# Acquisition Code

This directory contains the code required to run the SAMURAI system. This currently requires a direct path and connection to the U:/ network drive. This code has been tested lately with python 3.6 but should be backward compatable with 2.7.

## dependencies

* Anaconda with python 3.X
* pyvisa
* cannot remember the last one

### Installing Anaconda and Dependencies

* Install Anaconda
  1. Go to the anaconda download page [here](https://www.anaconda.com/distribution/#download-section)
  2. Select the correct operating system and installer
  3. Download and run the executable and follow the installer to install Anaconda 3.X
* Install pyvisa
  1. Type into the windows search ‘anaconda prompt’ and run the program
  2. In the terminal run conda install -c conda-forge pyvisa to install pyvisa

## Subdirectories

### /support

This subdirectory contains multiple classes and functions to run the SAMURAI system including control for the VNA and Meca500 Robot arm. This contains the following files

* /support/autoPNAGrabber.py
  + python control of PNAGrabber. Requires PNAGrabber executable to be run from command line with -r flag to ‘press’ the ‘meas all’ button without opening the UI
* /support/Meca500.py
  + Class for controlling Meca500 Robot
* /support/metaFileController.py
  + Class for interacting with already written metafile from a measurement
* /support/pnaController.py
  + Class to interact with PNA to get various settings and set some things on it
* /support/samurai\_metaFile.py
  + Class for creating metafile for SAMURAI measurements
* /support/samurai\_support.py
  + Some support functions to use. (mainly functions used when USC came out in Aug. 2018 for phased array calibration)
* /support/BislideController.py
  + A class to control a Velmex Bislide
* /support/samurai\_tktools.py
  + Some functions for quick building of GUIs with Tkinter (Tk)

# SAMURAI Hardware Information

graph TD;  
 A-->B;  
 A-->C;  
 B-->D;  
 C-->D;

(PICTURE OF SETUP)

## Hardware

### Meca 500 6 axis positioner

Small 6 axis positioner.

* [Website](https://www.mecademic.com/products/Meca500-small-robot-arm)
* [User Guide](hardware/datasheets/Meca500-R3-User-Manual.pdf)
* [Programming Manual](hardware/datasheets/Meca500-R3-Programming-Manual.pdf)

### Keysight PNA-X (N5245A)

10MHz to 50GHz VNA. Ports are 2.4mm Male typically with 2.4mm F-F connector savers on them.

* [Datasheet](hardware/datasheets/N5245.pdf)

### Antennas

* Sage-millimeter 17dBi WR-28 Horn antenna
  + [Datasheet](hardware/datasheets/17dBi_horn_sagemillimeter.pdf)
* Sage-millimeter 23dBi WR-28 Horn antenna
  + [Datasheet](hardware/datasheets/23dBi_horn_sagemillimeter.pdf)

### Cables

* Junkosha 2.4mm (M-M) 3m Cables
  + info (link datasheet)
* Junkosha 2.4mm (M-M) 0.25m Cables
  + info (link datasheet)

### Adapters

* Sage-millimeter 2.4mm to WR-28 Adapters

## Networking

Currently, the samurai system is run over a custom local network run through a simple network switch. This connects to the VNA, Meca500 Robot arm, and eventually cameras.

### Remote PNA-X control

A remote Keyboard, Video, Mouse box is used. This allows a keyboard, monitor, and a mouse to be placed far away from our VNA and a single CAT-5 cable (ethernet) to be run between the two. This comprises of a small box with 2 usb ports and a VGA connection. This box is then connected directly via a CAT-5 Cable near the VNA with a usb-B output and a second VGA connection. These two boxes provide remote control over the VNA - NOTE: This is not connected to the local network. These two boxes are only connected to one another and cannot be run over a network. They simply translate the usb and VGA info and transmit over a CAT-5 cable.

### IP and VISA Addresses

* PNA-X = [192.168.0.2](http://192.168.0.2)
  + VISA Address = ‘TCPIP0::10.0.0.2::inst0::INSTR’
* Meca500 = <192.168.0.5>
  + VISA Address = Could not get VISA to work correctly! Connect using sockets.
* IP Webcam = <192.168.0.11>
  + Username: admin – Password: 123456
  + A live stream will show up if you go to the above address and login
  + A VLC stream has higher latency but can be connected by the following steps:
    1. Open VideoLAN (VLC with the construction cone icon)
    2. Select Media->Open Network Stream...
    3. Enter rtsp://admin:123456@10.0.0.101:554/cam1/mpeg4 and click connect
    4. To take a snapshot click Video->Take Snapshot. This will save a snapshot to the users Pictures folder from which it can then be renamed and copied
    - The VLC stream has not always been reliable and may freeze. For this reason it is recommended to use the web interface except when taking snapshots of the setup
* Computer = [192.168.0.1](http://192.168.0.1)
  + VISA Address = N/A (local loopback is 127.0.0.1)
  + Setting Network adapter settings for local network:
    1. Go to Control Panel->Network and Internet->Network Connections
    2. Right click on the network controller for the local network and select Properties (admin status required)
    3. Click on TCP/IPv4 and then click Properties
    4. Click the radio button for Use the following IP address and type in the following parameters
       - IP address = 10.0.0.1
       - subnet mask = 255.0.0.0
       - Default gateway = DO NOT POPULATE
    5. Then click OK and Close to close out of the properties menu. You should now be able to access items on the local network.
* Network Switch = <192.168.0.239>
  + Password is password
* Optitrack Cameras = [192.168.0.???](192.168.0.255)
  + These IP addresses are unkown to the user
  + It is possible at some point in time these may conflict with one of the other devices on the network. If so change the IP of whatever device is conflicting.

# Running Python Scripts

This section explains how to run the python scripts that are mentioned in this document

## Running with the Spyder IDE

1. Open the Spyder IDE (Make sure to use Python 3.7)
2. Open the script in Spyder
3. Press the Green play button at the top of the window

## Running from the Anaconda command prompt

1. Run the program ‘Anaconda Prompt’
   * This can be done by searching for this in the Windows toolbar
2. In the prompt type python <script\_directory>/<script\_name>.py where <script\_directory> and <script\_name> create the path to your script

# Mounting and Unmounting the antennas

## Moving the Positioner to the mounting position

The robotic positioner can be moved to a location that is easier to mount/unmount the antennas with the following code:

from samurai.acquisition.SAMURAI\_System import SAMURAI\_System  
mysam = SAMURAI\_System() #initialize the class  
mysam.connect\_rx\_positioner() #connect and home the positioner  
mysam.move\_to\_mounting\_position() #move to the mounting position

Once the antennas have been remounted, return the positioner to its home position and disconnect with the following code:

mysam.zero() #move back to its home position  
mysam.disconnect\_rx\_positioner() #disconnect the positioner

## Connecting the antennas

Both the transmit and recieve antenna should always be contained in a 3D printed mounting holder. The newest version of this holder will have 3 steel ball bearings that fit into grooves on the Robot mount. Slide the antenna and its mount into the recieving side on the robot and connect the three 3mm nuts to snugly hold together the antenna and recieving mount. DO NOT OVERTIGHTEN THESE NUTS. The connection only needs to be lightly tightened (finger tight plus 1 turn or so). Overtightening will warp the plastic and damage the mount.

# Demo the SAMURAI System

A script has been made to run quick demonstration of the SAMURAI system. This demo will do the following:

1. Perform a 35x35 element planar sweep at 40 GHz
2. Measure and plot 3D beamformed data for the current channel
3. Measure and plot a PDP from the measured frequency range start/stop/step = 26.5GHz/40GHz/10MHz at a single aperture position

## Running the Demo

In order to run the demo the following steps must be taken

1. Open the Spyder IDE or the Anaconda command prompt
2. Run the script \\cfs2w\67\_ctl\67Internal\DivisionProjects\Channel Model Uncertainty\Measurements\demo\quick\_beamform\_demo\channel\_test.py
   * See the ‘Running Python Scripts’ section for instructions on how to run this file

# Running the SAMURAI System

This section covers the steps required to run a SAMURAI measurement

## Running from script

This section shows how to run from a premade python script. This requires the lowest amount of user input and is therefore the recommended method of control.

### 1. Create a new SAMURAI measurement directory

1. Make a copy of meas\_template in the directory U:\67Internal\DivisionProjects\Channel Model Uncertainty\Measurements\Synthetic\_Aperture
2. Rename the copy to the current date in the format mm-dd-yyyy
   * From here on, this newly created directory will be referred to as <working-directory>
3. Copy and paste the correct comma separated value (CSV) file containing the positions into <working-directory>/synthetic\_aperture/raw
   * Some commonly used templates are contained in <working-directory>/synthetic\_aperture/raw/position\_templates directory.
   * Once the desired CSV file has been copied, rename it positions.csv

### 2. Perform 2 Port VNA Calibration

1. In the windows file explorer navigate to <working-directory>/cal/calibration\_pre
2. double click on ‘cal.pnagrabber’ to start PNAGrabber for the calibration.
3. Attach each of the standards to the calibration plane with the naming convention <standard-port-1>\_<standard-port-2>.s2p
   * (e.g. load\_short.s2p is load on port 1 and short on port 2)
4. When the calibration is completed, make a copy of each of the .s2p files generated and put them into the <working-directory>/cal/calibration\_pre/raw folder

### 3. Open and update the script

1. Open the file <working-directory>/synthetic\_aperture/raw/run\_script.py
   * This contains the code to run the sweep along with metadata information and other input parameters
2. Set the csv file path by changing the line position\_file = './position\_templates/samurai\_planar\_dp.csv' to set position\_file to the relative path to the csv file of positions
3. Set the motive dictionary for camera tracking. For all rigid bodies create a new line with the entry motive\_dict['<rigid-body-name>'] = None. For each marker create a new line motive\_dict['<marker-name>'] = <marker-id-number>
4. Add any experiment info and notes to metafile\_info\_dict['experiment'] and metafile\_info\_dict['notes']
5. Add any additional metafile info to to the metafile\_info\_dict dictionary.

### 4. Run the script

Run the newly updated run\_script.py using the directions listed in section ‘Running Python Scripts’. This will save all data into the same directory as the run script.

## Running from python command line interface (CLI) (DEPRECATED)

*[CLI]: Command Line Interface*  
[IDE]: Integrated Development Environment (e.g. Spyder)  
The following steps are to run a SAMURAI measurement from the python CLI. The steps using the python CLI here are valid for the integrated command line within the Spyder IDE. While these steps will be similar using a basic python setup, the importing of the SAMURAI classes and libraries may be a bit more complex.

### 1. Create a new SAMURAI measurement directory

1. Make a copy of meas\_template in the directory U:\67Internal\DivisionProjects\Channel Model Uncertainty\Measurements\Synthetic\_Aperture
2. Rename the copy to the current date in the format mm-dd-yyyy
   * From here on, this newly created directory will be referred to as <working-directory>
3. Copy and paste the correct comma separated value (CSV) file containing the positions into <working-directory>/synthetic\_aperture/raw
   * Some commonly used templates are contained in <working-directory>/synthetic\_aperture/raw/position\_templates directory.
   * Once the desired CSV file has been copied, rename it positions.csv

### 2. Perform 2 Port VNA Calibration

1. In the windows file explorer navigate to <working-directory>/cal/calibration\_pre
2. double click on ‘cal.pnagrabber’ to start PNAGrabber for the calibration.
3. Attach each of the standards to the calibration plane with the naming convention <standard-port-1>\_<standard-port-2>.s2p
   * (e.g. load\_short.s2p is load on port 1 and short on port 2)
4. When the calibration is completed, make a copy of each of the .s2p files generated and put them into the <working-directory>/cal/calibration\_pre/raw folder

### 3. Import the SAMURAI\_System Module

1. Open the python CLI (e.g. the command window in Spyder)
2. Within the command line type the following

from samurai.acquisition.SAMURAI\_System import SAMURAI\_System

* NOTE: FOR NEW COMPUTERS ONLY - the code must be cloned from the gitlab repo and the directory containing the cloned samurai directory must be added the systems PYTHONPATH.

### 4. Create a SAMURAI\_System Object

1. With the SAMURAI\_System module imported, create a SAMURAI\_System object by typing mysam = SAMURAI\_System() into the CLI.

### 5. Change directory to measurement directory

1. Change the directory to <working-directory>/synthetic\_aperture/raw by running the following set of commands:

* import os  
  os.chdir(<working-directory>/synthetic\_aperture/raw)
* OR in certain iPython CLIs
* cd <working-directory>/synthetic\_aperture/raw

### 6. Mount the Antennas

1. Mount the Tx Antenna (usually port 2) to the fixed holder
2. Move the SAMURAI Robot to the mountain position using the commands below
   * The mysam object must exist for this step to work
   * Keep in mind, after this code the positioner is still connected and activated after these commands

* mysam.connect\_rx\_positioner() #connect and home the positioner  
  mysam.move\_to\_mounting\_position() #move to an easy position to mount the antenna

1. Use the four m3 screws to attach the Antenna to the Meca500

### 8. Open the Robot’s Web interface (Optional)

Before running the sweep we can perform the extra step of viewing the robot’s movement and status through its web monitoring interface. To open up the web monitoring interface:

1. Open a web browser (tested in chrome)
2. type [10.0.0.5](http://10.0.0.5) into the address bar
3. In the web interface, click the ‘Connection’ button on the top toolbar.
4. In the pop-up window select ‘Monitoring’ and click ‘Connect’

### 8. Run the Synthetic Aperture Sweep

Now we can begin the sweep

1. Ensure the working directory is set to <working-directory>/synthetic\_aperture/raw (see step 5)
   * Some editors/IDE’s (e.g. spyder) show this in a top bar of the screen
   * The current directory can be found from a python CLI by typing import os; os.getcwd()
2. Type the following code and hit enter to begin the sweep
   * This step assumes the robot has previously been connected and initialized (activated and homed)
   * This also assumes the mysam object has already been created

* mysam.csv\_sweep('./','./positions.csv',template\_path='template.pnagrabber');disconnect\_rx\_positioner()
  + NOTE: If a csv file is being tested, the flag run\_vna=False can be added to the mysam.csv\_sweep() call to prevent the VNA from running
  + NOTE: The robot can also be put into simulation mode where all commands are sent and the web interface shows the robot moving, but the robot does not physically move. For more information on this reference the code documentation.

### 9. Unmount the Antennas

1. Create mysam object if it does not exist
2. Connect to positioner (refer to ‘Mount the Antennas’ section)

### 10. Collect and Save data

1. copy data from <working-directory>/synthetic\_aperture/raw to <working-directory>/synthetic\_aperture/
2. Perform post-calibration in <working-directory>/cal/calibration\_post (refer to ‘Perform 2 Port VNA Calibration’ section)

### Example python script

Here we have an example python script to run the sweep. This is assuming we have already created a <working-directory>. This also assumes we have placed a pnagrabber template named template.pnagrabber and a list of positions called positions.csv in <working-directory>/synthetic\_aperture/raw.

import os #import os for chdir  
from samurai.acquisition.SAMURAI\_System import SAMURAI\_System #import the samurai system class  
  
mysam = SAMURAI\_System() #create a samurai system object  
mysam.connect\_rx\_positioner() #connect to the Meca500 (or other positioner)  
mysam.move\_to\_mounting\_position() #move to the position to unmount the antenna for calibration  
  
###################################  
# Unmount antenna from Meca500  
###  
# PERFORM CALIBRATION HERE  
###  
# Mount antenna onto Meca500  
###################################  
  
mysam.zero() #return the robot to its zero position  
os.chdir('<working-directory>/syntetic\_aperture/raw') #change into our measurement directory  
mysam.csv\_sweep('./','./positions.csv',template\_path='./template.pnagrabber') #run the csv sweep with the vna  
  
mysam.disconnect\_rx\_positioner() #disconnect from the Meca500 when finished

## Running from the Graphical User Interface (GUI) SAMURGUI

Code not currently complete

# Measurement TODO List

* ☒ Line of Sight Measurements
* ☒ Cylinder Measurement
* ☒ Cylinder non-LOS measurement
* ☐ Active (2 source) AoA measurements

# Code Editing TODO List

## Current Work

* ☐ Beamforming code for 3D and cylinder
* ☐ Angular resolution for each different aperture
* ☐ Read on AoA verification work
* ☐ Read on AoA algorithm work
* ☐ Create basic MUSIC, SAGE, Etc. algorithms
* ☐ Speed these up

## On backburner

* ☐ allow user to move\_to\_mounting\_position from SAMURGUI
* ☐ allow user to run csv\_sweep without connect/disconnect from SAMURGUI
* ☐ add metafile editing interface to SAMURGUI
* ☐ add Meca500 status viewer in SAMURGUI