Claruis High Frequency Linear Scanner

Contents

[Links 2](#_Toc148521899)

[Code Notes 3](#_Toc148521900)

[Pysidecaster.py 3](#_Toc148521901)

[pyCaster.py 3](#_Toc148521902)

[Weekly report 5](#_Toc148521903)

[Week 1 5](#_Toc148521904)

[Week 2 5](#_Toc148521905)

[Notes 7](#_Toc148521906)

[Week 3 7](#_Toc148521907)

[Week 4 7](#_Toc148521908)

# Links

Clarius cast api:  
<https://github.com/clariusdev/cast/tree/master>

Clarius cast api lib:   
<https://github.com/clariusdev/cast/releases>

# Code Notes

## Help pyclariuscast

NAME

    pyclariuscast

CLASSES

    Boost.Python.instance(builtins.object)

        Caster

        Imu

    class Caster(Boost.Python.instance)

     |  Class to wrap the Cast API.

     |

     |  Method resolution order:

     |      Caster

     |      Boost.Python.instance

     |      builtins.object

     |

     |  Static methods defined here:

     |

     |  \_\_init\_\_(...)

     |      \_\_init\_\_( (object)arg1, (object)arg2, (object)arg3, (object)arg4, (object)arg5, (object)arg6) -> None

     |

     |  \_\_reduce\_\_ = <unnamed Boost.Python function>(...)

     |

     |  connect(...)

     |      connect( (Caster)arg1, (str)ipAddress, (int)portNumber, (str)certificate) -> bool :

     |          Makes a connection attempt to a scanner on the same Wi-Fi network

     |

     |  destroy(...)

     |      destroy( (Caster)arg1) -> bool :

     |          Destroys the cast library

     |

     |  disconnect(...)

     |      disconnect( (Caster)arg1) -> bool :

     |          Disconnects from an existing connection

     |

     |  enableParam(...)

     |      enableParam( (Caster)arg1, (str)prm, (bool)enable) -> bool :

     |          Enables or disables a level parameter

     |

     |  init(...)

     |      init( (Caster)arg1, (str)keysDir, (int)width, (int)height) -> bool :

     |          Initializes the cast library

     |

     |  isConnected(...)

     |      isConnected( (Caster)arg1) -> bool :

     |          Retrieves connected status of the caster

     |

     |  readRawData(...)

     |      readRawData( (Caster)arg1) -> object :

     |          Retrieves the raw data from a previous request, will be bundled as a compressed tar file

     |

     |  requestRawData(...)

     |      requestRawData( (Caster)arg1, (int)start, (int)end, (bool)lzo) -> int :

     |          Requests raw data from the device

     |

     |  separateOverlays(...)

     |      separateOverlays( (Caster)arg1, (bool)en) -> bool :

     |          Sets the flag to separate overlays into separate images

     |

     |  setFormat(...)

     |      setFormat( (Caster)arg1, (int)fmt) -> bool :

     |          Sets the image format

     |

     |  setOutputSize(...)

     |      setOutputSize( (Caster)arg1, (int)width, (int)height) -> bool :

     |          Sets the output size of the images

     |

     |  setParam(...)

     |      setParam( (Caster)arg1, (str)prm, (float)value) -> bool :

     |          Sets a low level parameter

     |

     |  setPulse(...)

     |      setPulse( (Caster)arg1, (str)prm, (str)shape) -> bool :

     |          Sets the pulse shape for a low level parameter

     |

     |  userFunction(...)

     |      userFunction( (Caster)arg1, (int)fn, (float)value) -> bool :

     |          Calls a user function to control parameters

     |

     |  ----------------------------------------------------------------------

     |  Data and other attributes defined here:

     |

     |  \_\_instance\_size\_\_ = 112

     |

     |  ----------------------------------------------------------------------

     |  Static methods inherited from Boost.Python.instance:

     |

     |  \_\_new\_\_(\*args, \*\*kwargs) from Boost.Python.class

     |      Create and return a new object.  See help(type) for accurate signature.

     |

     |  ----------------------------------------------------------------------

     |  Data descriptors inherited from Boost.Python.instance:

     |

     |  \_\_dict\_\_

     |

     |  \_\_weakref\_\_

    class Imu(Boost.Python.instance)

     |  Method resolution order:

     |      Imu

     |      Boost.Python.instance

     |      builtins.object

     |

     |  Static methods defined here:

     |

     |  \_\_init\_\_(...)

     |      \_\_init\_\_( (object)arg1) -> None

     |

     |  \_\_reduce\_\_ = <unnamed Boost.Python function>(...)

     |

     |  ----------------------------------------------------------------------

     |  Data descriptors defined here:

     |

     |  ax

     |      accelerometer x

     |

     |  ay

     |      accelerometer y

     |

     |  az

     |      accelerometer z

     |

     |  gx

     |      gyroscope x

     |

     |  gy

     |      gyroscope y

     |

     |  gz

     |      gyroscope z

     |

     |  mx

     |      magnetometer x

     |

     |  my

     |      magnetometer y

     |

     |  mz

     |      magnetometer z

     |

     |  qw

     |      quaternion w

     |

     |  qx

     |      quaternion x

     |

     |  qy

     |      quaternion y

     |

     |  qz

     |      quaternion z

     |

     |  tm

     |      timestamp

     |

     |  ----------------------------------------------------------------------

     |  Data and other attributes defined here:

     |

     |  \_\_instance\_size\_\_ = 136

     |

     |  ----------------------------------------------------------------------

     |  Static methods inherited from Boost.Python.instance:

     |

     |  \_\_new\_\_(\*args, \*\*kwargs) from Boost.Python.class

     |      Create and return a new object.  See help(type) for accurate signature.

     |

     |  ----------------------------------------------------------------------

     |  Data descriptors inherited from Boost.Python.instance:

     |

     |  \_\_dict\_\_

     |

     |  \_\_weakref\_\_

FUNCTIONS

    version(...)

        version() -> str :

            Version of this library

## Pysidecaster.py

Is has a qt gui that let you stream the images for the device, change basic parameters like gain and depth, and take image and recording and save it to the device.

Nots about the functions:

**Cast.userFunction()**

Is used to do things in the mobile app like take a screenshot on the app or change the gain or depth. Will only do thing in the mobile app though.

Example:

CMD\_FREEZE: Final = 1

CMD\_CAPTURE\_IMAGE: Final = 2

CMD\_CAPTURE\_CINE: Final = 3

CMD\_DEPTH\_DEC: Final = 4

CMD\_DEPTH\_INC: Final = 5

CMD\_GAIN\_DEC: Final = 6

CMD\_GAIN\_INC: Final = 7

CMD\_B\_MODE: Final = 12

CMD\_CFI\_MODE: Final = 14

def tryFreeze():

            if cast.isConnected():

                cast.userFunction(CMD\_FREEZE, 0)

the args could be an integer like cast.userFunction(1,0), will also work

## pyCaster.py

This script does not use a qt gui. Use inputs in the terminal when run to interact with the script. Can connect to the probe but cannot save images to the computer. But can take image on the mobile app.

# Weekly report

## Week 1

I got connected to the Clarius device. I had to be invited to the intuition from the administrator to be able to use the device. I was able to use the app to see the ultrasound data and was able to record a short recording of my arm moving and store it on my device (can also store on the Clarius cloud). Found some python api on github, not sure on how they work, or which one is better or how to connect it.

Using Clarius cast API

<https://github.com/clariusdev/cast/tree/master>

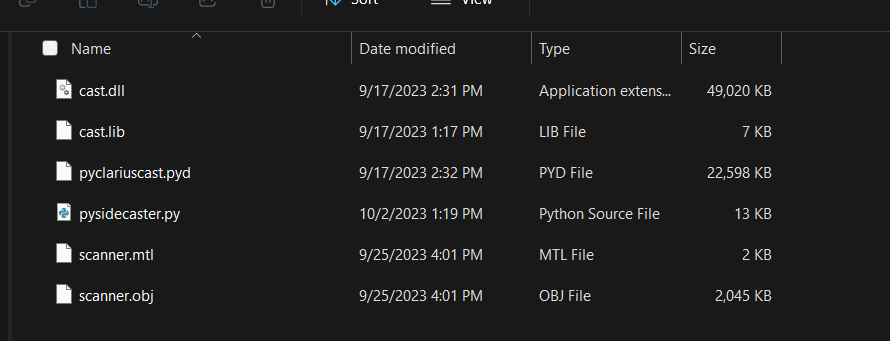
Can not find the library for the python example code.

## Week 2

Found the library download location  
<https://github.com/clariusdev/cast/releases>

I downloaded the windows.x64 version(download for whatever os you have). Inside the zip, there are different versions of the lib for different version of python, get the pyclariuscast.pyd file for the python version that you are using. There is also the cast.dll and cast.lib In the zip file, you will as need them. Download the pysidecast.py python example and scanner.mtl and scanner.obj from the github. You will also have to download PySide6 for this example if you don’t already have it. Use

pip install PySide6

to install it. The file directory should look like something like this:  


Now, on the Clarius mobile app, in settings change the clarius cast to Research(5828)  
(Note: if does not show try restarting app or connect to the devices network)  
A screenshot of a device

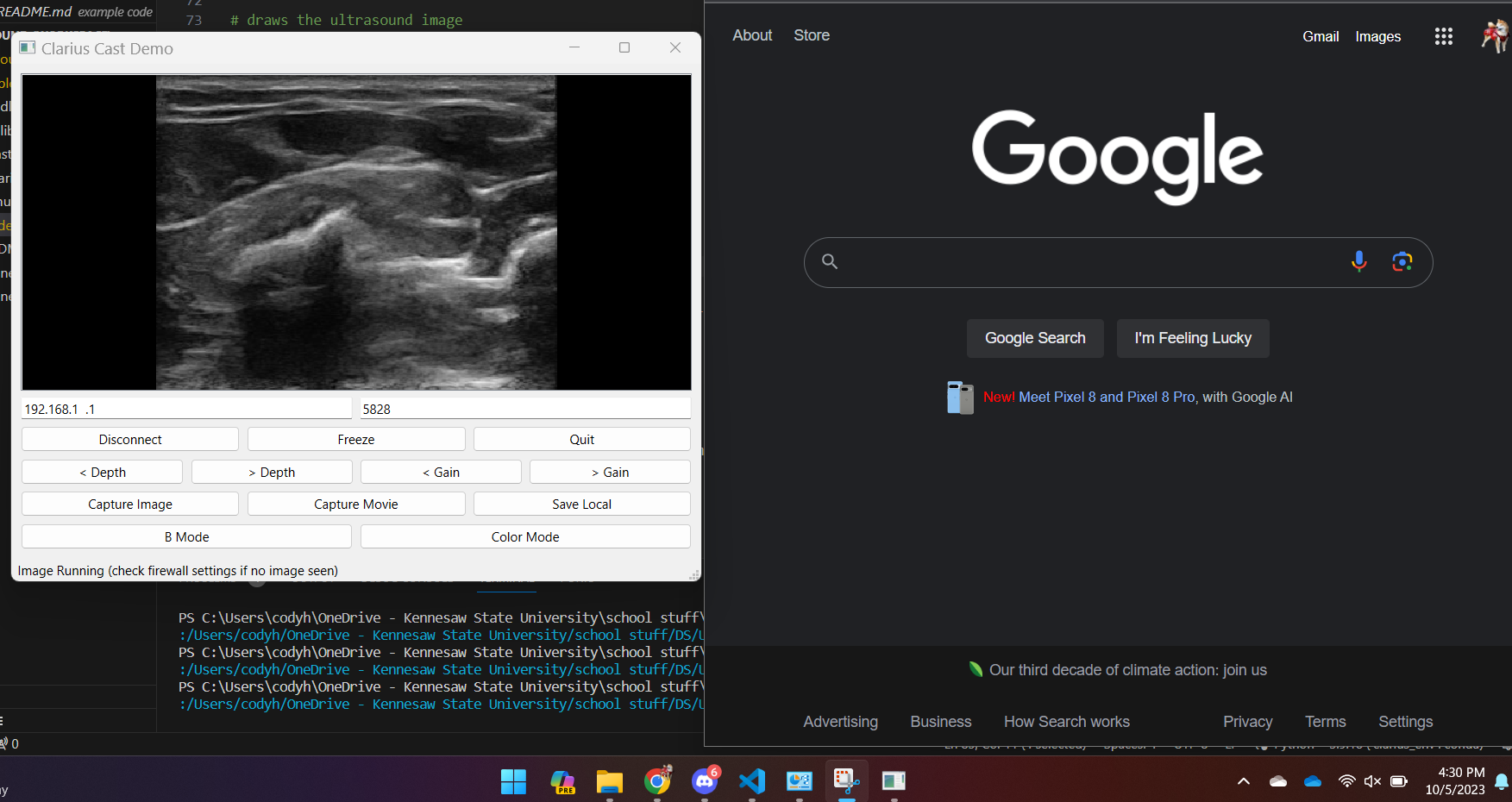
Description automatically generated  
Now turn on and connect the Clarius probe using the app. You will also have to connect your computer to the probe using Wi-Fi. To get the probes Wi-Fi name look at what your phones Wi-Fi is connected to. The password to connect will be on the phone’s clipboard when you connect the phones to the probe for 60 seconds. So go somewhere you can paste text like a web browser and paste it there to see it.

Once you have the probe connected to the phone and computer, you can run pysidecast.py. The Ip and port will be on the mobile app at the top.  
A screenshot of a device

Description automatically generated

When you connect the desktop application to the probe you can take a picture or record. If you don’t see the probe stream on the desktop app on window, you will have to disable your firewall.

### Notes

If I use the wifi dongle from the emg armband I can connect to the KSU wifi and the probe at the same time.  


## Week 3

Did not get much done this week. Just trying to understand the code and how it works. Not going well, no or could not find documentation about cast api for python. Tried print(help(pyclariuscast)), does not really help.

## Week 4

Kind of understand the code, kind of. I mostly studied pysidecast.py and ptcaster.py, I have put notes on what I learned in the notes section. This week will try to make a basic script to connect to the probe and take an image and save it to the computer, both for greyscale and rf. If I get that done, will make qt gui to stream the imagines using cast api like in the pysidecast.py demo.

I made a short script to connect to the probe using it ip and port, used pycaster.py as an example for this. Was able to connect to the probe using my script.

Now I will work on save an image to my computer.

## Week 5

Was able to save images. Now it will try to work on getting it to run in qt or getting the script to run with the armband script at the same time. Might try threading or subprocess, probably threading will work best. Also need to try rf

### Notes

From what I understand when you connect to the device and the **image is not frozen**. By frozen I mean to run the user function to freeze the image on the mobile app. The device will stream images to the program. When the program receives an image, it will run the newProcessedImage function, where it will save the image to a file. To stop the program, you must freeze the image first, I don’t know why.