# Integration Development and Testing of Rear Transition Monitor for Beam Current Monitoring System

Brianna Dewey, SIST – University of Notre Dame

FERMILAB-POSTER-24-0172-STUDENT

## Background

Addressing baseline effects in accelerator environments is crucial for accurate data acquisition and analysis, since baseline effects can obscure signal clarity and impact the reliability of beam current monitoring systems. There are many potential contributors to baseline noise, such as variations in beam dynamics, electromagnetic interference from nearby equipment, or RF interference. Previous applications of noise reduction systems don't sufficiently filter sources of asynchronous noise, so a new algorithm was implemented. A simulation dataset was created to replicate beam conditions and a Red Pitaya FPGA was used to collect data through the streaming application. A Python script was developed to implement noise reduction algorithms and efforts were made to integrate real-time data streaming with the Redis platform and Acnet Front End infrastructure.

## Test Bench and Simulation Data

- Create simulation data that mimics pulsed particle beam current with asynchronous noise
  - Square wave: periodic nature of beam current pulses
  - Sine wave: common low-frequency noise components
  - Resistors: impedance characteristics of BCM and noise source
  - Additional noise data sourced from prior beam measurements
- Collect data via Red Pitaya FPGA - Data Stream Control application

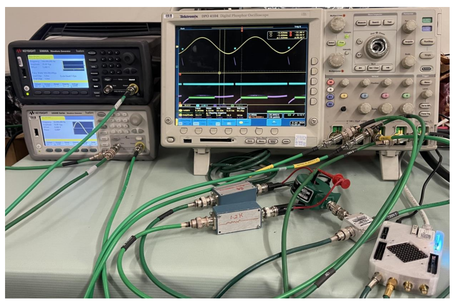
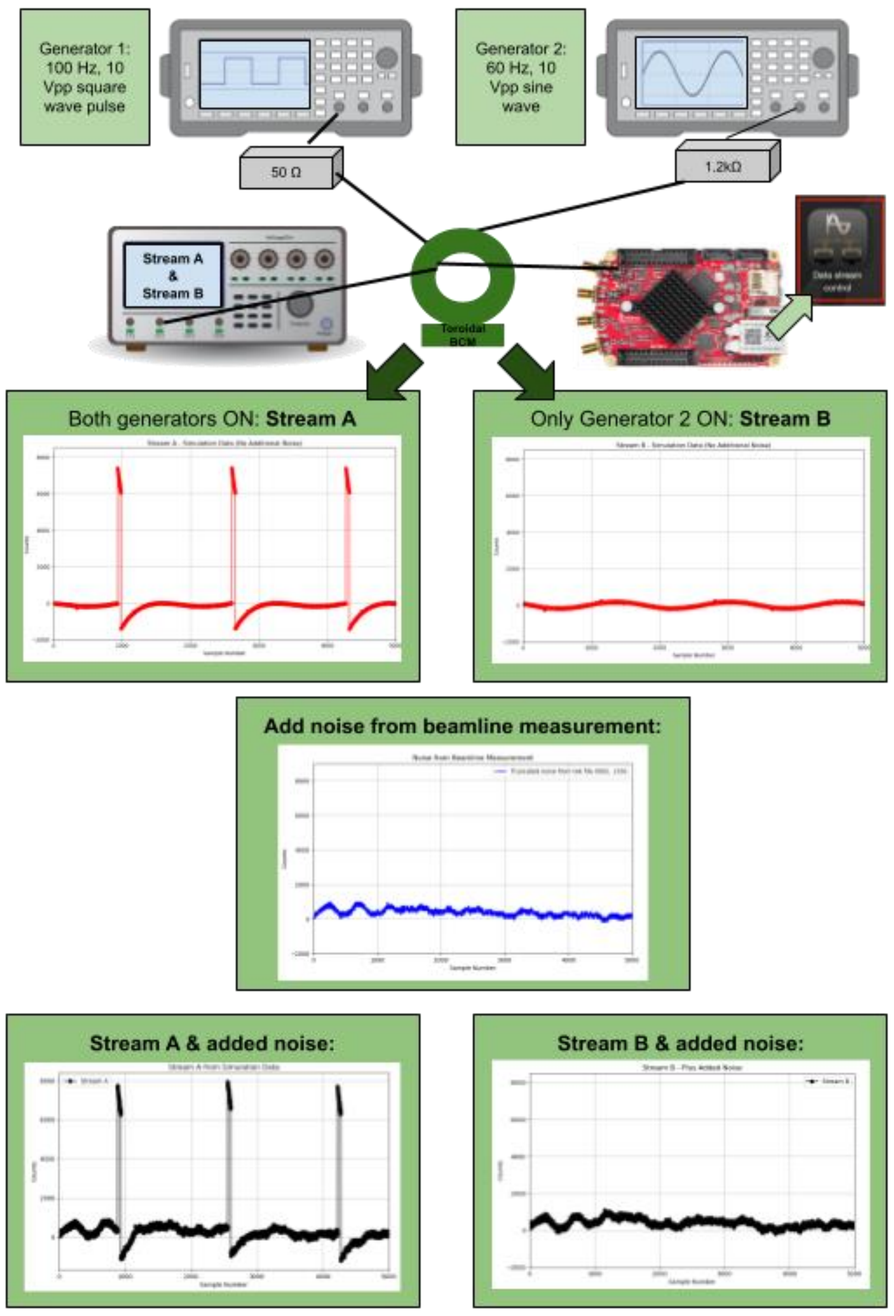


**Image of test bench setup with two signal generators, toroidal BCM, Red Pitaya FPGA, resistors, and oscilloscope.**

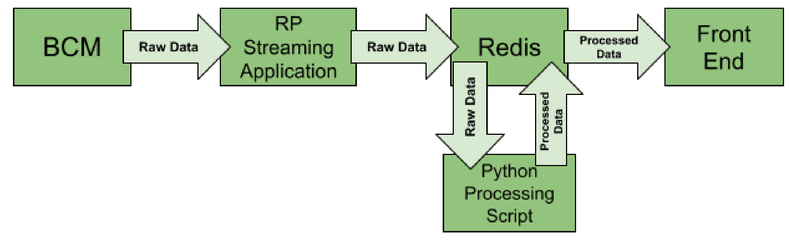



**Model showing the generation of simulation datasets – where Stream A represents beam conditions with baseline effects and Stream B represents only baseline effects.**

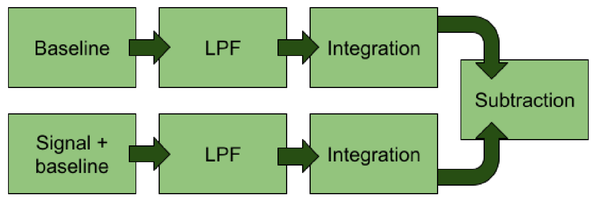
## Data Path

- Prior system processed data on FPGA using microprocessor
  - Limits data collection size due to storage capabilities of FPGA
  - Doesn't address asynchronous noise sources
- Collected data is sent to and accessed via a Redis stream
  - Noise filtering function in Python processes data off-chip and returns beam intensity value to Redis

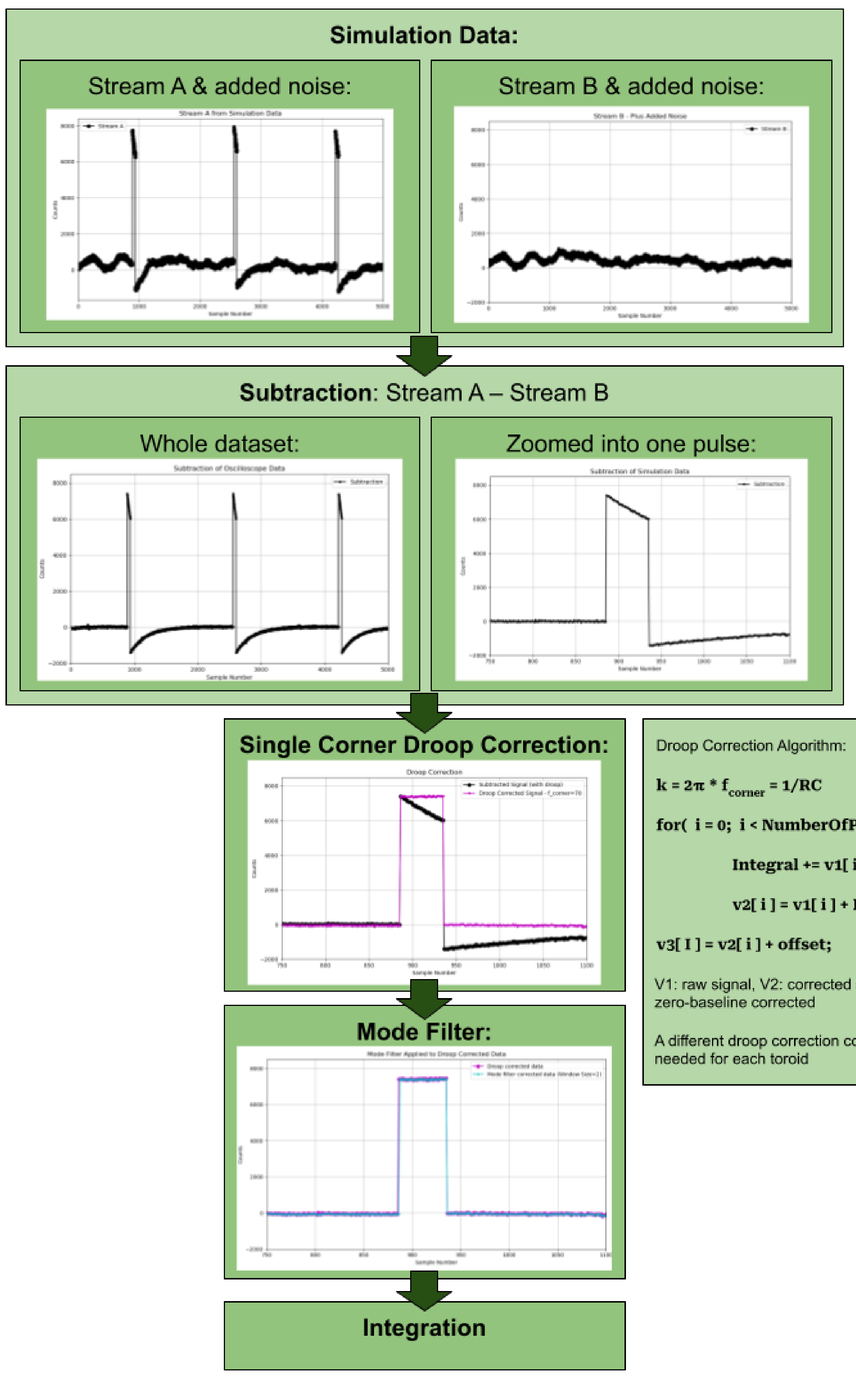
**Block diagram of data stream from BCM to FE.**




## Signal Processing Algorithm




**Block diagram of previous processing system – implemented by the microprocessor on the FPGA.**




**Block diagram of new signal processing function implemented in Python. Plots included to demonstrate the effects of each step of the algorithm.**

Droop Correction Algorithm:

$k = 2\pi * f_{corner} = 1/RC$

```
for( i = 0; i < NumberOfPoints; i++ ) {
        Integral += v1[ i ] * TimeStep;
        v2[ i ] = v1[ i ] + Integral * k; }
v3[ I ] = v2[ i ] + offset;
```

V1: raw signal, V2: corrected signal, V3: zero-baseline corrected

A different droop correction constant (k) will be needed for each toroid

## Future

- Automate process of starting data collection and transferring filename
  - Currently requires manual selection of start button and filename designation
- Install and calibrate system during shutdown
- Take measurements when beamline is active
- Side by side comparison of old vs. new filtering
  - If better (less than 5-10% error pulse-to-pulse), convert other systems in Linac and PIP-II to this setup

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.