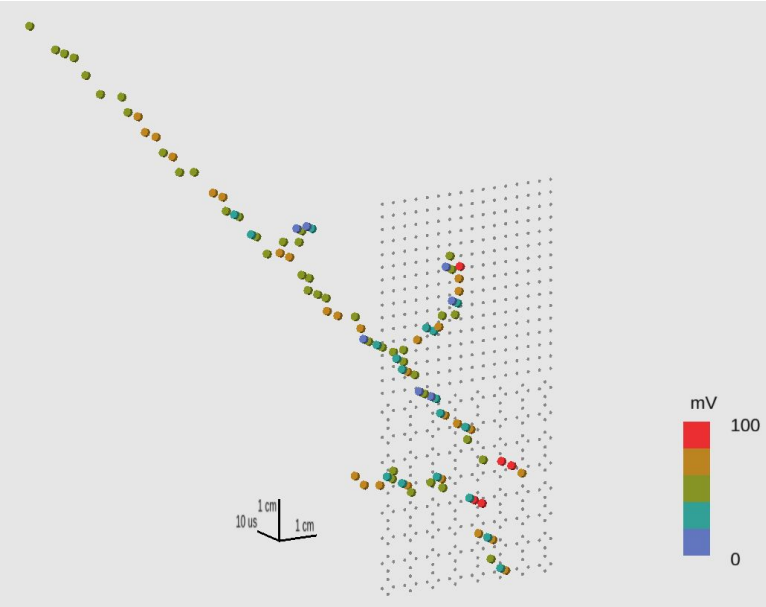
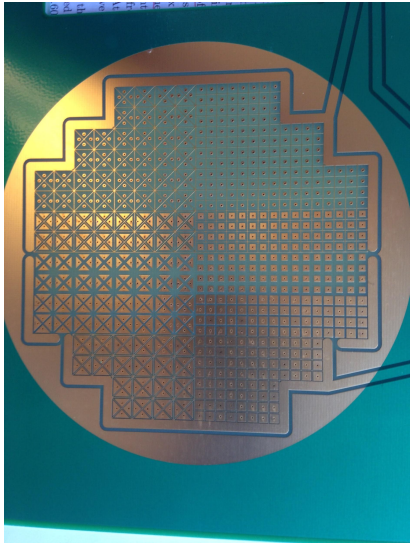
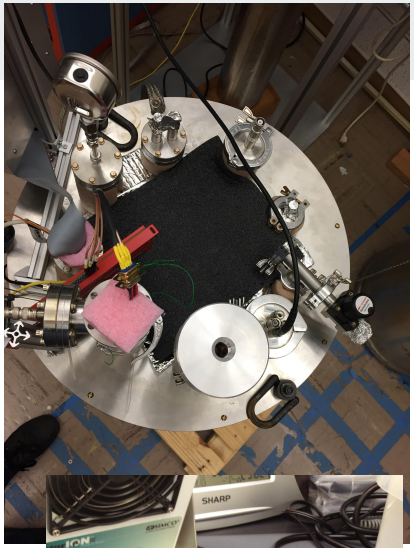
The LArPix Data Acquisition System: Present and Future

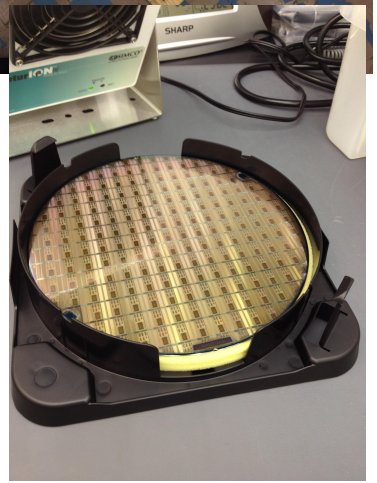
Sam Kohn
UC Berkeley / Lawrence Berkeley National Lab
12 June 2018

LArPix: LAr pixel readout ASIC



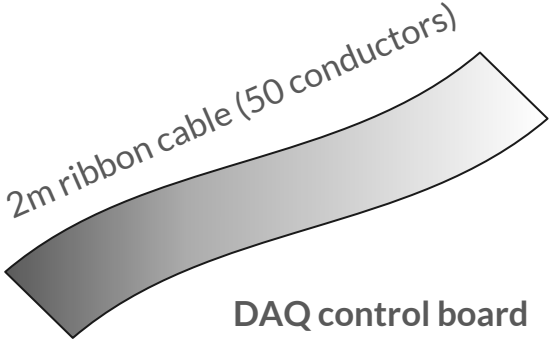


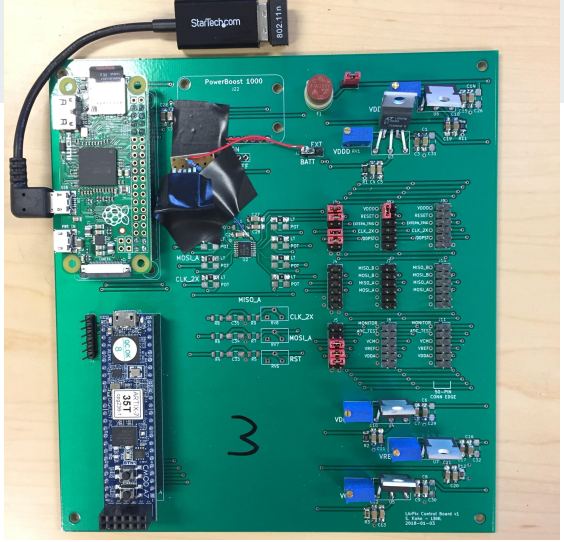




Prototype-scale DAQ

DAQ hardware





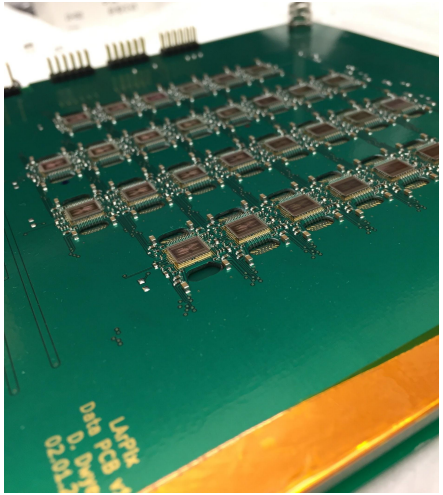
Chip and data board

Digitization

Cold (in LAr)

Signal routing to warm

Designed by D. Dwyer



DAQ control board

Warm (room temperature)

FPGA to convert signal to standard format

Raspberry Pi to interpret signal and store data

Raspberry Pi controlled via SSH over WiFi (crucial for isolating system from laptop/desktop GND)

Designed by S. Kohn

Chip and data board

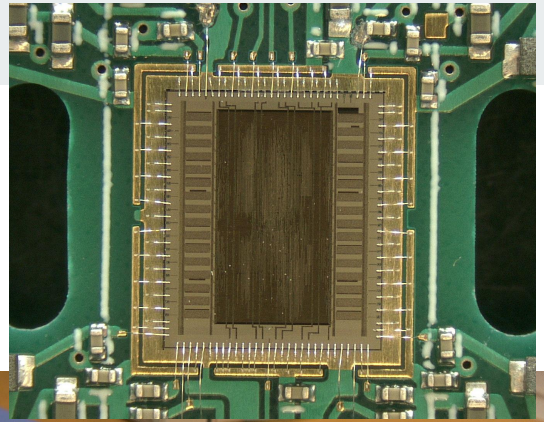
Analog signal arrives through cavity in PCB

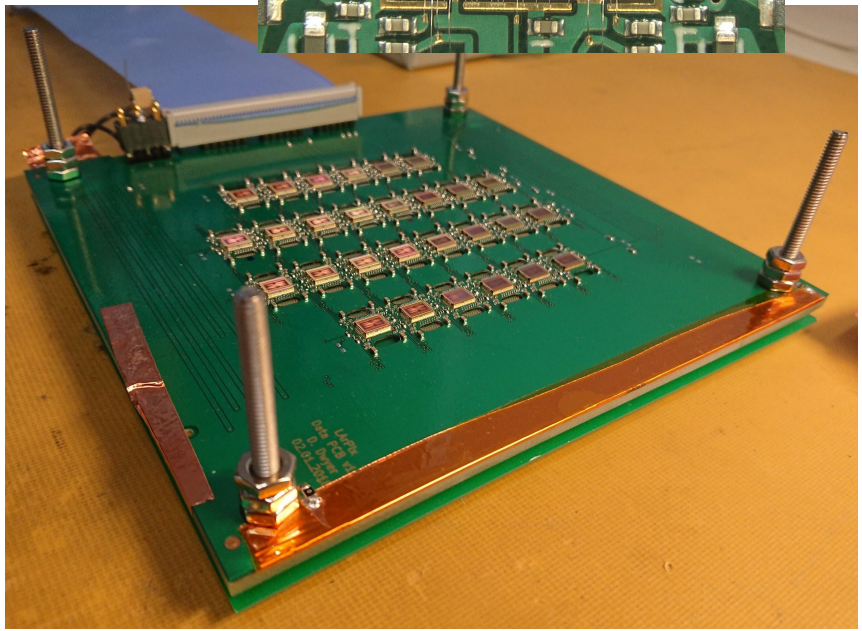
6-bit ADC (i.e. LArPix ASIC)

Digital data IO via daisy chain of 28 chips (256 max)

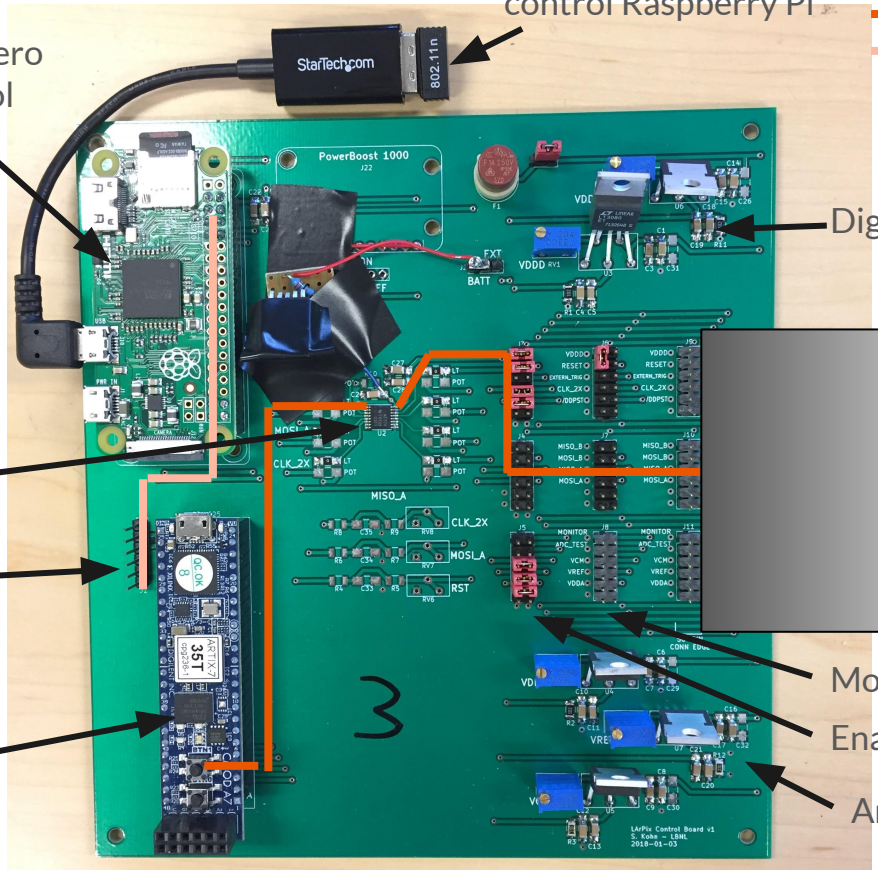
One data wire in, one data wire out of cryostat

Data sent along ribbon cable carrying supply voltages,CLK, & reset lines





DAQ control board



WiFi module for SSH connection to control Raspberry Pi

LArPix UART data path

Serial UART data path

Raspberry Pi Zero onboard control computer

Digital power regulators

Level translator from 3V (FPGA) to current VDDPST

Ribbon cable from cryostat

Serial connection to optional external control computer

Monitor header pins

Enable header pins

FPGA converts from 54-bit packet @ 5 MHz to Serial UART @ 1 MHz

Analog power regulators

More on DAQ control board

Raspberry Pi Zero

- Raspbian Stretch OS version 2017-11-29 (versions are named by release date)
- Runs the LArPix DAQ software
- Saves data to the onboard SD card for retrieval via WiFi or by reading directly off the SD card
- Built-in Tx and Rx pins for communicating via Serial UART

FPGA

- Glorified buffer: LArPix side @ 5 MHz, Raspberry Pi side @ 1 MHz
- Converts format from 54-bit UART to/from 8-bit UART (bi-directional)
- Does not do any data interpretation; just blindly copies bits and bytes

If Raspberry Pi Zero were fast enough, the FPGA would not be necessary

Prototype-scale DAQ

Signal path

Electrons in LAr & pixel pad

LArPix amplifier

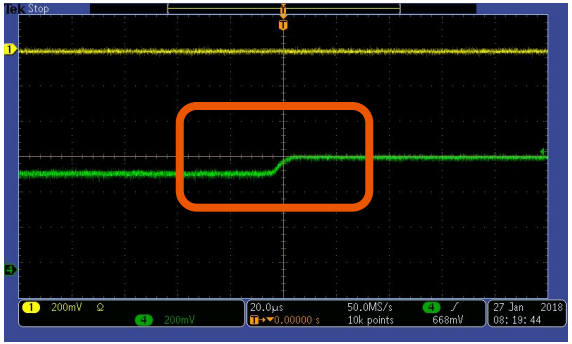
Analog pulse

LArPix ADC & FIFO

LArPix 54-bit data packet (based on UART), 5 MHz clock

FPGA firmware

10-byte serial UART (8N0-type), 1 MHz clock / 1 Mbaud



10 ×

By IngenieroLoco - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53165575

GitHub: https://github.com/samkohn/larpix-control
Docs: http://larpix-control.readthedocs.io

DAQ Software: larpix-control

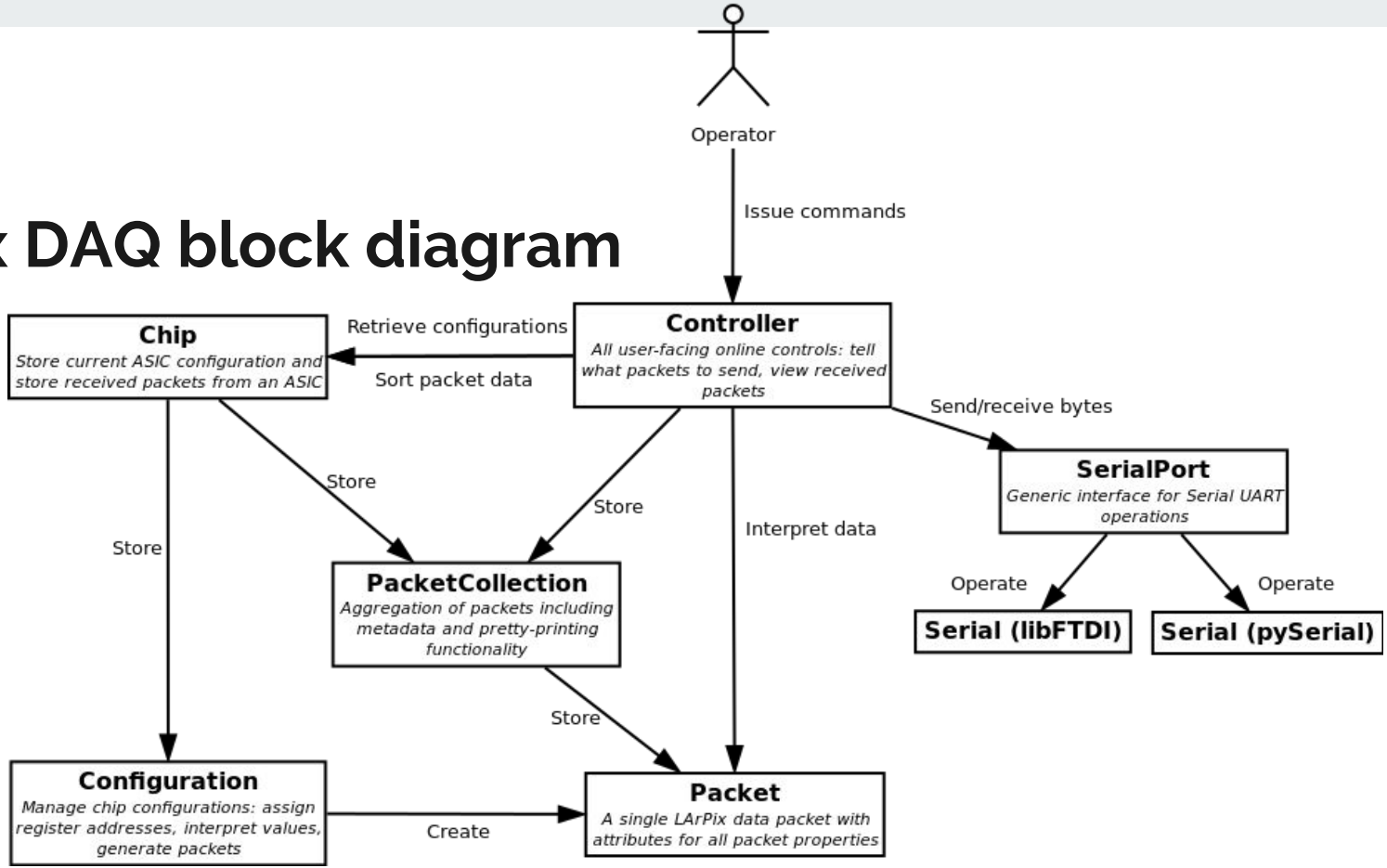
Python library for DAQ and configuration of LArPix ASICs

Completely decoupled from physics software like numpy, ROOT, art, LArSoft, etc.
- Trivial to install on Raspberry Pi Zero (1 optional git clone + 1 pip install)
- Takes up very little disk space (<50MB)
- Saves data in custom binary format (conversion scripts available to ROOT & HDF5)

Interactive & scripted usage supported

Caveat: was not designed to be a full-featured DAQ system. Just enough to get LArPix working!

LArPix DAQ block diagram



Minimal example

```
import larpix.larpix as larpix

controller = larpix.Controller()  # auto-detect serial port address
chip = larpix.Chip(0, 0)  # (chip_id, daisy_chain)
controller.chips.append(chip)  # give the controller control over the chip
chip.config.load('physics.json')  # load physics run mode configuration in software
controller.write_configuration(chip)  # send configuration packets to LArPix ASIC
controller.run(10, 'trial data run')  # listen for packets for 10 seconds
print(controller.reads[0])  # print out the packets from the zero-th data run
```

```
[ Data | Chip: 0 | Channel: 0 | Timestamp: 0 | ADC data: 0 | FIFO Half: False | FIFO Full: False | Parity: 0 (valid: False) ]
[ Data | Chip: 0 | Channel: 0 | Timestamp: 1000 | ADC data: 2 | FIFO Half: False | FIFO Full: False | Parity: 0 (valid: True) ]
[ Data | Chip: 0 | Channel: 0 | Timestamp: 2000 | ADC data: 4 | FIFO Half: False | FIFO Full: False | Parity: 0 (valid: True) ]
[ Data | Chip: 0 | Channel: 0 | Timestamp: 3000 | ADC data: 6 | FIFO Half: False | FIFO Full: False | Parity: 0 (valid: True) ]
```

Developer features: configurations & packets

Knows about chip configuration register names & possible values

```
>>> chip.config.global_threshold = 120  # good
>>> chip.config.global_threshold = 256  # ValueError: global_threshold out of bounds
```

Knows about packet types and packet structure

```
>>> packet.packet_type = Packet.CONFIG_WRITE_PACKET
>>> packet.register_data = 100
[ Config write | Chip: 0 | Register: 0 | Value:  100 | Parity: 0 (valid: False) ]
>>> packet.assign_parity()
[ Config write | Chip: 0 | Register: 0 | Value:  100 | Parity: 1 (valid: True) ]
>>> packet.bytes()
b'\x02\x00\x90\x01\x00\x00 '  # note space (b' ' == b'\x20') includes nonzero parity bit
```

Developer features: data io

Interface for communications implemented for Mac and Linux systems

- Mac uses pylibFTDI and a USB-serial cable
- Linux (laptop) uses built-in driver + pySerial and a USB-serial cable
- Linux (Raspberry Pi) uses pySerial to drive onboard Tx/Rx pins

```
serial = SerialPort(port='/dev/ttyUSB0', baudrate=1000000, timeout=0.1)  # initialize
read_bytes = serial.read(nbytes)  # read maximum of <nbytes> arriving within <timeout>
serial.write(b'\x05\x23')  # write bytes
```

Operator features

Controller object interfaces with the DAQ software base

- Create and configure Chip objects representing individual ASICs
- Read and write configurations from/to ASICs; read data and test packets from ASICs
- Examine received packets

Automatically saves every byte sent and received through the communications interface in a custom data format with extension .dat

Each bytestream is stored along with a log message, timestamp, and some other metadata

Integrated module allows for reading and writing .dat data files, including parsing back into larpix-control data objects to manipulate in Python

Functionality and reliability

Hardware and software described here have been used during every LArPix data run so far

At LBNL

- Demonstrated potential for low-noise operation (low thresholds)
- Noise contribution from DAQ system is tolerable

At Bern

- Demonstrated long(ish)-term reliability
- Uninterrupted operation for 5(ish) days in wide variety of operational modes
- No data lost to crashes or bugs

Upgrade plans

Overview of LArPix DAQ plans

Target: ArgonCube 2x2 Demonstrator

~6m^2 readout area ⇒ almost 10^6 channels ⇒ ~30,000 ASICs ⇒ >100 I/O daisy chains

Sounds complicated? This is what LArPix is designed for!

Will require upgrades to DAQ system to gracefully handle 1000 more ASICs

- Higher data rates
- More than one I/O daisy chain
- More files / more complex files / larger files
- Higher power draw (not technically part of DAQ but current DAQ board also supplies power)

Hardware needs

100 I/O chains is a little more complicated than 1

Routing the additional lines out of the cryostat and onto DAQ control board

Interfacing with the DAQ computer

- Currently using FPGA, but only can accommodate 20 I/O chains
- DAQ computer must be able to handle ~1000x data rate; is Raspberry Pi Zero fast enough?

Ensuring all I/O chains are synchronized: likely will require hardware-level design; software may be enough

Power consumption (again, not technically DAQ but let's put it here): won't be able to run on AA batteries

Software needs

Multi–I/O chain control and data handling

Data synchronization between I/O chains

File format

File creation sequence / data logging / preventing data loss

New I/O interface to support hardware upgrade to the current FPGA functionality

Automate run sequences, configurations and calibrations

Improve user interface to handle more complex routines with more ASICs

May require development of low-level (C or C++) library with Python bindings (only if Python is too slow) (I add this possibility with great reluctance)

Conclusions

LArPix DAQ system works very well for what it was designed for: a small-scale prototype

Upgrades will be necessary to accommodate higher data rates, etc. from a larger apparatus (ArgonCube)

Upgrades seem (mostly) straightforward; no showstoppers here

Thank You!