U.S. MAGNET DEVELOPMENT PROGRAM

MDP Update 08/31/2022

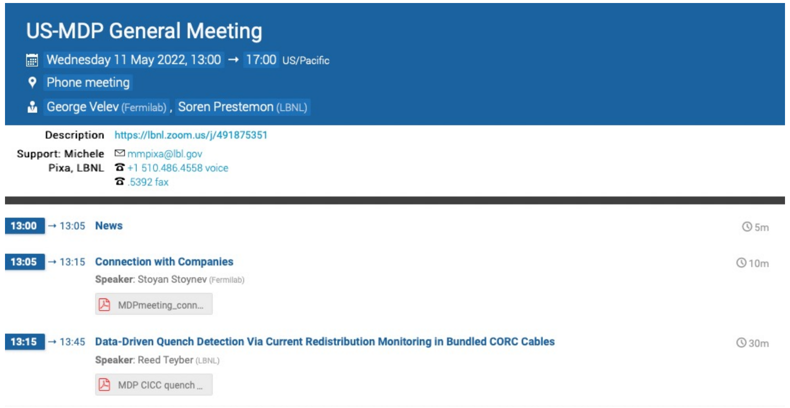
Reed Teyber

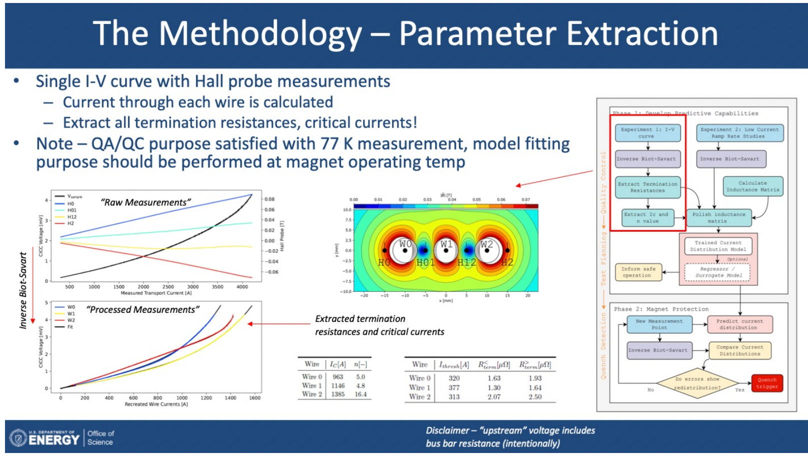
Outline

- Technical Presentation
  - Cable-Of-Cable Current Distribution Monitoring – towards real time hardware implementation
    - *Milestone – "Advance numerical and experimental abilities to monitor and predict current distributions in ReBCO cables for accelerator magnets"*
- Brief update on other milestones
  - Cable scanner for current distribution monitoring
    - *Milestone - "Develop quality control capabilities to identify defects and performance limiting regions in REBCO cables and accelerator magnets"*
  - Quench antennas in Sub 2 reassembly test (T3)
    - *Milestone – "Development of multi-element and flexible quench antennas and localization of quenches in using flexible quench antenna arrays"*

Cable-Of-Cable Current Distribution Monitoring

- MDP general presentation given on May 11
  - In CICC with no current sharing (i.e. CORC 6 around 1), use Hall probe arrays to recreate current distributions in real time
    - Allows electric circuit parameters to be extracted
  - Extracted parameters allow network model to simulate performance in real time – compare recreated (experimental) with simulated values
    - potential to identify conductor degradation, failing joint, or quench in real time
- The methodology is promising, however the underlying processes are time consuming and need to be accelerated
  - Inverse Biot-Savart to recreate cable currents from Hall probes
  - **Electric circuit model (e.g. NGSPICE) to simulate current distribution**
  - Target is order of 1-5 ms (200-1000 Hz)

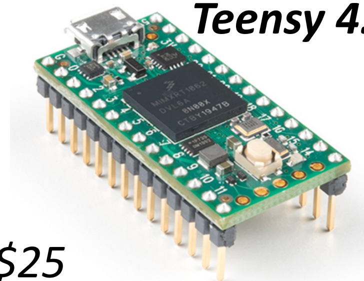




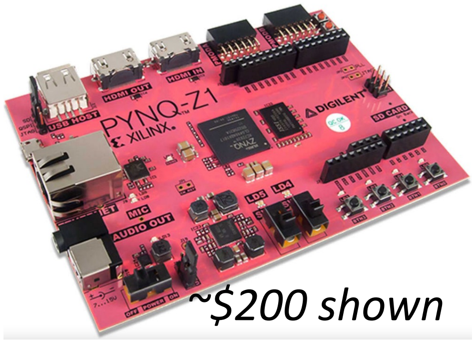
Quench Detection Box

- Three promising hardware approaches being explored, current focus is microcontroller
  - **Microcontroller / MCU – "Teensy"**
    - Teensy 4.0 – 600 Mhz ARM processor (note – Arduino is 16 Mhz) – Focus of Today
    - Hardware treatment of floating point operations
    - No operating software – very fast, but low memory
    - Large community of open source software
  - FPGA (Vivado)
    - Extremely fast, potential to be operated cryogenically
    - Can be FPGA only (cryo) or "System on Chip" (ARM-FPGA) – ZYNQ 7000 series
    - Great for simple logic or common tasks (i.e. facial recognition), but can take a year (+) of full time effort to develop for "non-typical" algorithms like CICC protection framework
  - Embedded system – "fast raspberry pi with big GPU"
    - Nvidia Jetson systems
    - Can communicate with fast ADC's using SPI in C++
    - Slower solution, but best and fastest for implementing more complex algorithms (i.e. high FPS facial recognition)
    - *Can't buy due to chip shortage*

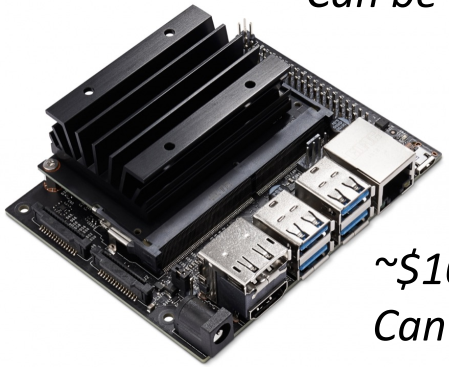
Teensy 4.0



~$25

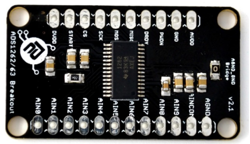


~$200 shown
Can be many $k



~$100 shown
Can be ~$2K

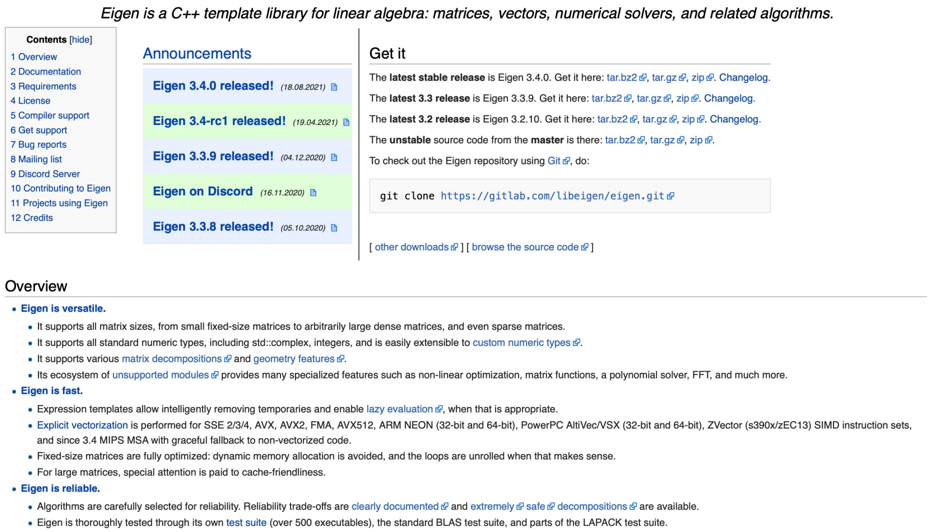
~$50+



All options require external ADC (SPI, I2C…)
Current focus is ADS1262
High resolution, but a bit slow (~ 5 kHz)

Current Recreation

- Previously discussed procedure for processing measured Hall array measurements into wire currents
- Need to solve a matrix system $A^T A x = A^T b$ at every time step where A -> geometry, b -> fields and x -> wire currents
- There is a fork of the “Eigen” C++ linear algebra library that works on ARM microcontroller processors
  - Processes implemented on Teensy and ADS1262!
    - Real time knowledge of current distribution in CICC
  - Ability to maximize computation ahead of time – solution takes ~ than 50 microseconds with 4 sensors and 3 cables


Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

Contents [hide]

Announcements

- Eigen 3.4.0 released! (18.08.2021)
- Eigen 3.4-rc1 released! (19.04.2021)
- Eigen 3.3.9 released! (04.12.2020)
- Eigen on Discord (16.11.2020)
- Eigen 3.3.8 released! (05.10.2020)

Get it

The **latest stable release** is Eigen 3.4.0. Get it here: tar.bz2, tar.gz, zip. Changelog.

The **latest 3.3 release** is Eigen 3.3.9. Get it here: tar.bz2, tar.gz, zip. Changelog.

The **latest 3.2 release** is Eigen 3.2.10. Get it here: tar.bz2, tar.gz, zip. Changelog.

The **unstable** source code from the **master** is there: tar.bz2, tar.gz, zip.

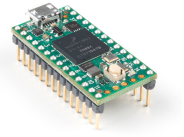
To check out the Eigen repository using Git, do:

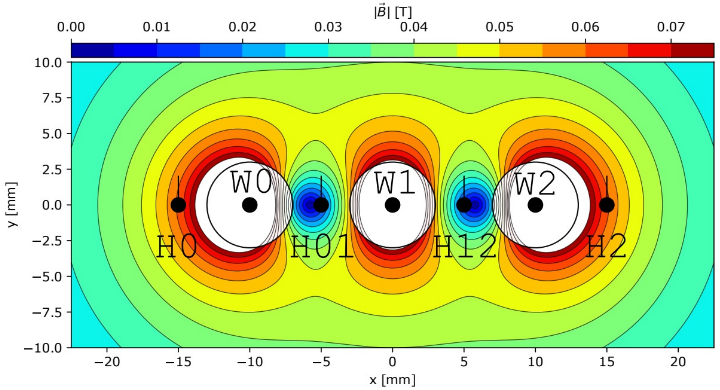
```
git clone https://gitlab.com/libeigen/eigen.git
```

[other downloads] [browse the source code]

Overview

- **Eigen is versatile.**
  - It supports all matrix sizes, from small fixed-size matrices to arbitrarily large dense matrices, and even sparse matrices.
  - It supports all standard numeric types, including std::complex, integers, and is easily extensible to custom numeric types.
  - It supports various matrix decompositions and geometry features.
  - Its ecosystem of unsupported modules provides many specialized features such as non-linear optimization, matrix functions, a polynomial solver, FFT, and much more.
- **Eigen is fast.**
  - Expression templates allow intelligently removing temporaries and enable lazy evaluation, when that is appropriate.
  - Explicit vectorization is performed for SSE 2/3/4, AVX, AVX2, FMA, AVX512, ARM NEON (32-bit and 64-bit), PowerPC AltiVec/VSX (32-bit and 64-bit), ZVector (s390x/zEC13) SIMD instruction sets, and since 3.4 MIPS MSA with graceful fallback to non-vectorized code.
  - Fixed-size matrices are fully optimized: dynamic memory allocation is avoided, and the loops are unrolled when that makes sense.
  - For large matrices, special attention is paid to cache-friendliness.
- **Eigen is reliable.**
  - Algorithms are carefully selected for reliability. Reliability trade-offs are clearly documented and extremely safe decompositions are available.
  - Eigen is thoroughly tested through its own test suite (over 500 executables), the standard BLAS test suite, and parts of the LAPACK test suite.



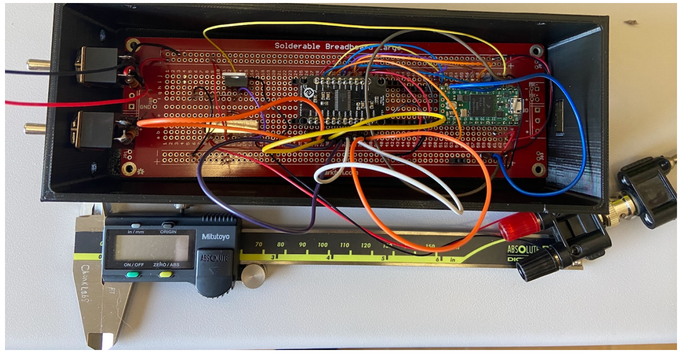


Each CORC Wire

Each Hall Sensor

$$\begin{bmatrix} A^*_{00} & A^*_{i0} & A^*_{n_i 0} \\ A^*_{0k} & A^*_{ik} & A^*_{n_i k} \\ A^*_{0 n_k} & A^*_{i n_k} & A^*_{n_i n_k} \end{bmatrix} \begin{bmatrix} I_{0,z} \\ I_{i,z} \\ I_{n_i,z} \end{bmatrix} = \begin{bmatrix} B_0 \\ B_k \\ B_{n_k} \end{bmatrix}$$

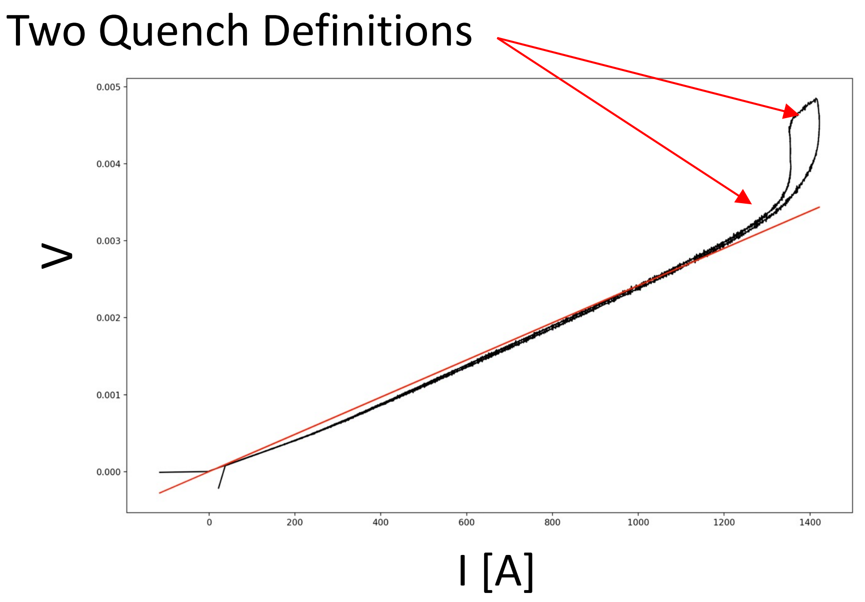
Repeating for each Hall sensor yields matrix — *Currents* — *Field Measurements*

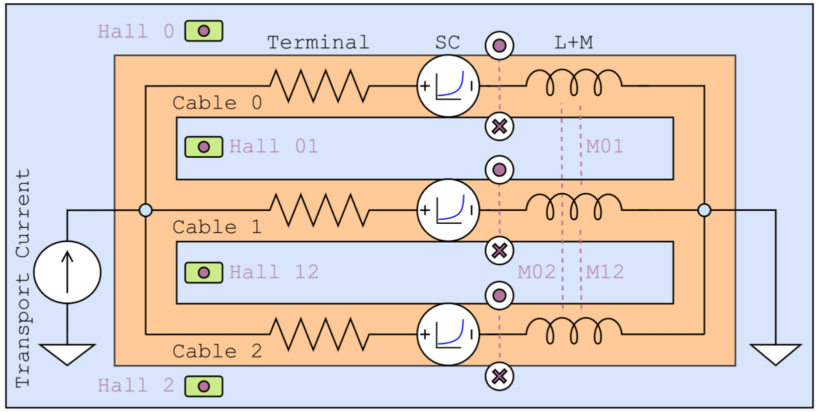


Dynamic Simulation

- Next – how to mimic NGPSICE on a microcontroller?
- Redefining safe operating limit here as "any superconducting voltage exhibited" has favorable mathematical implications - although is a bit conservative
  - High n value superconductor makes computation slow
- Expected current distribution with no superconducting voltage
  - System of ODE's
  - $\Delta V = I_0 R_{T0} + L_{00}\frac{dI_0}{dt} + M_{01}\frac{dI_1}{dt} + M_{02}\frac{dI_2}{dt}$
  - $\Delta V = I_1 R_{T1} + L_{11}\frac{dI_1}{dt} + M_{01}\frac{dI_0}{dt} + M_{12}\frac{dI_2}{dt}$
  - $\Delta V = I_2 R_{T2} + L_{22}\frac{dI_2}{dt} + M_{02}\frac{dI_0}{dt} + M_{12}\frac{dI_1}{dt}$
  - $I_{transport} = I_0 + I_1 + I_2$
- Need to find the solution vector satisfying this
  - Given initial conditions from last time step

$$\vec{x} = \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \Delta V \end{bmatrix}$$





Dynamic Simulation

- Quench detection system needs to solve system of ODE's in real time from one time step to the next
  - Initial condition = previous solution = $\vec{x}_{old}$
- Finite difference of derivative
  - $\frac{dI_0}{dt} = \frac{1}{\Delta t}(I_0 - I_0^{old})$

Original ODE System

$$\Delta V = I_0 R_{T0} + L_{00}\frac{dI_0}{dt} + M_{01}\frac{dI_1}{dt} + M_{02}\frac{dI_2}{dt}$$
$$\Delta V = I_1 R_{T1} + L_{11}\frac{dI_1}{dt} + M_{01}\frac{dI_0}{dt} + M_{12}\frac{dI_2}{dt}$$
$$\Delta V = I_2 R_{T2} + L_{22}\frac{dI_2}{dt} + M_{02}\frac{dI_0}{dt} + M_{12}\frac{dI_1}{dt}$$
$$I_{transport} = I_0 + I_1 + I_2$$

Matrix system for one time step

$$A = \begin{bmatrix} R_{T0} + L_{00}/\Delta t & M_{01}/\Delta t & M_{02}/\Delta t & -1 \\ M_{01}/\Delta t & R_{T1} + L_{11}/\Delta t & M_{12}/\Delta t & -1 \\ M_{02}/\Delta t & M_{12}/\Delta t & R_{T2} + L_{22}/\Delta t & -1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} L_{00}/\Delta t & M_{01}/\Delta t & M_{02}/\Delta t & 0 \\ M_{01}/\Delta t & L_{11}/\Delta t & M_{12}/\Delta t & 0 \\ M_{02}/\Delta t & M_{12}/\Delta t & L_{22}/\Delta t & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} I_0^{old} \\ I_1^{old} \\ I_2^{old} \\ \Delta V^{old} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mathbf{I_{transport}} \end{bmatrix}$$

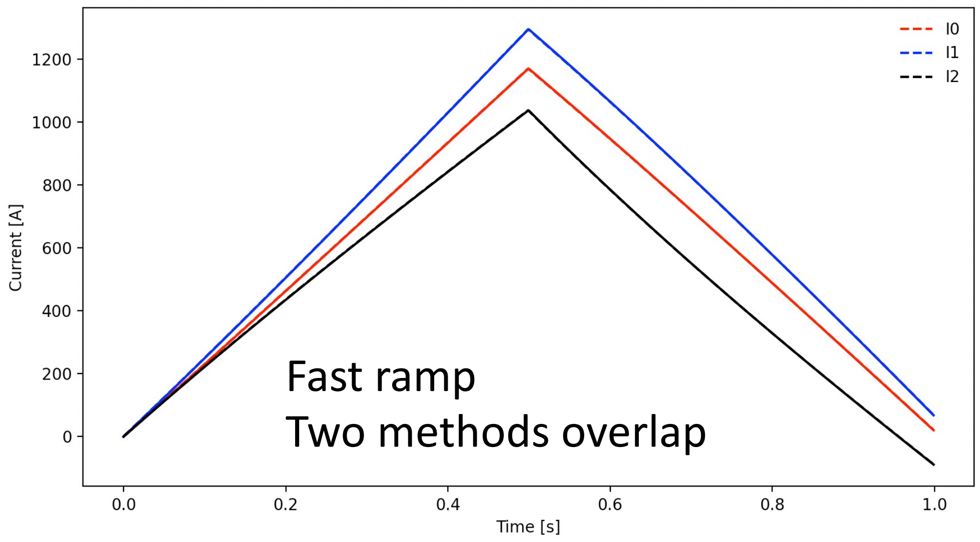
Obtain expected current distribution

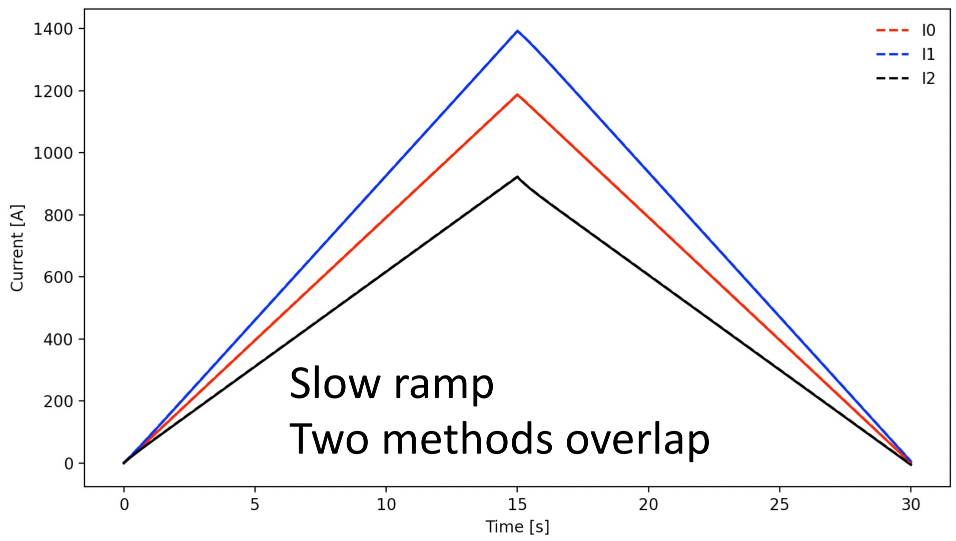
$$\vec{x}_{new} = \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \Delta V \end{bmatrix} = A^{-1}b$$

Validation with NGSPICE

- Prototyped in Python here, validated with NGSPICE simulation
  - With programmed Δt= 1 ms "update", simulations overlap. Fast ramp (left), slow ramp (right)

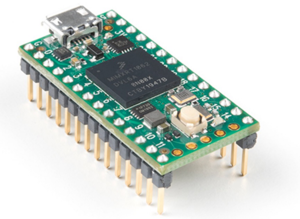
```
Rt_0 = 1.93e-6
Rt_1 = 1.64e-6
Rt_2 = 2.5e-6
R_term_CICC = [Rt_0, Rt_1, Rt_2]


Ic_CICC = [1e9, 1e9, 1e9]
n_CICC = [25, 25, 25]
Vcriteria = 0.0005 #1/2 mV# (1e-6)*length_corc/(100)
branch_elements = 3



L00 = 0.53e-6
L11 = 0.52e-6
L22 = 0.51e-6
M01 = -0.37e-6
M12 = -0.35e-6
M02 = -0.29e-6
```

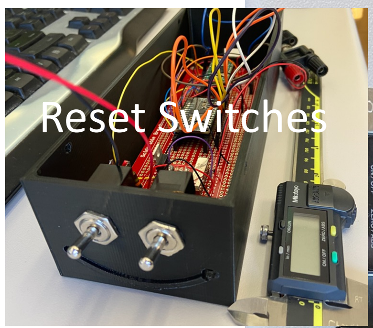


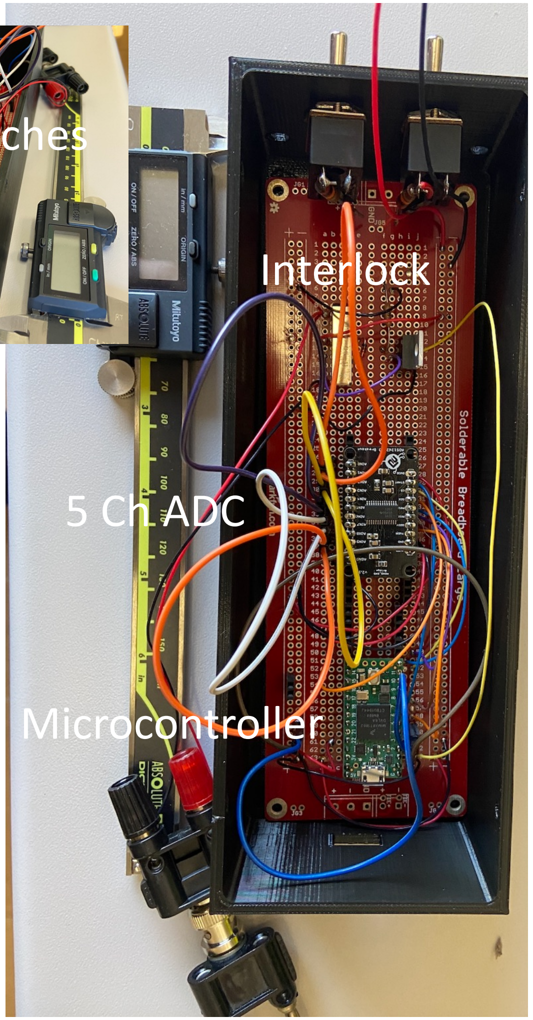


Teensy Implementation

- Programmed, **but need to test on actual sample (Jeremy Weiss)**
  - Get ADC measurements
  - Solve inverse Biot-Savart using LU matrix factorization
  - Solve expected current distribution from history of transport current
  - Look for rate of change of error
    - “large bounds” on absolute value of current error
    - “small bounds” on searching for current redistribution traits
  - Mosfet and Reed relay open interlock for SORENSEN power supply
  - ***Currently takes 2.5 ms to do this, with majority of time spent in ADC!!!***
- After testing the existing prototype, should look into faster and higher channel count ADC’s with acceptable resolution
    - Shouldn’t use delta-sigma conversion framework at high speed…
- This microcontroller is proving very fast and powerful, with many sophisticated open source packages available without too steep of a learning curve
    - Continue development for this application, as well as explore new applications in magnet protection



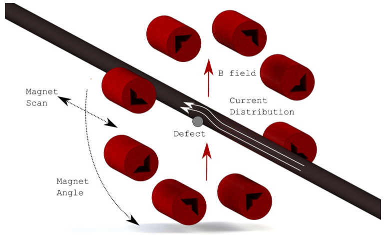


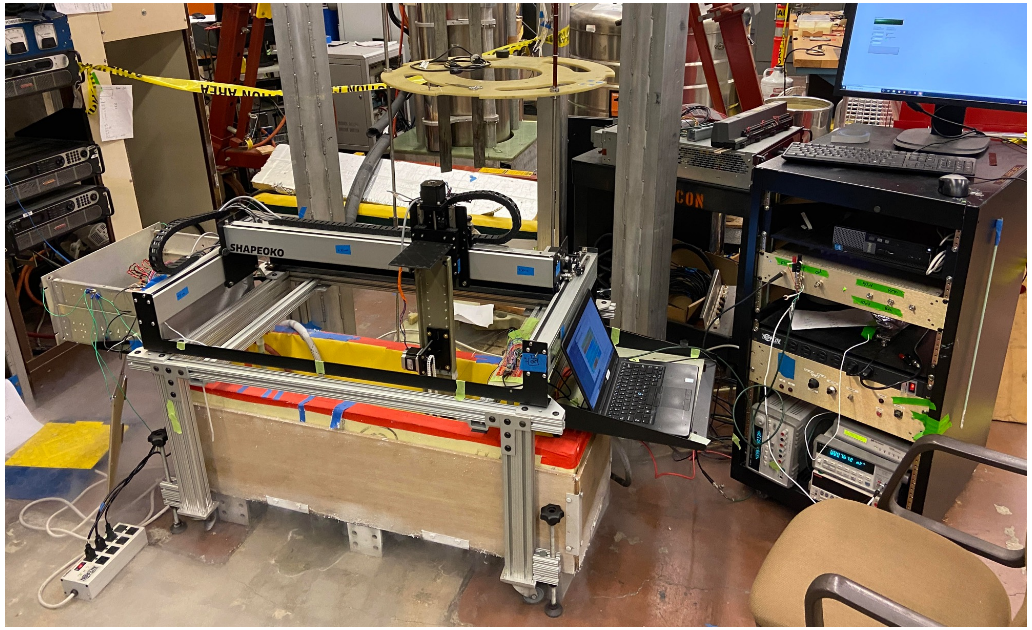


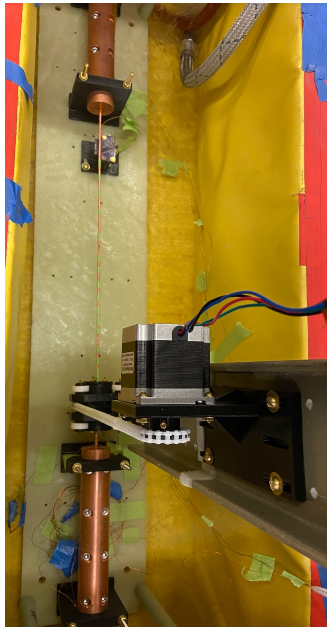
- Previous slides were technical
- Following slides give brief highlights on other progress

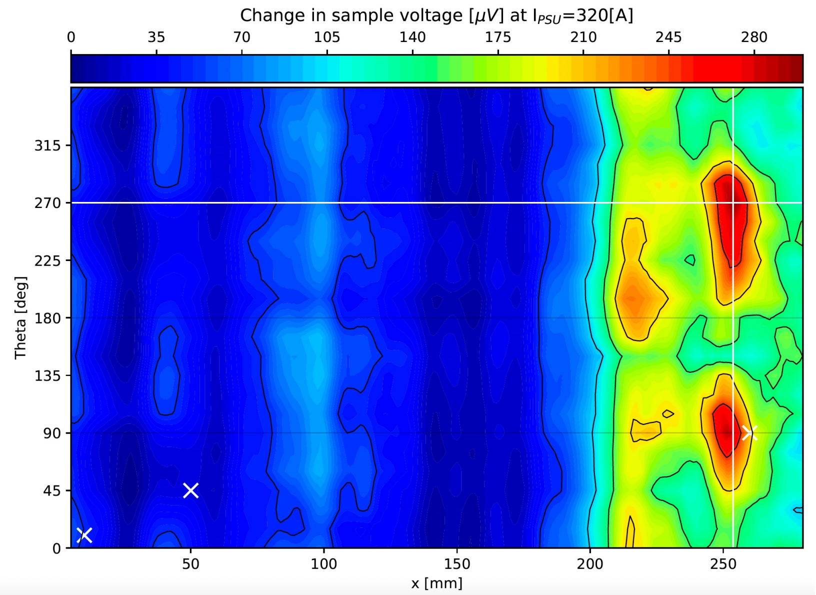
CORC Defect Characterization – Old Results

- Recall these "**old**" results presented at 2022 MDP collaboration meeting
  - Map CORC performance as function of external dipole field angle and position
    - 2,000 automated I-V curves
- Interesting results obtained with potential to identify weak point
  - But need more quantitative results and more insight



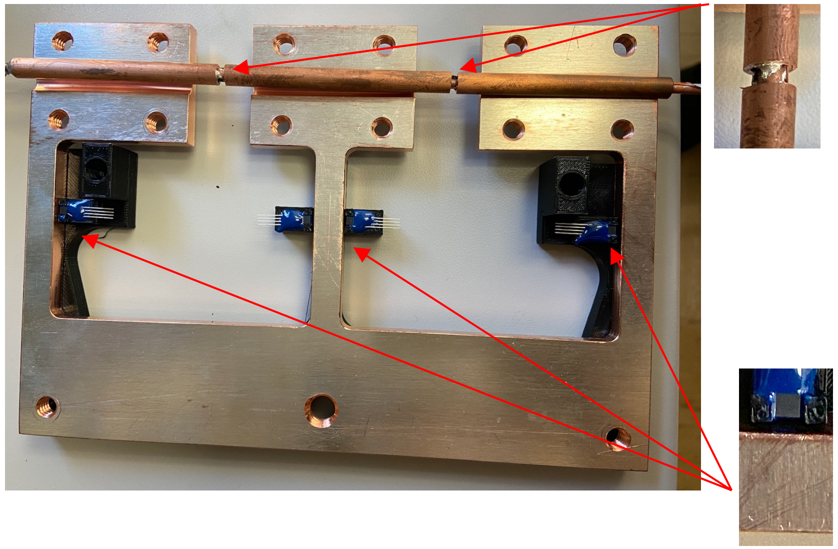


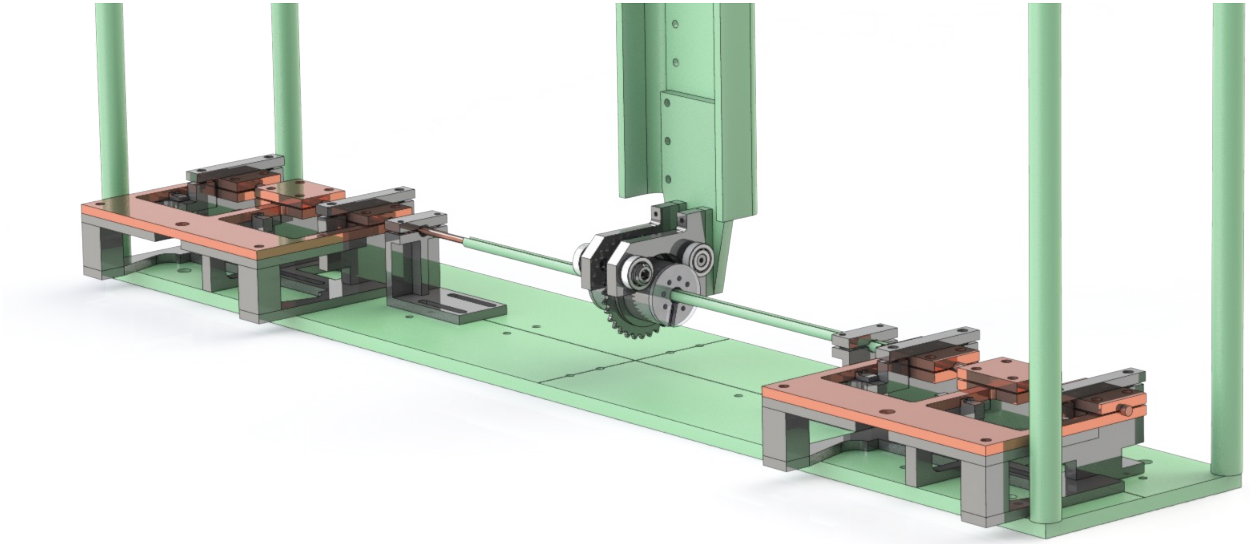




CORC Defect Characterization

- Device modifications since MDP collaboration update
  - New CORC sample with insulated terminals in layer, machined for isolation
  - New terminals with Hall probes to measure current into each partition
  - Map current into each “branch” as function of magnet scan (angle, position)
  - Explore possibility to connect results with model to extract quantitative information regarding spatial distribution of performance
- Still some engineering tasks to address before testing





Quench antennas in Sub 2 reassembly (T3)

- New quench antenna data measured during recent Sub 2 reassembly test
  - Hope to present results soon
- Student (Alberto Plebani), co-supervised with Emanuela Barzi, working on machine learning (clustering) of antenna events along ramp
  - Ultimate goal is to see if there are representative events along ramps that could, possibly, be tied to different physical mechanisms
  - Currently building a feature matrix for each event, hoping to move to K means clustering in coming weeks