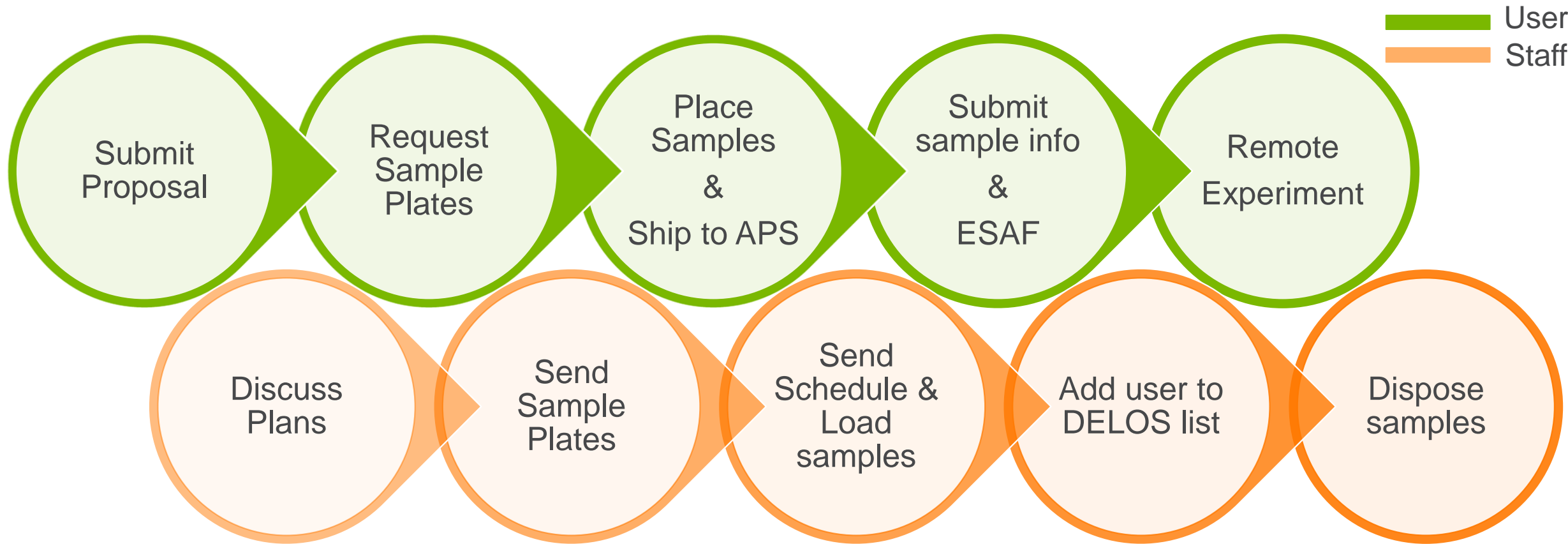


# Remote Operation of 12-ID SAXS/GISAXS with UR3 robot

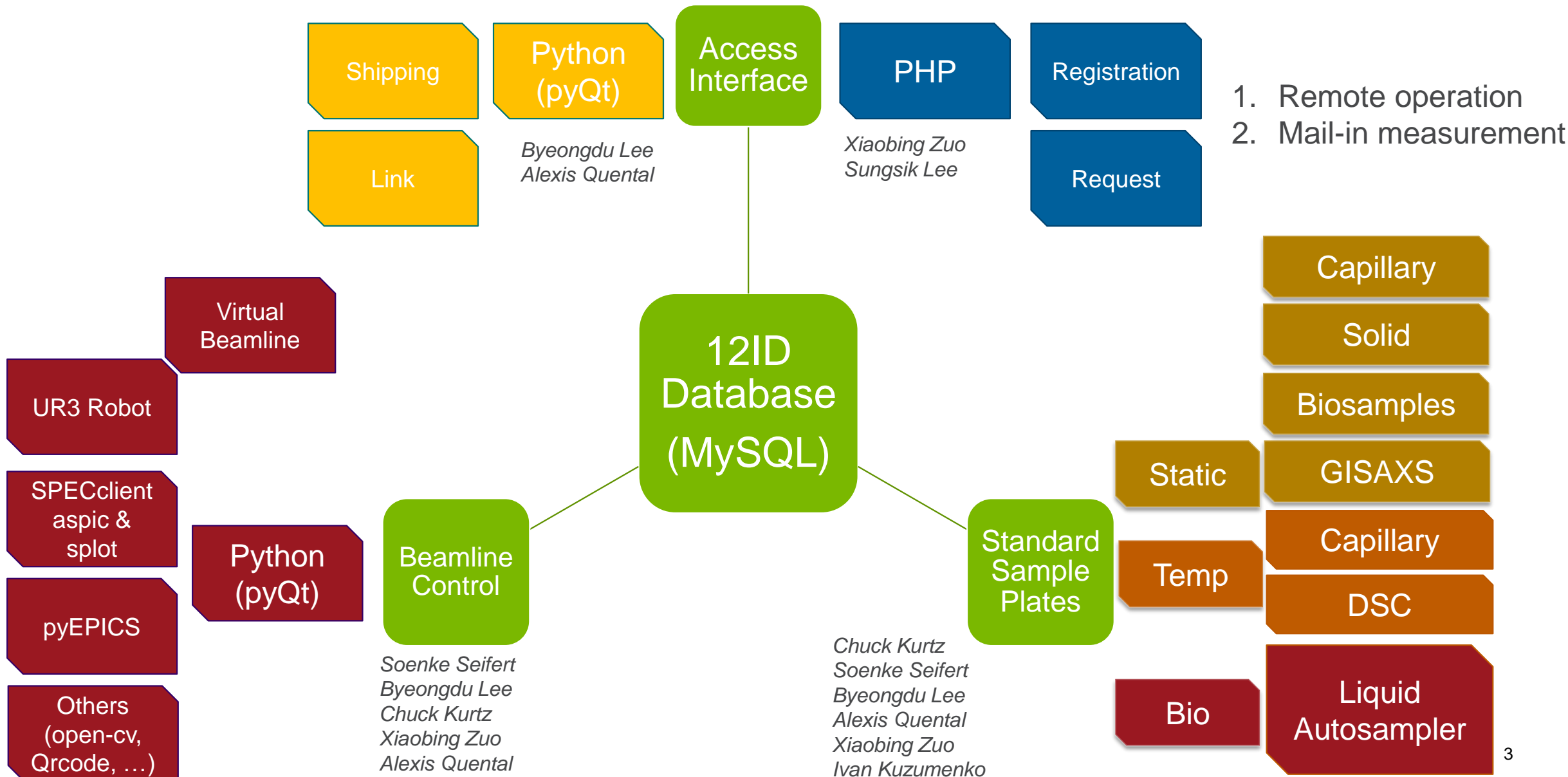
BYEONGDU LEE

CMS GROUP

# User flow chart



# 12-ID Remote Experiment System



# Database

## Mysql (Xiaobing, Byeongdu)

- Tables
  - frames, frame\_items, order, order\_items, users, order\_statuses
  - These are constructed from an example for a store that needs to manage product inventories, orders, and shipping.
- Can be easily communicated with python and PHP.
- Running on a PC at the beamline

# Web Interface

## PHP programming (Xiaobing)

- Portal to communicate with 12ID Remote Operation Database
- For Users
  - User registration
    - Not linked to APS DB
    - User ID : email address
    - Other info to provide: mailing address and GUP number
  - Requesting sample plates
  - Registering sample information for the sample plates
- For staff
  - DB searching for statistics and management
  - Update status of the user's sample plates

The screenshot shows a web browser window with the URL `12id.xray.aps.anl.gov/remoteOperation_request.html`. The page features a cyan header with the text "Welcome to Beamline 12-ID-B". The main content area is divided into a left sidebar and a right main section. The sidebar contains three orange buttons: "Mail-in/ remote Operation Procedures", "Available Sample Holders", and "Instruction on Sample Mounting & Beamline Operation". The main section contains a "Registration:" form with the following fields: First Name (John), Last Name (Smith), Email (jsmith@example.com), Badge Number (543210), Shipping Address 1/2/3 (three empty text boxes), City, State, and Zip Code. A "Submit Registration" button is located below the form. Below the registration form is a "Request sample holders:" section with fields for Registration email (jsmith@example.com), User Badge (Number only) (543210), GUP# (Number only) (612345), and Request to Beamline.

# Sample Plate Shipping

## PHP/Python codes for Dymo label printer

- Print shipping labels (user's registered address)
- Scan QRcodes, generate web-links, and email to users
- Can check the status of a sample frame.



12ID Remote User Tool

Search

Beamline:

User Email:

User Address:

Clemmie Betchley  
5 Spohn Circle,  
Arlington, TX 3675

What User Requested:

order_id	user_id	GUP	order_date	comments
1	5	11111	2019-01-30	None
3	5	11115	2017-08-25	

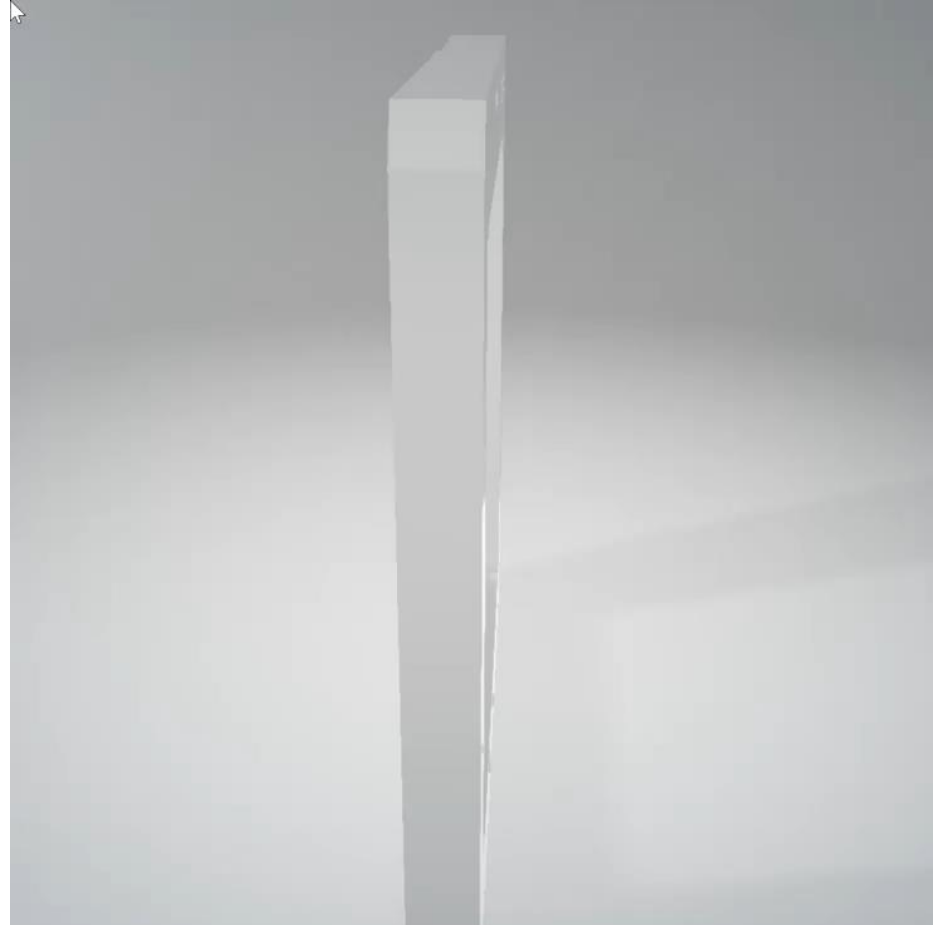
id	order_id	frame_id	frame_index	item_status	shipped_date	returned_date	
1	7	1	Capillary Plate	3	2	2020-09-15	None

QR Code Scanned:

[https://12id.xray.aps.anl.gov/CPT\\_input.php?sample\\_holder=CPT003](https://12id.xray.aps.anl.gov/CPT_input.php?sample_holder=CPT003)

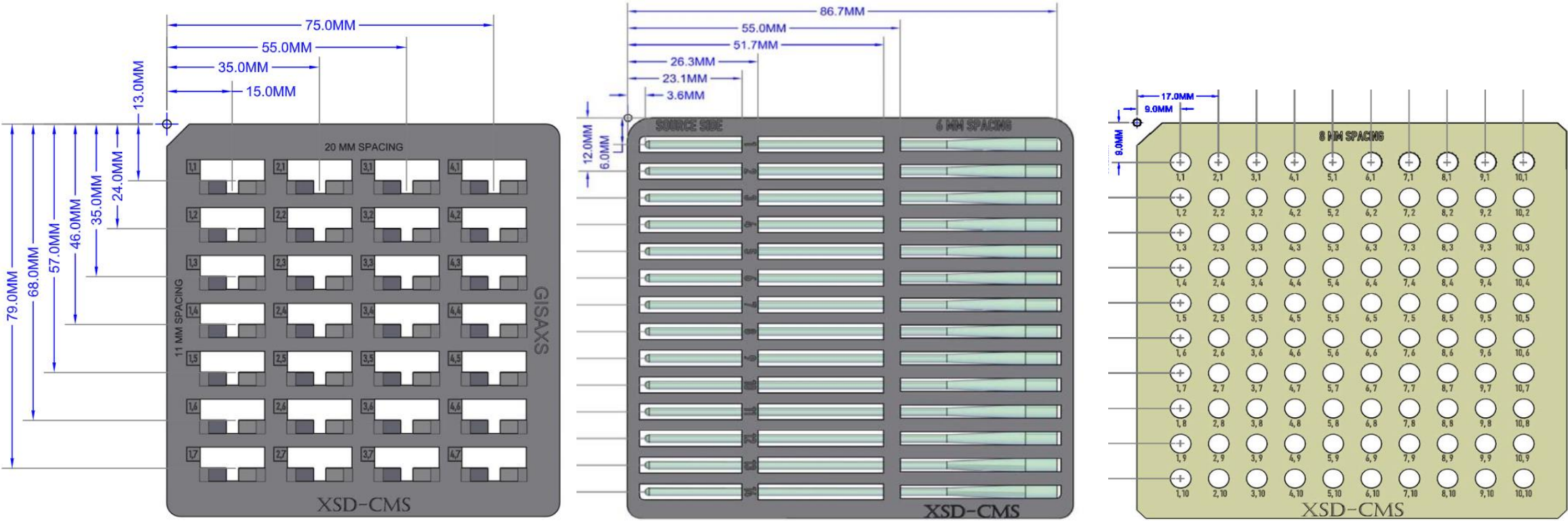
# Sample Frame

3D printed (Chuck, Alexis, Soenke, Ben Reinhart)



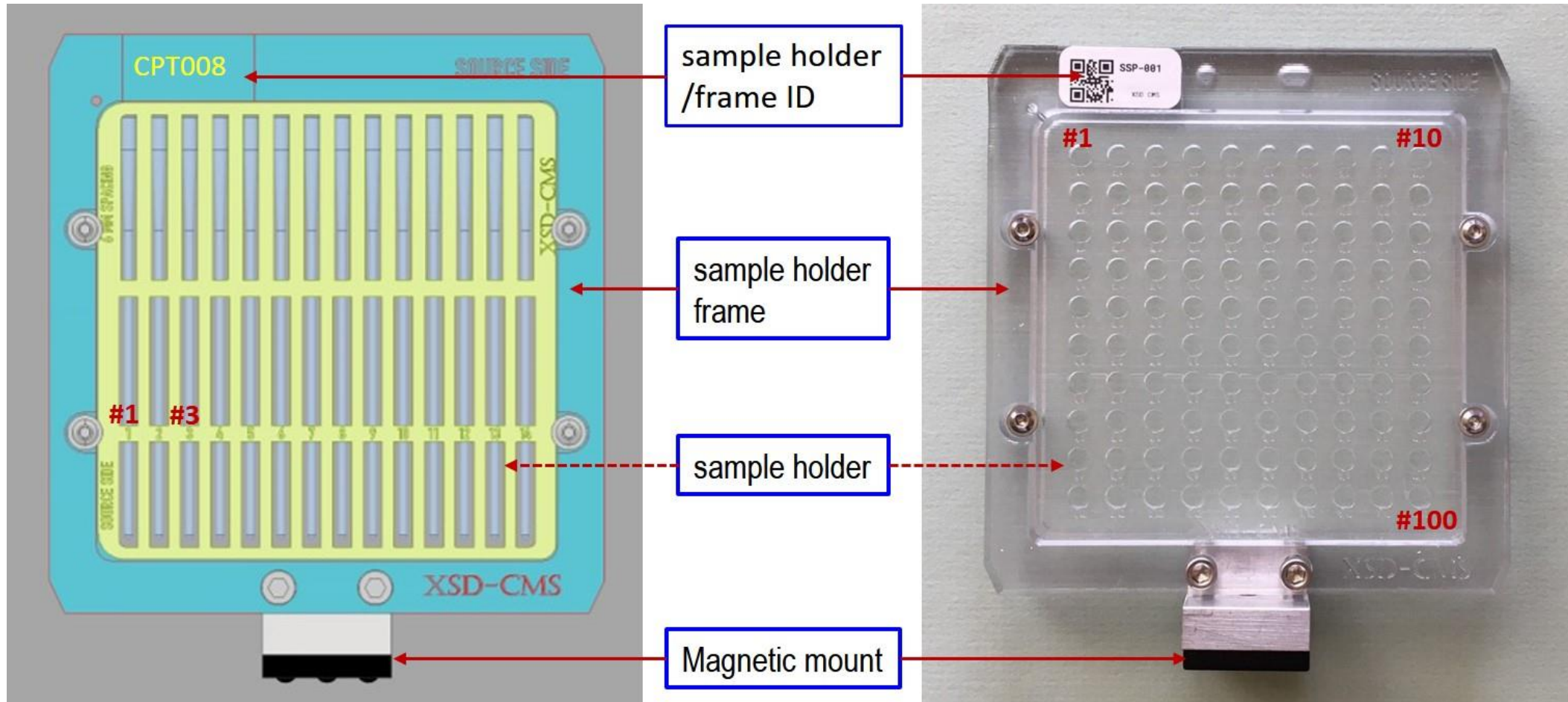
# Sample Plates

3D printed

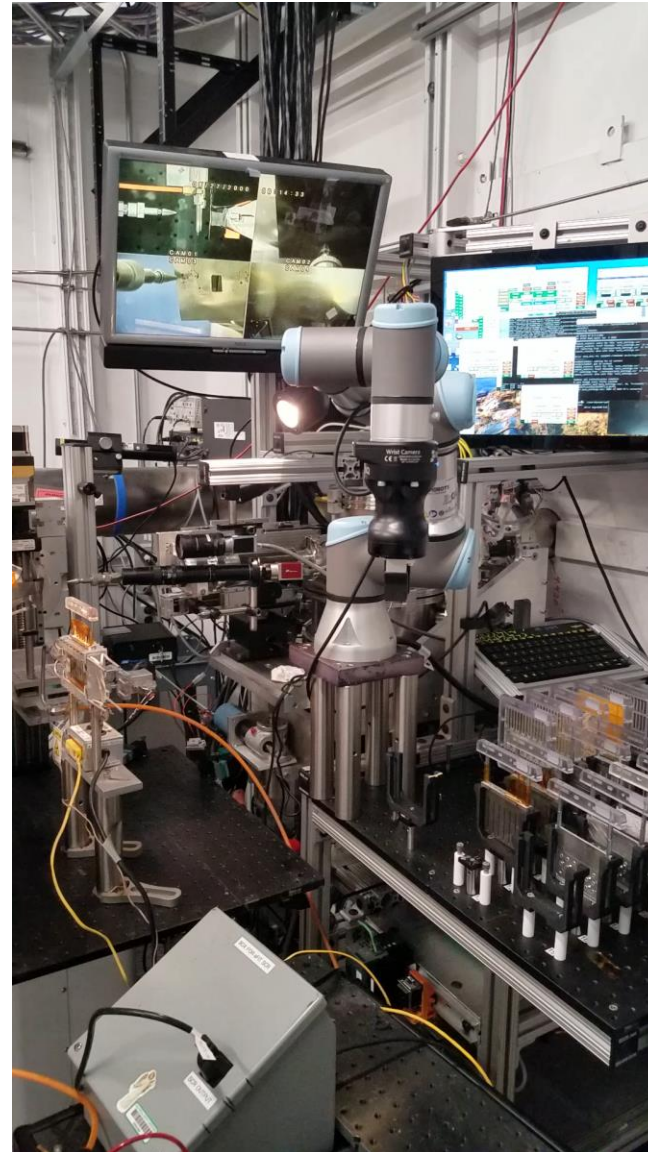




# Assembly of Sample Plate on Sample Frame



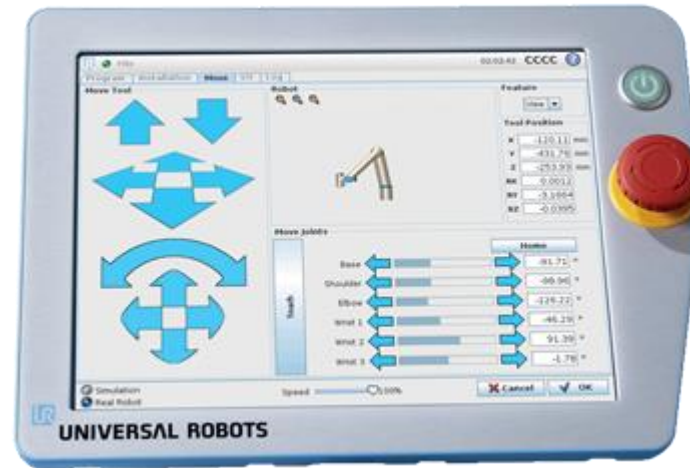
# Robot



- How to get started.
  - [https://youtu.be/nFP\\_z5l68\\_g](https://youtu.be/nFP_z5l68_g)
  - So many YouTube videos.
- Vendor's online training courses ← Short but very informative
  - <https://www.universal-robots.com/academy/>

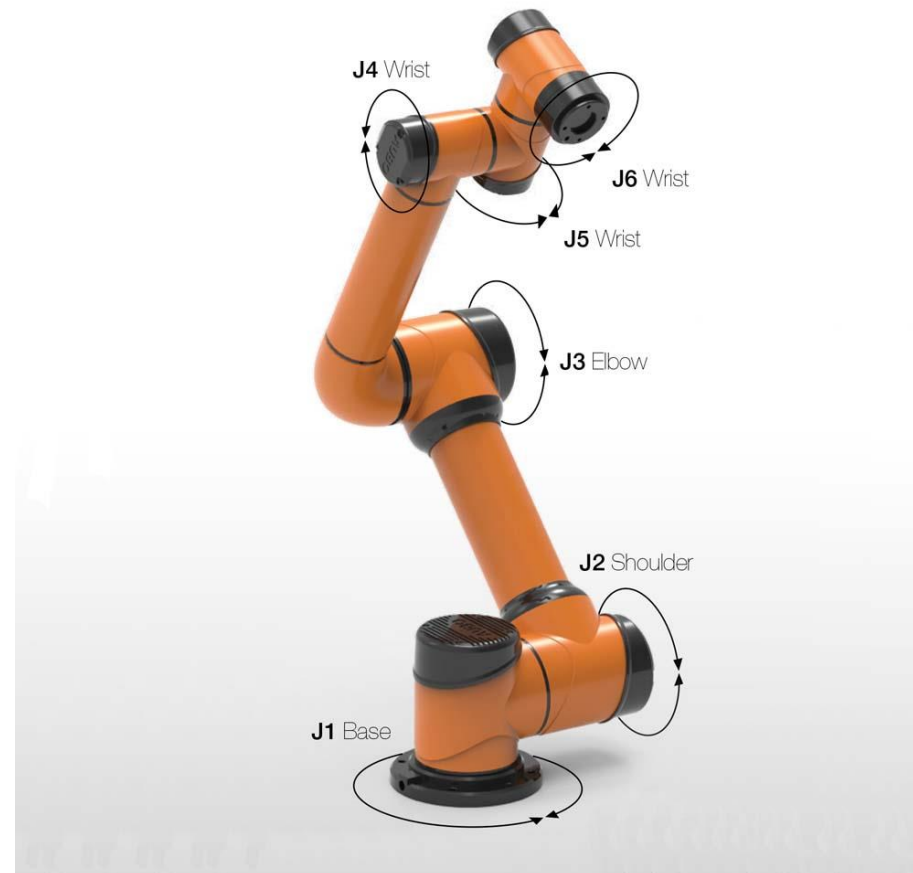


# Teach Pendant and Control box

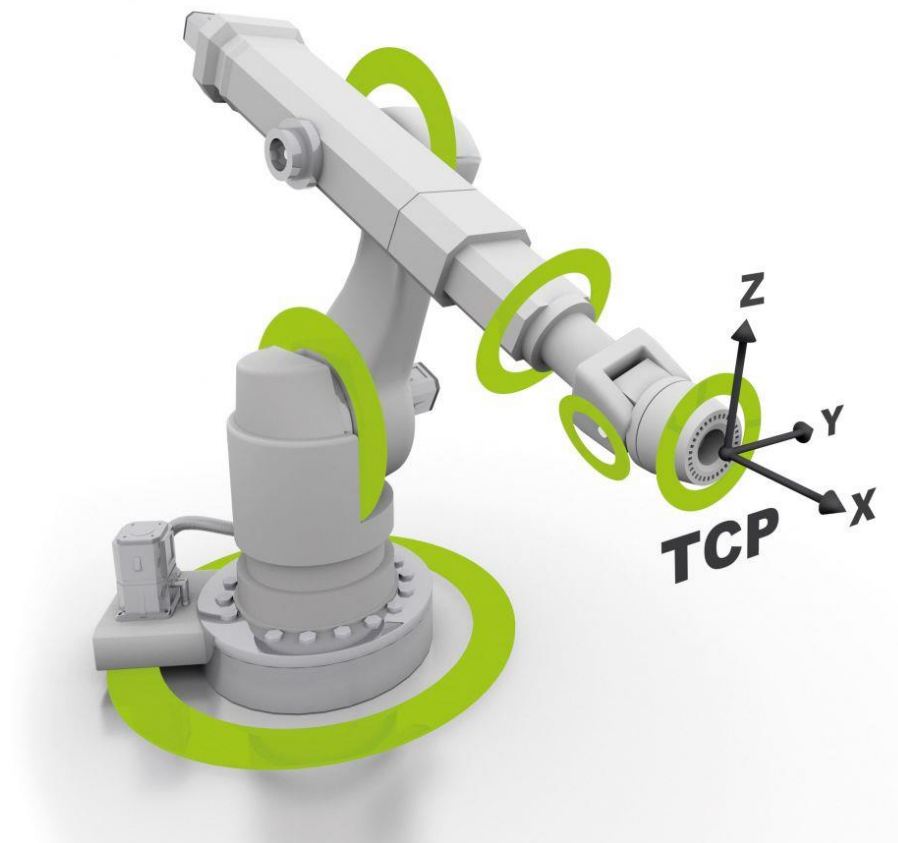


Linux, Debian  
Many control I/O  
Communication: USB,  
Ethernet  
Comes with python2.7

# Definition



# TCP (Tool Center Point)



# Forward and Inverse Kinematics

Diagram of a 4-link robotic arm with joints  $q_1, q_2, q_3, q_4$  and link lengths  $L_1, L_2, L_3, L_4$ . The end effector is labeled "End I" with orientation  $\theta$ . A coordinate system  $(X, Y)$  is shown at the base.

Software interface showing joint angles and TCP coordinates:

Joint	Angle (°)
Base	-91.71
Shoulder	-98.96
Elbow	-126.22
Wrist 1	-46.29
Wrist 2	91.39
Wrist 3	-1.78

Feature TCP coordinates:

Coordinate	Value (mm)
X	-120.11
Y	-431.76
Z	-253.93
RX	0.0012
RY	-3.1664
RZ	-0.0395

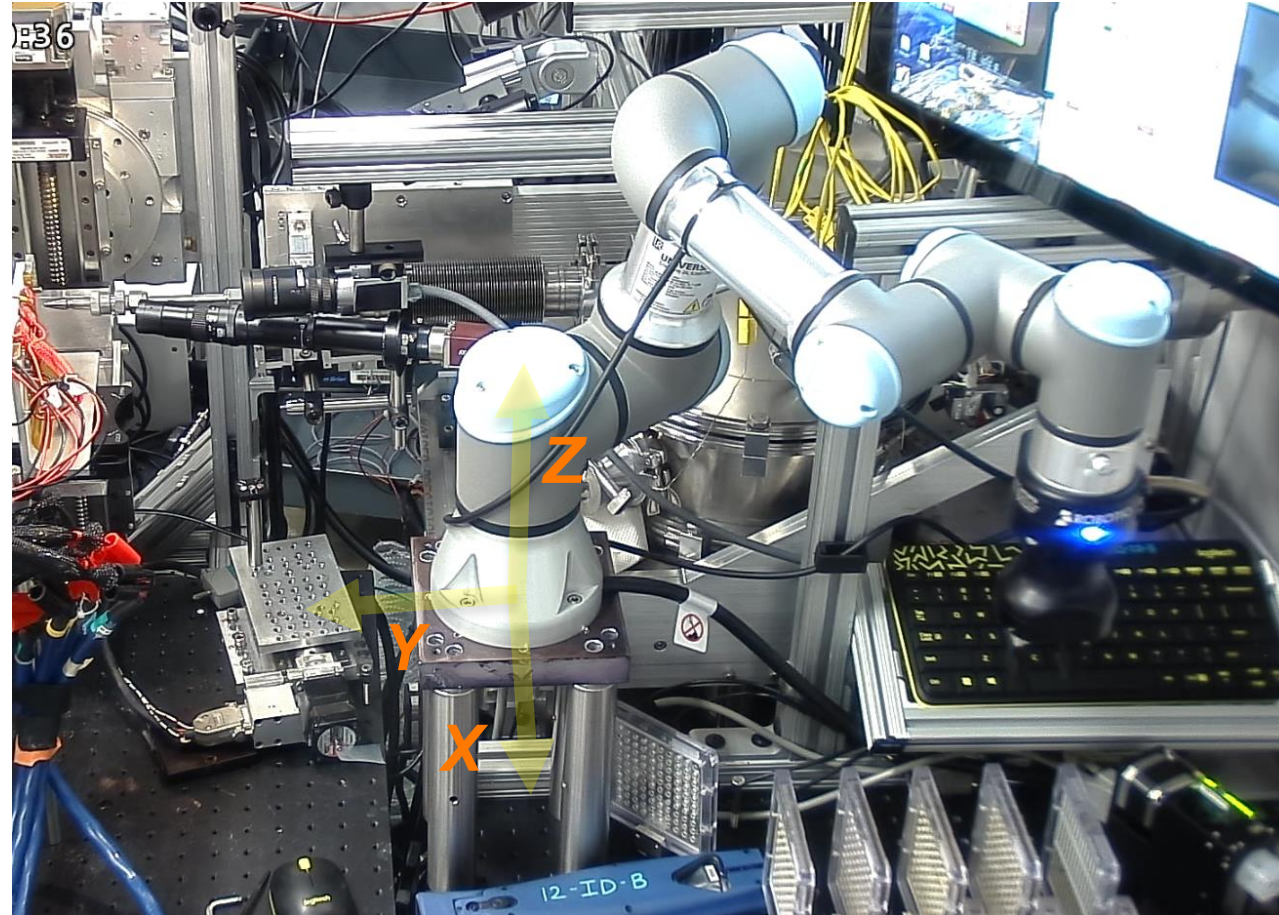
Labels: Forward Ki, Inverse Kinematics (IK),  $x, y, \theta$

UR3:  $q: q_1, q_2, \dots, q_6$

$p: X, Y, Z, ax, ay, az$

TCP position and orientation.

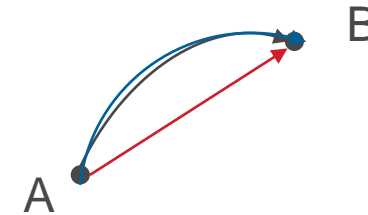
# Coordinate definition





# How to program on the teach pendant

- Free drive to a position A
- Free drive to a position B
- Decide how you like to move from A to B
  - Movej : move minimum number of joints simultaneously
  - MoveL
  - MoveC
  - MoveP
- If you want to wait for a signal, define the signal
- Save and run
- The graphical program will be saved as a UR script
  - /data/program/\*.script



UR5 (ur-20205501124:0) - VNC Viewer

PROGRAM <unnamed>\*  
INSTALLATION default\*

Run Program Installation Move I/O Log

New... Open... Save...

Local

Basic

Move  
Waypoint  
Direction  
Wait  
Set  
Popup  
Halt  
Comment  
Folder

Advanced  
Templates  
URCaps

Command Graphics Variables

MoveP

Specify how the robot will move between waypoints.

The values below apply to all child waypoints and depend on the selected movement type.

Set TCP: Use active TCP

Tool Speed: 250.0 mm/s

Feature: Base

Tool Acceleration: 1200.0 mm/s<sup>2</sup>

Blend with radius: 25.0 mm

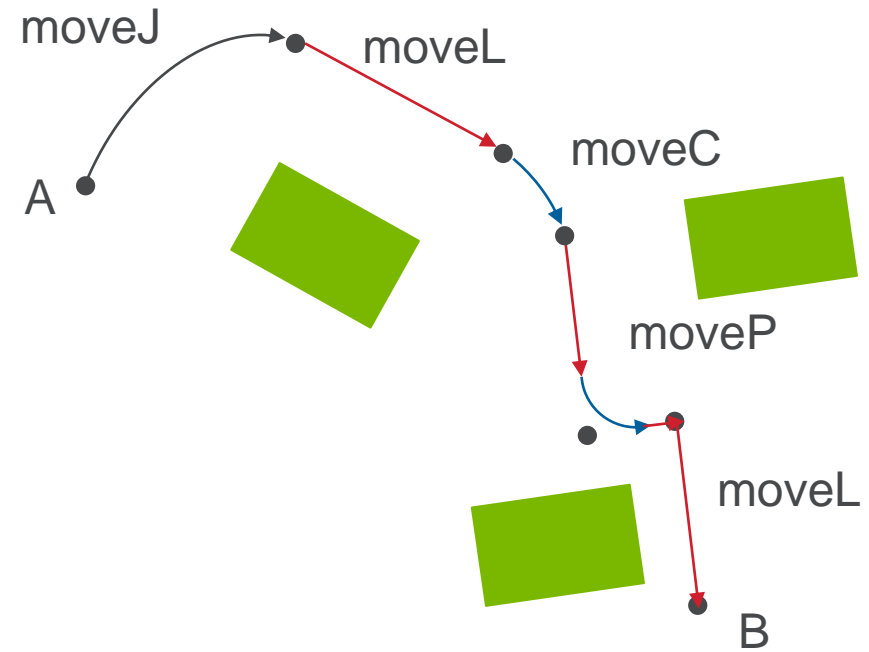
Use joint angles

Reset

+ Add circle move

Normal Speed 100% Simulation

# Teaching waypoints



# Advanced scripting

The screenshot shows the UR5 VNC Viewer interface. At the top, the window title is "UR5 (ur-20205501124:0) - VNC Viewer". Below the title bar is a toolbar with icons for Run, Program, Installation, Move, I/O, Log, and file operations (New, Open, Save). The main interface is divided into several sections:

- Left Panel:** A navigation menu with "Basic" and "Advanced" sections. Under "Advanced", there are options for Loop, SubProg, Assignment, If, Script, Event, Thread, Switch, Timer, Screwdriving, and Home. Below these are "Templates" and "URCaps".
- Program Tree:** A tree view showing a "Robot Program" folder containing two items: "1" and "2 <empty>".
- Command Panel:** A panel with tabs for "Command", "Graphics", and "Variables". The "Command" tab is active, showing a "Program" section with instructions: "Here you can program your robot to do tasks." and "To program your robot, select the nodes from the **Node List** and they will appear on the **Program Tree**." Below this are two panes: "Node List" (empty) and "Program Tree" (containing a small yellow icon). At the bottom of the Command panel are three checkboxes: "Add Before Start Sequence" (unchecked), "Set Initial Variable Values" (unchecked), and "Program Loops Forever" (checked).
- Bottom Panel:** A control bar with a "Normal" status indicator, a "Speed" slider set to 100%, and a "Simulation" toggle switch.

# x11vnc

- Having the Pendant screen on your PC.
- Debian installation file can be obtained from the vendor (Tanaka).
- Steps to do:
  - Login to the control box using putty with an IP of 164.54.xxx.xxx.
  - Run x11VNC
  - Then on your windows computer, use a VNC client such as “VNC viewer” and connect to 164.54.xxx.xxx:5901

# Operation of UR3

- GUI mode
- Script level (using UR Script)
- C-API level

## 1.5 Function

A function is declared as follows:

```
def add(a, b):  
    return a+b  
end
```

The function can then be called like this:

```
result = add(1, 4)
```

It is also possible to give function arguments default values:


```
def add(a=0,b=0):  
    return a+b  
end
```

URScript also supports named parameters.

# Interface for programming

- <https://www.universal-robots.com/articles/ur/interface-communication/overview-of-client-interfaces/>
- Primary/Secondary interfaces
  - Primary can send additional messages
- Real-time Interface
- RTDE interface
- Dashboard Server (controlled by sending simple commands to the GUI over a TCP/IP socket)
- Socket communication (UR robot became a client and Urscript provides socket commands)
  - For example, in polscope, use 'socket\_open',
- XML-RPC (to transfer structured data between programs over sockets)
  - For example, make the robot move using a pose retrieved from a remote camera. The remote camera program provides the next target pose based on the camera image analysis. The camera program can be python or C++.

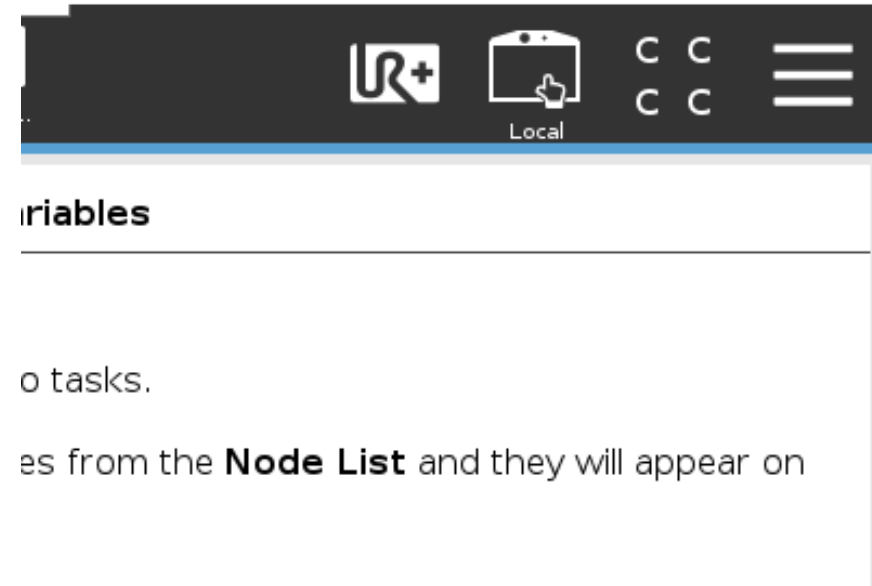
Ideal for off-line programming. For example, to synchronize a real UR3 with a virtual one on a computer.



e-Series							
	Primary		Secondary		Real-time		Real-time Data Exchange (RTDE)
Port no.	30001	30011	30002	30012	30003	30013	30004
Frequency [Hz]	10	10	10	10	500	500	500
Receive	URScript commands	-	URScript commands	-	URScript commands	-	Various data
Transmit	See attachment from the bottom		See attachment from the bottom		See attachment from the bottom		See <a href="#">RTDE Guide</a>
CB-Series							
	Primary		Secondary		Real-time		Real-time Data Exchange (RTDE)
Port no.	30001	30011	30002	30012	30003	30013	30004
Frequency [Hz]	10	10	10	10	125	125	125
Receive	URScript commands	-	URScript commands	-	URScript commands	-	Various data
Transmit	See attachment from the bottom		See attachment from the bottom		See attachment from the bottom		See <a href="#">RTDE Guide</a>

# Python-URX

- Control the robot in REMOTE mode.
- Communicate through the real-time and secondary interface.
  - Main comm interface is the secondary.
  - Using real-time interface only for force sensing, which requires fast comm.
- <https://github.com/SintefManufacturing/python-urx>
  - Sending a UR script and execute
  - This has a script to control a “Robotiq Gripper”



# UR3 programming

## Python-urx

- Uses URscript
  - A command string
    - ✓ Ex) Set\_gravity(5)
    - ✓ Ex) movej([1,1,1,1,1,1],a=1,v=1,r=1)
  - A full program
    - ✓ def myProg():
    - ✓ ....
    - ✓ end
- Support force mode and gripper.

```
class URRobot(object):  
  
    """  
    Python interface to socket interface of UR robot.  
    programs are send to port 30002  
    data is read from secondary interface(10Hz?) and real-  
    time interface(125Hz) (called Matlab interface in documentation)  
    Since parsing the RT interface uses som CPU, and does not support all robots  
    versions, it is disabled by default  
    The RT interfaces is only used for the get_force related methods  
    Rmq: A program sent to the robot i executed immediatly and any running progr  
    am is stopped  
    """  
  
    def __init__(self, host, use_rt=False, use_simulation=False):  
        self.logger = logging.getLogger("urx")  
        self.host = host  
        self.csys = None  
  
        self.logger.debug("Opening secondary monitor socket")  
        self.secmon = unsecmon.SecondaryMonitor(self.host, use_simulation) # dat  
a from robot at 10Hz  
  
        self.rtmon = None  
        if use_rt:  
            self.rtmon = self.get_realtime_monitor()  
        # precision of joint movem used to wait for move completion  
        # the value must be conservative! otherwise we may wait forever  
        self.joinEpsilon = 0.01  
        # It seems URScript is limited in the character length of floats it acce  
pts  
        self.max_float_length = 6 # FIXME: check max length!!!  
  
        self.secmon.wait() # make sure we get data from robot before letting cli  
ents access our methods
```



# 12-ID python codes

- Modification URX for 12IDB use
  - /home/beams15/S12IDB/python\_codes
  - Enable force mode (This is the only part using RT).
  - Timeout time changed
  - The thread for “wait” is changed to prevent an infinite loop when a timeout occurs.
- 12ID code
  - Classes for basic operation: Robot12idb.py
  - Qt GUI : multiheaterWin2.py
  - Some other code to tweaking TCP (tweakRobot class in multiheaterWin2.py).
  - [https://wiki.aps.anl.gov/s12id/index.php/UR3\\_Robot](https://wiki.aps.anl.gov/s12id/index.php/UR3_Robot)

# Robot Control

## Remote Ops software (Python/pyQt5)

Menu to control sample stage using pyEPICS

Menu to control the UR robot using Robot12idb.py

The screenshot shows the 'Sample Stage' menu open over a table with 12 rows. The menu options are: Choose Stage (with sub-menu: Standard, Aux on SampleTable), Move to First Sample, Move to QR reading Position, Move to Dropoff Position, Move to Reference Position, Set Current Position as First Sample, Set Current Position as Dropoff, Set Current Position as Reference, Set Current Position as QR reading, Run All Samples, In Current Samp, and Stop. The table has a 'filename' column and a 'Sample Stage' column.

The screenshot shows the 'Robot Arm' menu open over a table with 12 rows. The menu options are: Choose Robot (with sub-menu: Initialize Robot), Return Sample to Magazine, Load Next Sample Plate, Load Specific Sample Plate, Grab Finger, Release Finger, Move Finger up to transport, Move Finger down to Sample Stage, Move Finger down to Magazine, Transport Finger to Sample Stage, Translate Magazine to Sample N, Pickup Test on Magazine, Pickup Test on Sample Stage, Set Current as Dropoff, Set Current as Pickup, and Show all Motors. The table has a 'filename' column and a 'Sample Stage' column.

User can load their sample information that they have provided through the 12ID web.

Users can choose a sample frame to load, number 0 to 19.

The loaded information goes onto this table and users can change as needed.

Users can perform various data acquisition scans for positioning or collecting scattering data either automatically or one at a time manually.

# Virtual Environment (offline programming)

- CoppeliaSim
  - Young Soo Park (Robotics, AMD), Summer students (Rian Simpson)
- RoboDK
  - The free version allow only one robot or a translator.
- Visual Component (SolidWorks)
  - [https://www.youtube.com/watch?v=JVxYZbDpu\\_8&ab\\_channel=VisualComponents](https://www.youtube.com/watch?v=JVxYZbDpu_8&ab_channel=VisualComponents)
  - OPC UA server/client
- Conversion of CAD file to step files
  - Brian Rusthoven (Design & Draft), Summer students
  - Made a simplified drawing version (Chuck and Byeongdu)

# Future direction

- Autonomous experiment (Weekend users)
  - Self-positioning the robot by camera image analysis coupled with AI
  - Data acquisition and screening
- Easier virtual environment generation
  - A software program to convert complex 3D CAD into a simplified 3D SAT file.