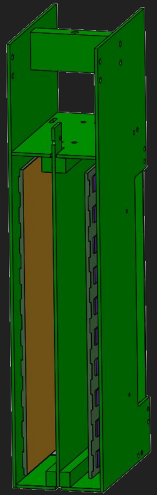
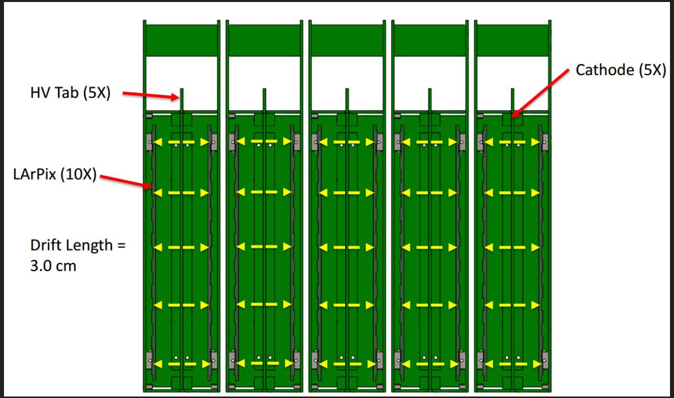
UTA -- LArPix QC Preparations

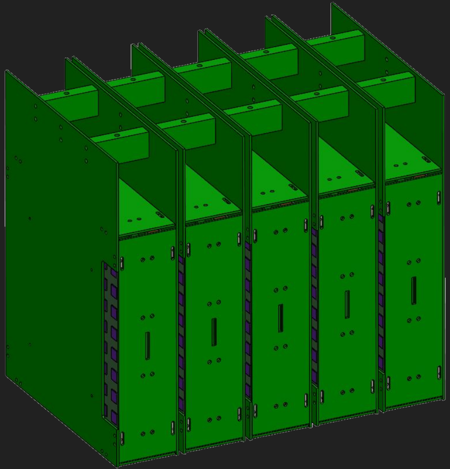
Team Members: Youstina Abraham, Jonathan Asaadi, James deLeon, Kit Kessler, Curtis Kizer-Pugh

Introduction

- **UTA will attempt to perform QC testing of the LArPix tiles by operating them in a functional LArTPC TPC**
  - Mechanical structure is identical to Single Cube design, just with shorter drift length
    - Allows for many tiles to be tested at once
      - Targeting 10 tiles per cryo-cycle





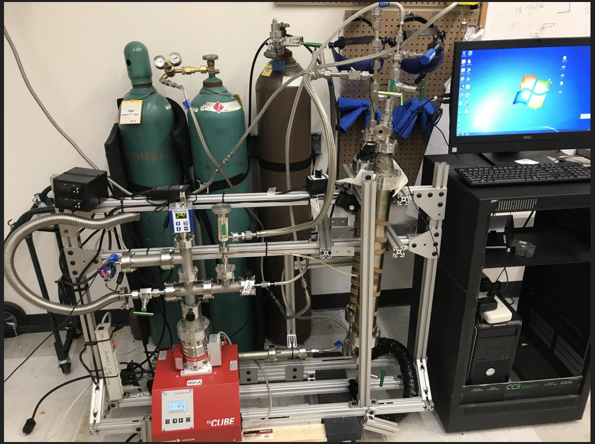


Introduction

- **These tests will take place inside UTA's "larger" cryostat**
  - Cryofab 3048, 540 Liter Cryostat
    Inner Diameter: 75.5 cm
    Height: 121 cm

- **We'll utilize our liquid argon purification rig**
  - Getter Max 133 Cu granulas
    Sigma Aldrich 208604 Molecular Sieves
  - Associated scroll pumps (not shown) and turbo pumps for pumping down and purging the system

- **We've been working to get various aspects of our system ready to receive boards for testing**

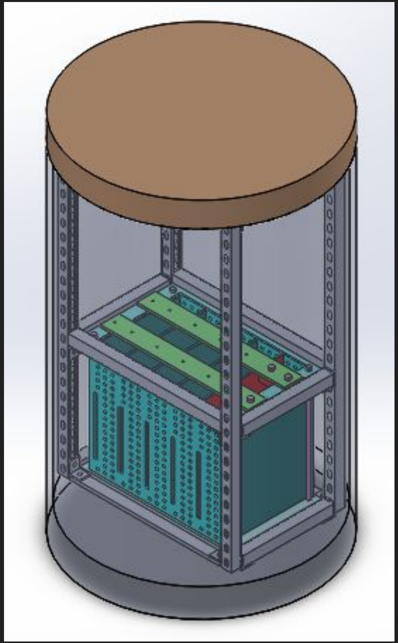


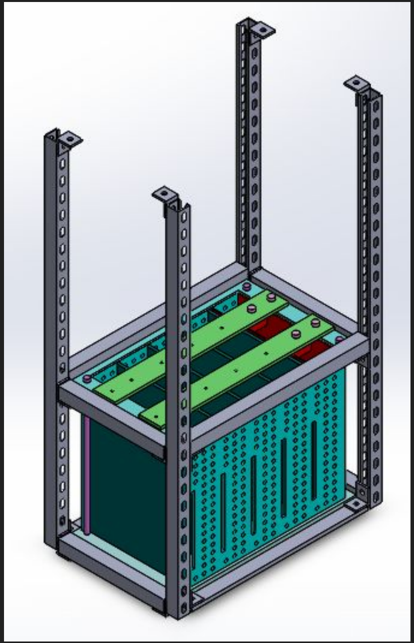
Introduction

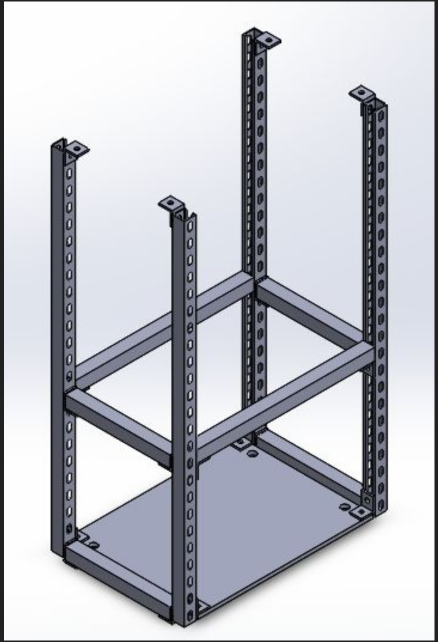
- **Team right now is myself and 4 undergraduate researchers**
  - Hopefully a new post-doc will be joining the team later this year
- **Work right now is divided across a few fronts**
  - **Curtis:** Testing rig design
  - **James:** LArPix Readout and Slow controls
  - **Youstina and Kit:** Liquid Argon Purification System
  - **Jonathan:** Deputy Trouble Maker
    - We've been trying to cross-pollinate these different aspects so lots of people know other aspects since it will take a team effort to perform these tests
    - We are still limiting the number of people in the lab at one time to 2 people per shift, so progress building and testing is slower than it could be….but maintains safety
- **We'll do a little bit of "presentation karaoke" so you can hear from the people doing the work and ask them each questions**

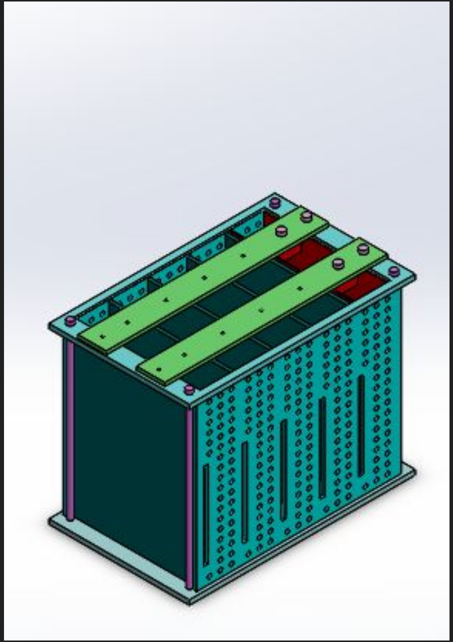
Testing Rig

Testing Rig


TPC Basket + Support Structure in Cryostat

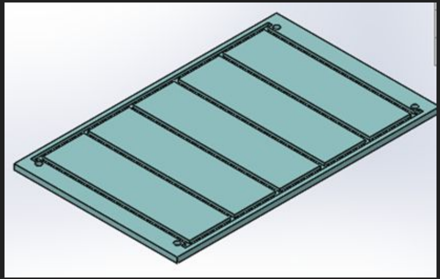

TPC Basket + Support Structure

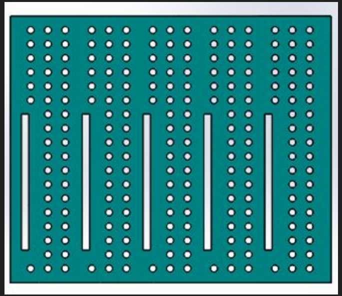

Support Structure


TPC Basket

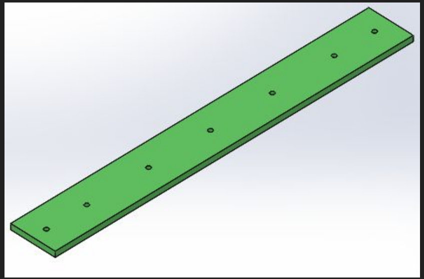
TPC Basket

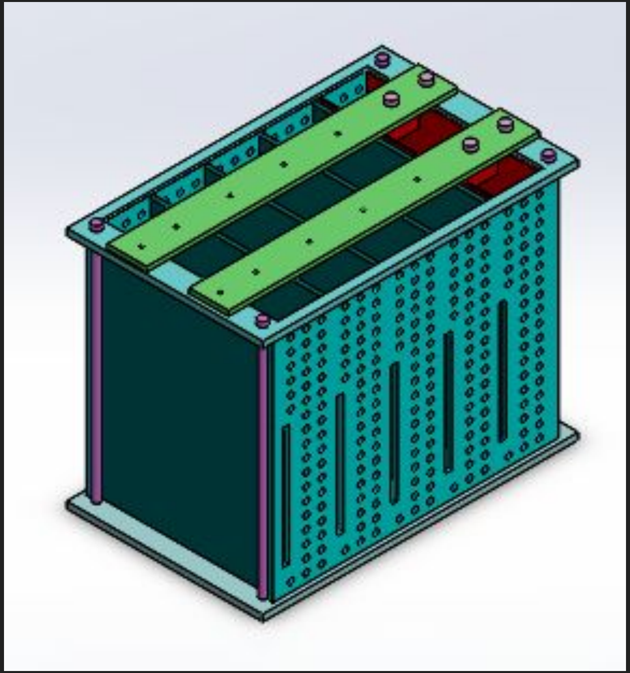
- **Slotted G10 Construction**
  - Stainless Steel Tension Rods & Compressible Washers
- **Holds Five TPCs at 7.2 mm apart**


Base


Side Panels

Top

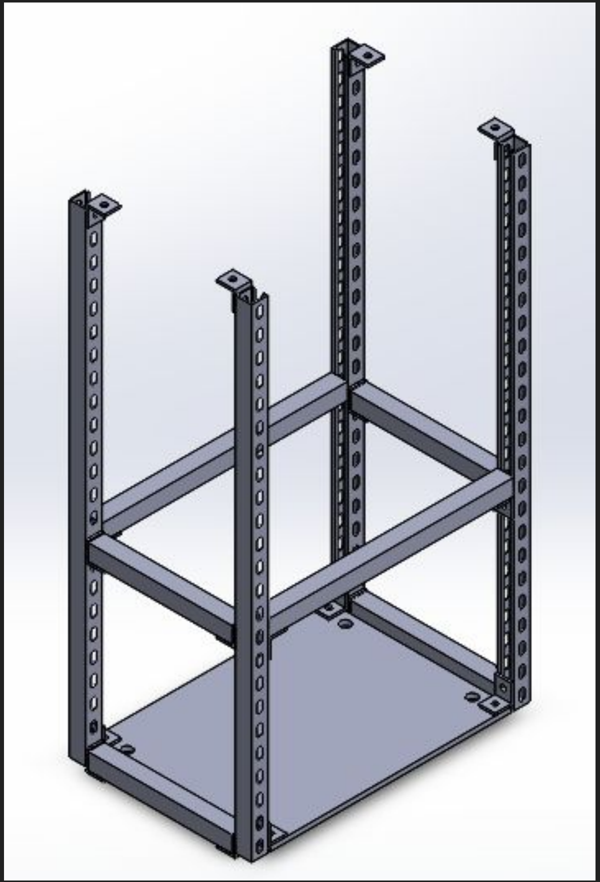

Support Brace


TPC Basket with TPC module inside

Basket Support Structure

- **Made from Stainless Steel Strut**
  - G10 Shelf on the bottom
- **Will attach to the Cryostat's Lid**
  - Working now to find a shop to machine the lid to support fixing the support to the lid
  - Helps keep the cabling simpler
- **Modular Design**
  - Adjustable Braces
  - Supports Ancillary Sensor Attachment

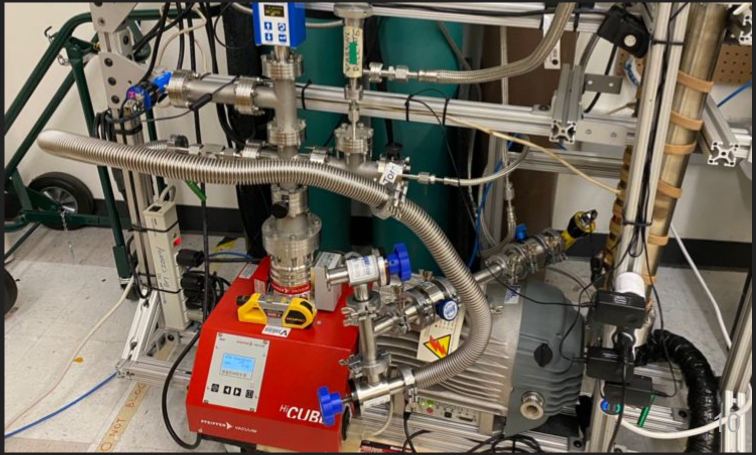
- **Comments and feedback welcome on the design**
  - Once we incorporate people's thoughts, we will order parts (hopefully next week) and get to building!



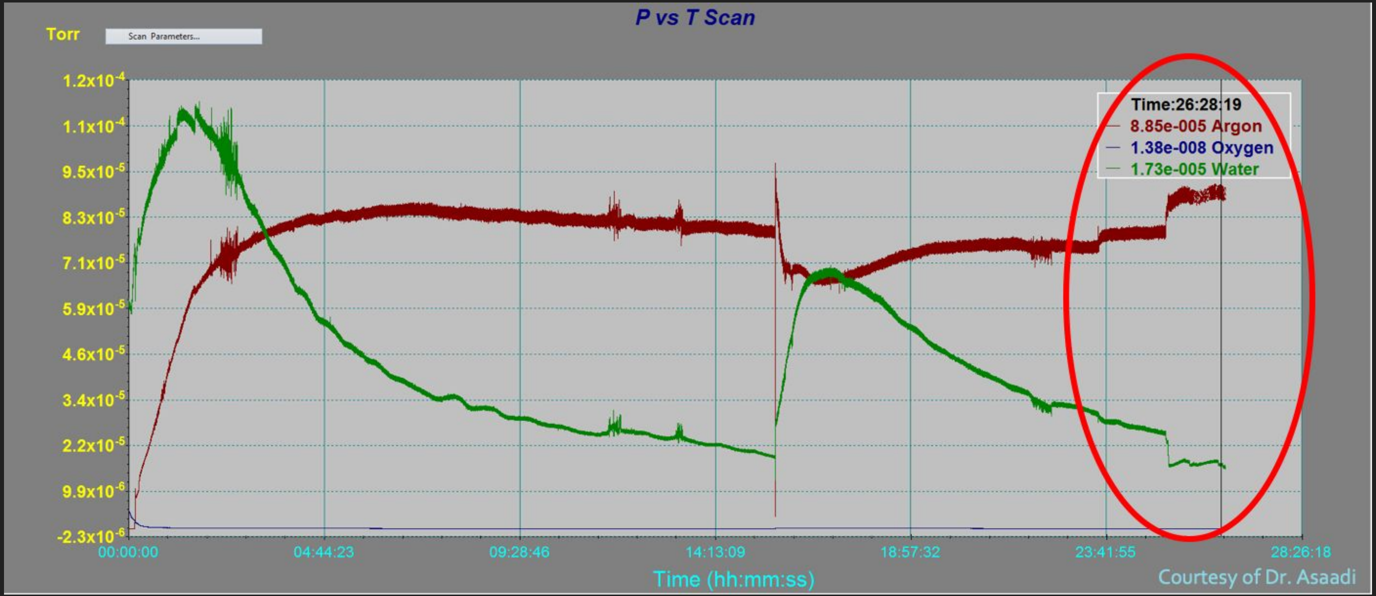
Purification System

Purification System

- **This week completed regenerating the filters**
  - Pumped system down (10^{-5} torr) and ran heater tapes overnight
  - Flowed Ar/H mixture into system for purification (~24 hours)
  - Monitored progress using a bleed in valve into a residual gas analyzer (RGA)
    - RGA used to confirm we've reached an asymptote for the level of water and oxygen in the system



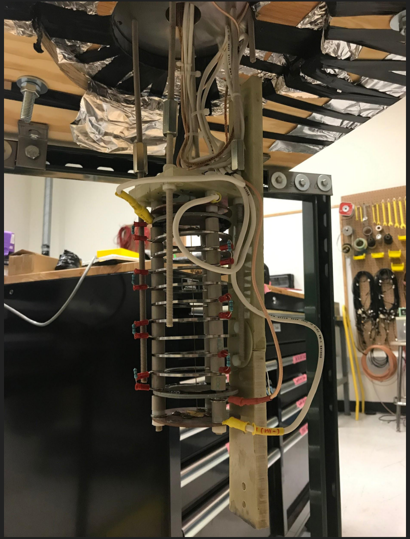
Example of RGA Plot

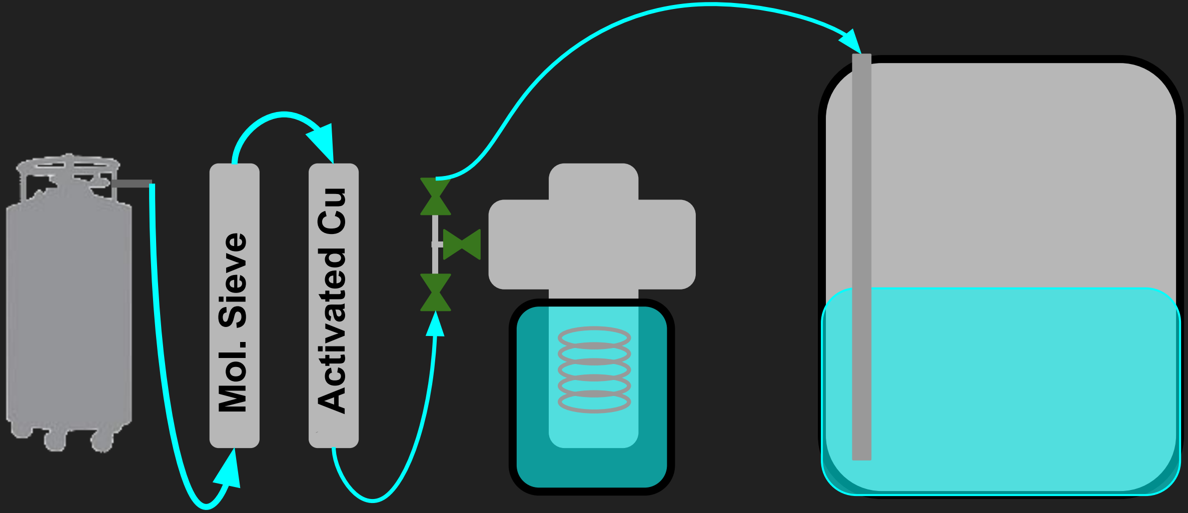


- Next steps:
  - Planning on testing things out with purity monitor sometime within the next week or so
  - Updates to follow

Purification System

- **We are planning on running a 20cm purity monitor in line with our purification system (but external to the larger cryostat) to monitor the argon before we fill**
    - System can be easily valved off and run independently
    - Purity monitor sits in a small argon bath (doesn't take much argon to fill (can used industrial argon for bath)
    - Planning a demonstrator run within 1-2 weeks

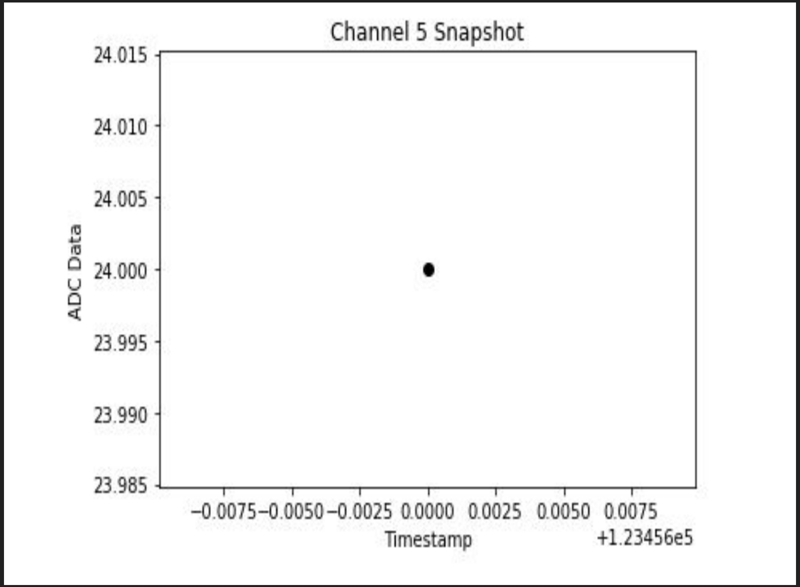




Readout and Data Management System

Data Management System (v0.0) [Prototype]

- **Utilizes larpix-control v2.3.0**
- **Utilizes FakeIO() Interface**
- **A basic structure is defined**
  - Acquire packets
  - Read packets (to/from HDF5 file)
  - Plot packets to screen
- **Utilizes specifically designed data**
  - verifies the file-logging process



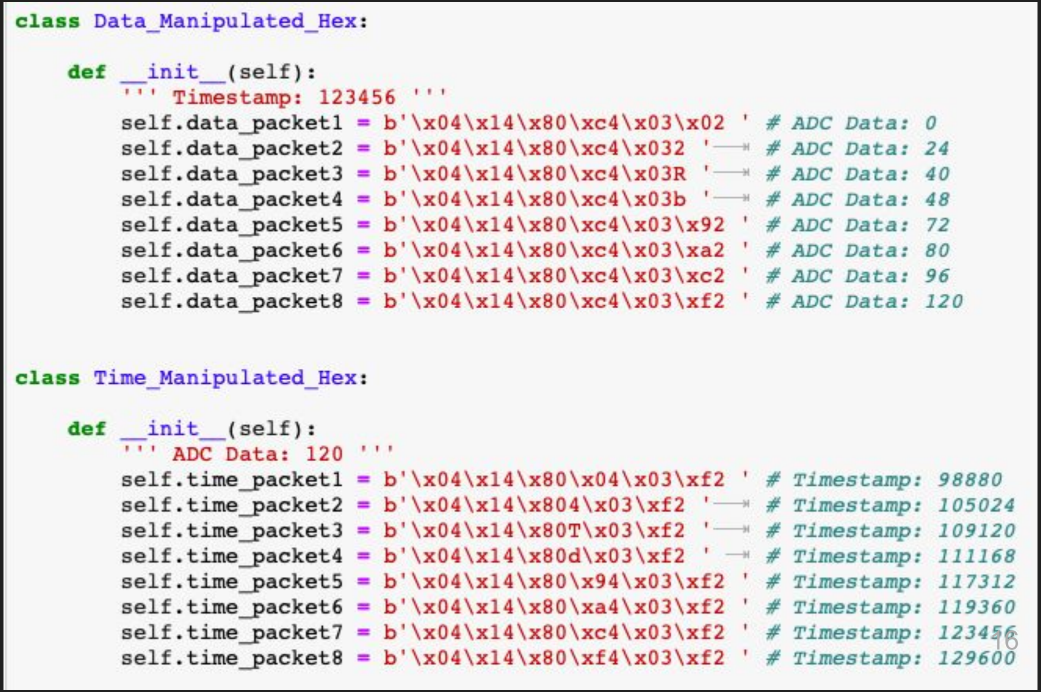
LArPix v1.5 Anode Complications

- Previous troubles in communicating with the board
  - Initiating test pulses with the v1.5 anode has not been accomplished
  - Registers could not be communicated with by the DAQ board

- As a temporary solution, the current readout and Data Management System is utilizing the FakeIO() interface with a simulation-based database that I will pull packets from.
- A more long term solution is understanding how the commands work for the v2 ASIC board in order to begin testing right away

So, where is the data coming from?

- **Packet database to interact with the FakeIO() class exclusively**
- **Each trial will call from class**



```
class Data_Manipulated_Hex:

    def __init__(self):
        ''' Timestamp: 123456 '''
        self.data_packet1 = b'\x04\x14\x80\xc4\x03\x02 ' # ADC Data: 0
        self.data_packet2 = b'\x04\x14\x80\xc4\x032 '    # ADC Data: 24
        self.data_packet3 = b'\x04\x14\x80\xc4\x03R '    # ADC Data: 40
        self.data_packet4 = b'\x04\x14\x80\xc4\x03b '    # ADC Data: 48
        self.data_packet5 = b'\x04\x14\x80\xc4\x03\x92 ' # ADC Data: 72
        self.data_packet6 = b'\x04\x14\x80\xc4\x03\xa2 ' # ADC Data: 80
        self.data_packet7 = b'\x04\x14\x80\xc4\x03\xc2 ' # ADC Data: 96
        self.data_packet8 = b'\x04\x14\x80\xc4\x03\xf2 ' # ADC Data: 120


class Time_Manipulated_Hex:

    def __init__(self):
        ''' ADC Data: 120 '''
        self.time_packet1 = b'\x04\x14\x80\x04\x03\xf2 ' # Timestamp: 98880
        self.time_packet2 = b'\x04\x14\x804\x03\xf2 '    # Timestamp: 105024
        self.time_packet3 = b'\x04\x14\x80T\x03\xf2 '    # Timestamp: 109120
        self.time_packet4 = b'\x04\x14\x80d\x03\xf2 '    # Timestamp: 111168
        self.time_packet5 = b'\x04\x14\x80\x94\x03\xf2 ' # Timestamp: 117312
        self.time_packet6 = b'\x04\x14\x80\xa4\x03\xf2 ' # Timestamp: 119360
        self.time_packet7 = b'\x04\x14\x80\xc4\x03\xf2 ' # Timestamp: 123456
        self.time_packet8 = b'\x04\x14\x80\xf4\x03\xf2 ' # Timestamp: 129600
```

- <u>Disclaimer:</u>

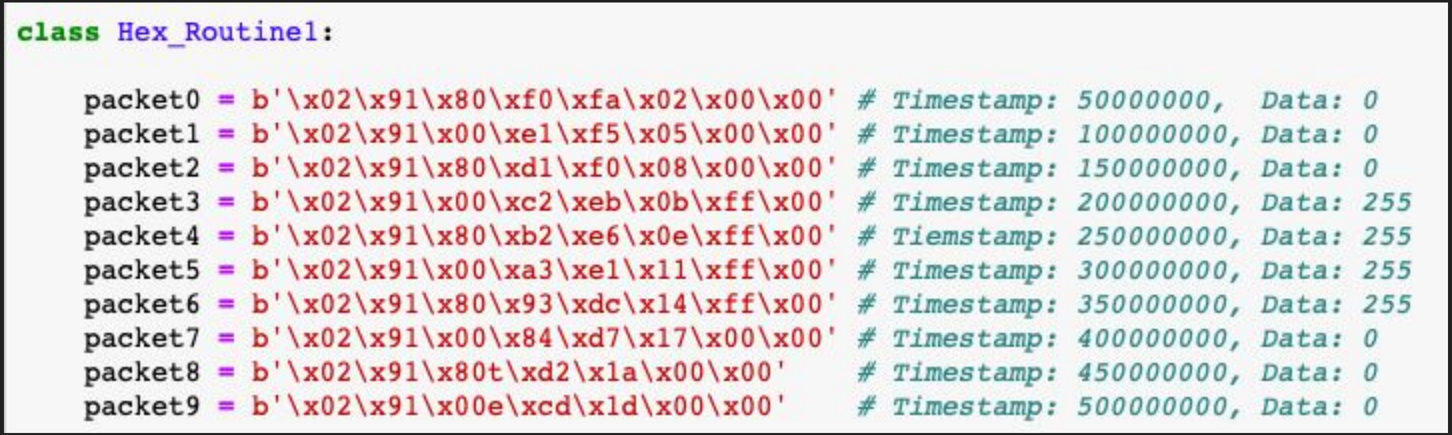
  This is a mere simulation

  for a real reading of ADC

  values

Data Management System (v1.0)

- Identical to v0.0 except written with the updated software (larpix-control v3.1.1)
- Allows for more than one trial to be analyzed at a time
- Demonstrated with Hex_Routine1 class



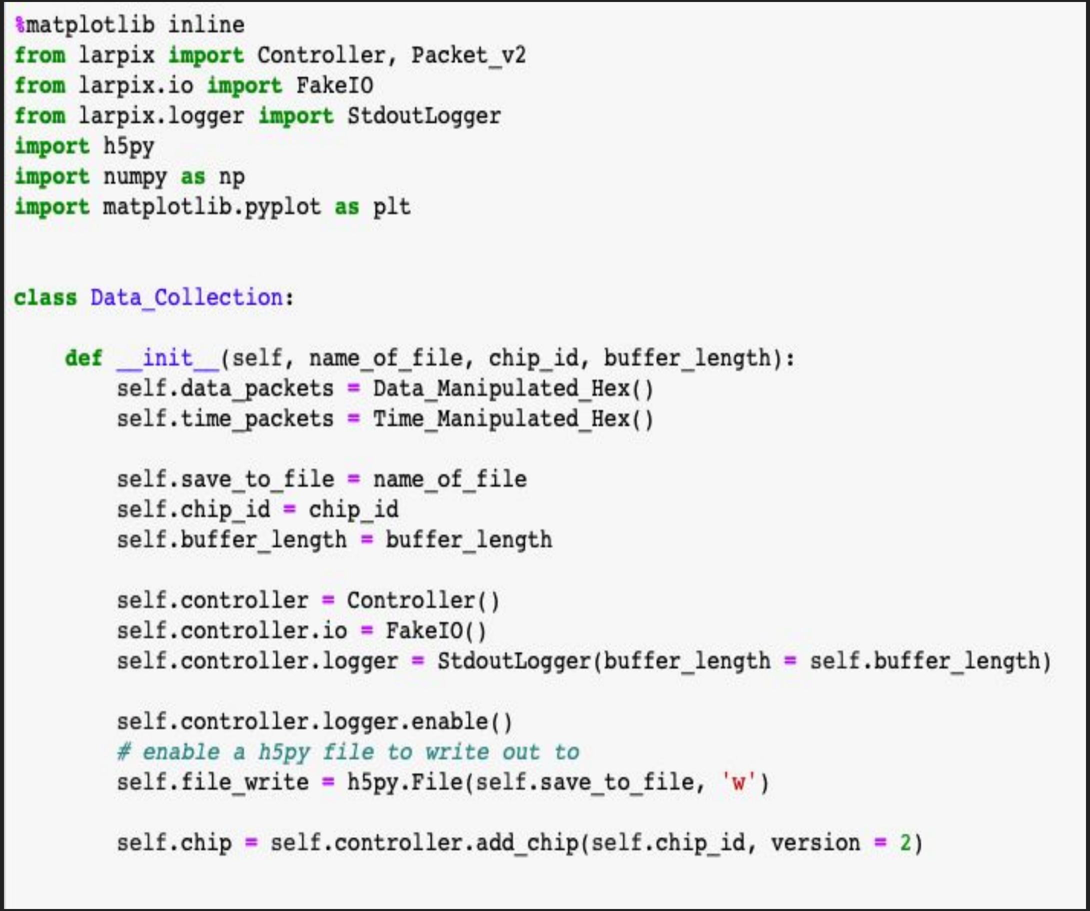
```python
class Hex_Routine1:

    packet0 = b'\x02\x91\x80\xf0\xfa\x02\x00\x00' # Timestamp: 50000000,  Data: 0
    packet1 = b'\x02\x91\x00\xe1\xf5\x05\x00\x00' # Timestamp: 100000000, Data: 0
    packet2 = b'\x02\x91\x80\xd1\xf0\x08\x00\x00' # Timestamp: 150000000, Data: 0
    packet3 = b'\x02\x91\x00\xc2\xeb\x0b\xff\x00' # Timestamp: 200000000, Data: 255
    packet4 = b'\x02\x91\x80\xb2\xe6\x0e\xff\x00' # Tiemstamp: 250000000, Data: 255
    packet5 = b'\x02\x91\x00\xa3\xe1\x11\xff\x00' # Timestamp: 300000000, Data: 255
    packet6 = b'\x02\x91\x80\x93\xdc\x14\xff\x00' # Timestamp: 350000000, Data: 255
    packet7 = b'\x02\x91\x00\x84\xd7\x17\x00\x00' # Timestamp: 400000000, Data: 0
    packet8 = b'\x02\x91\x80t\xd2\x1a\x00\x00'    # Timestamp: 450000000, Data: 0
    packet9 = b'\x02\x91\x00e\xcd\x1d\x00\x00'    # Timestamp: 500000000, Data: 0
```

Initialization Function:

The Controller, chip, and IO interface is defined here.

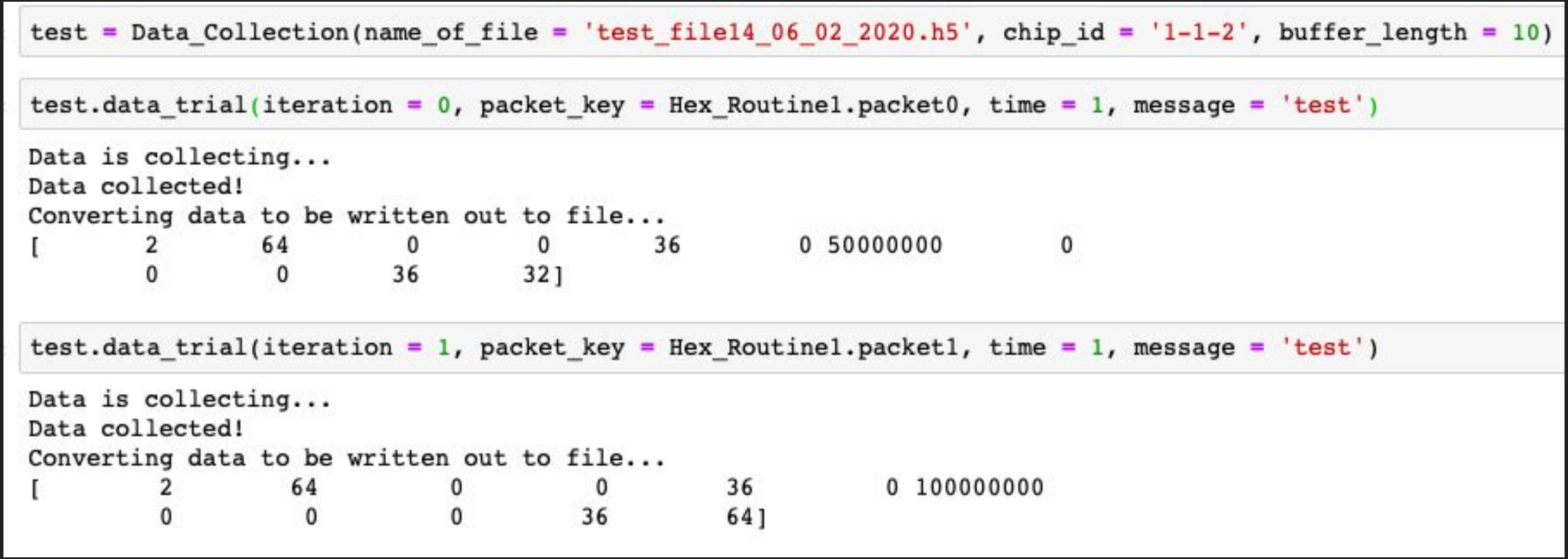
Functions following will specify the acquisition and analysis of data

Future versions will upgrade beyond the FakeIO interface and chips will be defined as groups



```
%matplotlib inline
from larpix import Controller, Packet_v2
from larpix.io import FakeIO
from larpix.logger import StdoutLogger
import h5py
import numpy as np
import matplotlib.pyplot as plt


class Data_Collection:

    def __init__(self, name_of_file, chip_id, buffer_length):
        self.data_packets = Data_Manipulated_Hex()
        self.time_packets = Time_Manipulated_Hex()

        self.save_to_file = name_of_file
        self.chip_id = chip_id
        self.buffer_length = buffer_length

        self.controller = Controller()
        self.controller.io = FakeIO()
        self.controller.logger = StdoutLogger(buffer_length = self.buffer_length)

        self.controller.logger.enable()
        # enable a h5py file to write out to
        self.file_write = h5py.File(self.save_to_file, 'w')

        self.chip = self.controller.add_chip(self.chip_id, version = 2)
```

Data Management System (v1.0) continued...



```
test = Data_Collection(name_of_file = 'test_file14_06_02_2020.h5', chip_id = '1-1-2', buffer_length = 10)
```

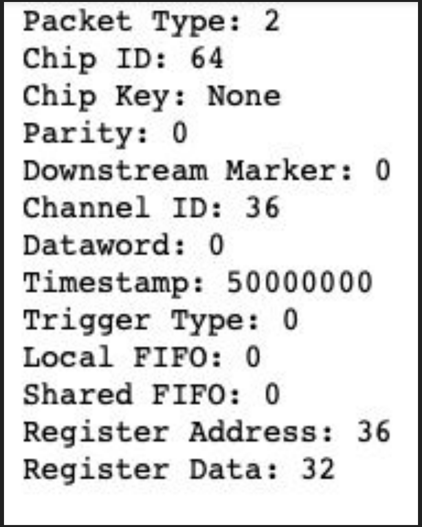
```
test.data_trial(iteration = 0, packet_key = Hex_Routine1.packet0, time = 1, message = 'test')
```

```
Data is collecting...
Data collected!
Converting data to be written out to file...
[        2        64         0         0        36         0 50000000         0
         0         0        36        32]
```

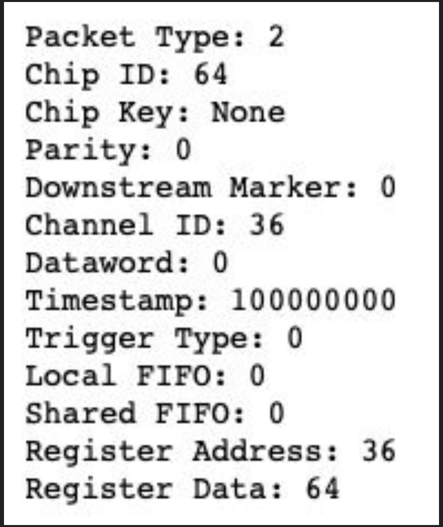
```
test.data_trial(iteration = 1, packet_key = Hex_Routine1.packet1, time = 1, message = 'test')
```

```
Data is collecting...
Data collected!
Converting data to be written out to file...
[         2         64          0          0         36          0 100000000
          0          0          0         36         64]
```

Data Management System (v1.0) continued...

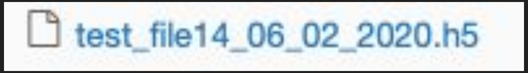


```
Packet Type: 2
Chip ID: 64
Chip Key: None
Parity: 0
Downstream Marker: 0
Channel ID: 36
Dataword: 0
Timestamp: 50000000
Trigger Type: 0
Local FIFO: 0
Shared FIFO: 0
Register Address: 36
Register Data: 32
```



```
Packet Type: 2
Chip ID: 64
Chip Key: None
Parity: 0
Downstream Marker: 0
Channel ID: 36
Dataword: 0
Timestamp: 100000000
Trigger Type: 0
Local FIFO: 0
Shared FIFO: 0
Register Address: 36
Register Data: 64
```

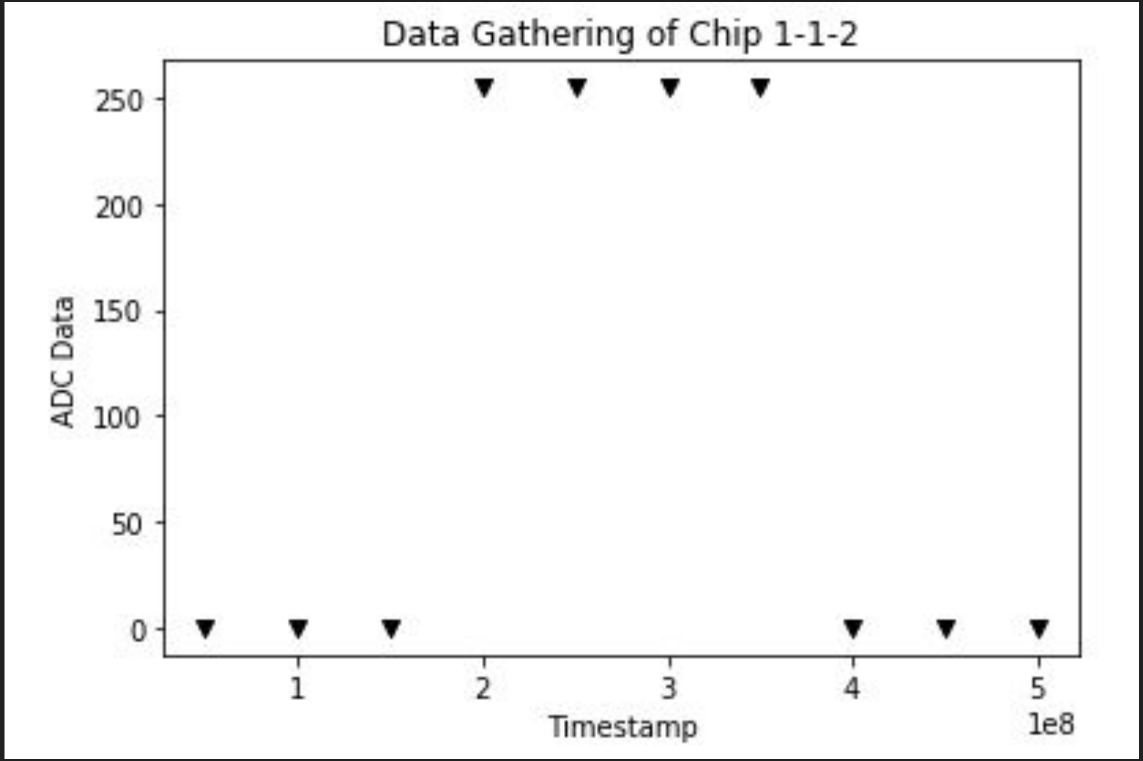
...



test_file14_06_02_2020.h5

Data Management System (v1.0) continued...

- Simulation of a Square Wave
- Data Plotted at this stage of development is for one channel only



Updates to be made and things to consider...

- [The system is designed to be modified easily]
- ZMQ_IO() interface
- Representation of multiple channels and chips at a time
- Hydra Networking
- Plotting Feature (Manual or Automatic?)
- Minor features such as error handling, where files are saved on computer, etc...

Thanks for your attention

*** Link to Github displaying the progress using the software involved ***

https://github.com/jdeleon-py/LArPix-v1-Documentation-JD