NSF Cooperative Agreement EEC-1160504

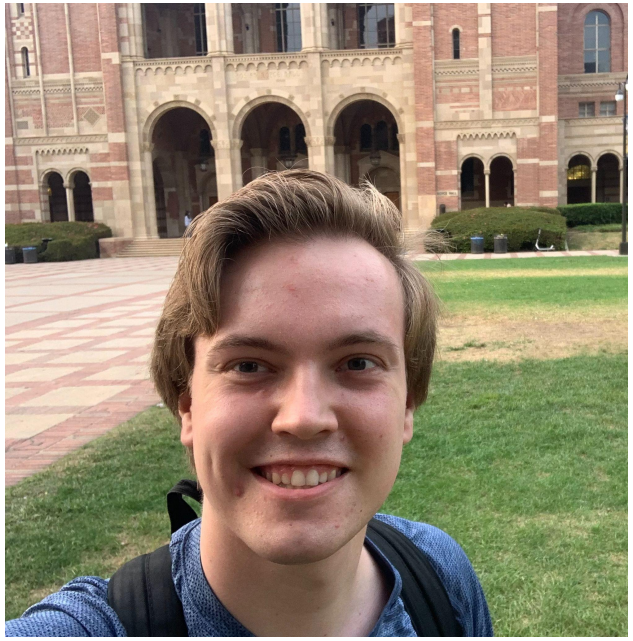
Thermal Camera Imaging for Current Sheet Magnetic Quadrupoles

Nathan Gong, Zach Mustafa
Mentor: Andrew Arrieta
PI: Rob Candler
UCLA

Team Introduction

PI: Dr. Robert Candler

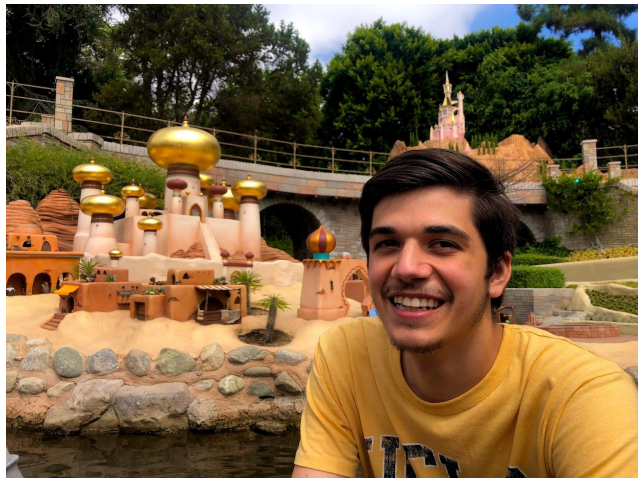
Mentor: Andrew Arrieta



1st Year MS/PhD student, Electrical Engineering

Team Introduction: Undergraduate Members

Zach Mustafa



Year: 5th
Major: Mathematics of Computation

Career Goals: Pursue a PhD in Computer Science, conduct research in Artificial Intelligence

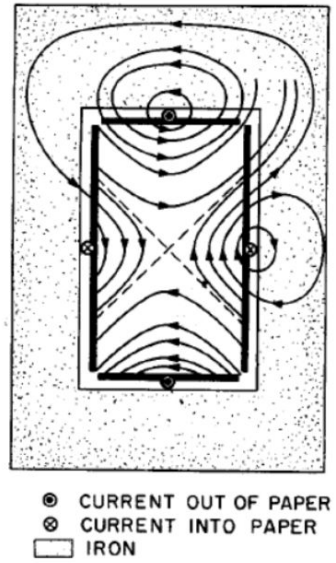
Nathan Gong

Year: 1st
Major: Computer Science and Engineering

Career Goals: Interested in software engineering and data applications in the technology and financial industries

Project Overview: Magnetic Quadrupoles

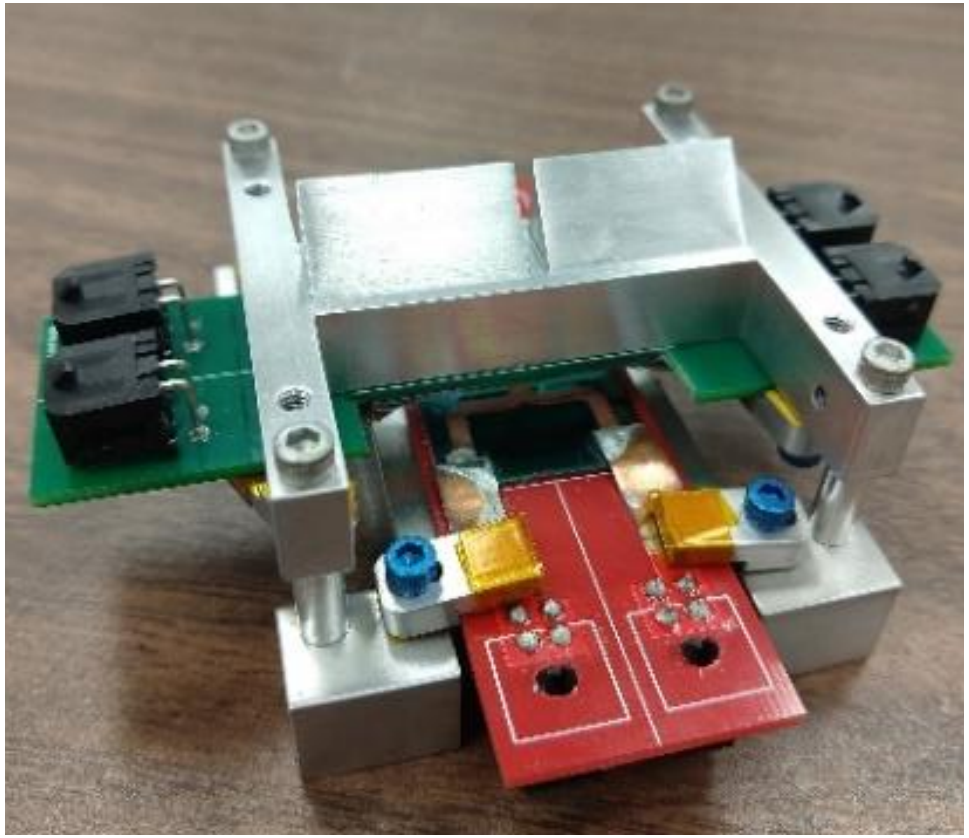
- Our overarching goal is to analyze quadrupoles which use current sheets to generate a magnetic field.

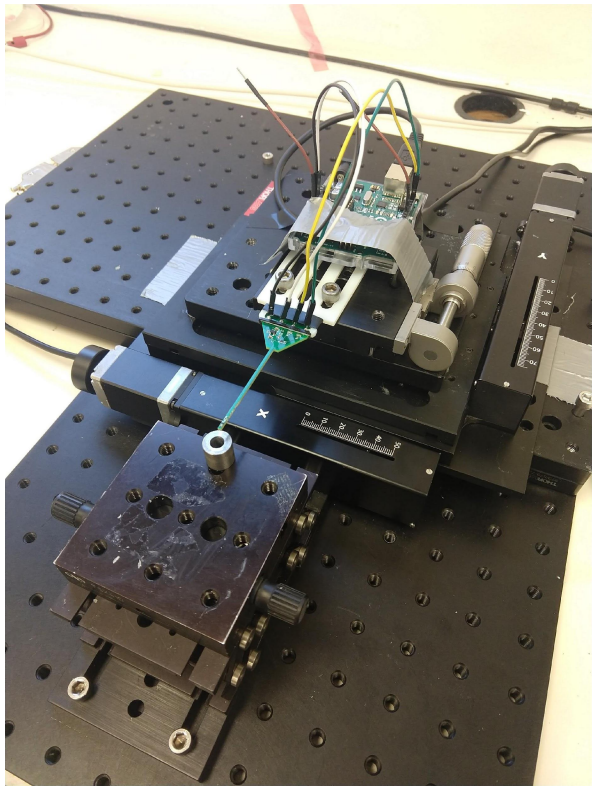


- The magnetic field generated by our devices has a constant gradient
  - **By** (y component of magnetic field) increases linearly with x
  - **Bx** increases linearly with y
- This magnetic field shape focuses an incoming electron beam in one direction, and defocuses the same beam in the other direction.

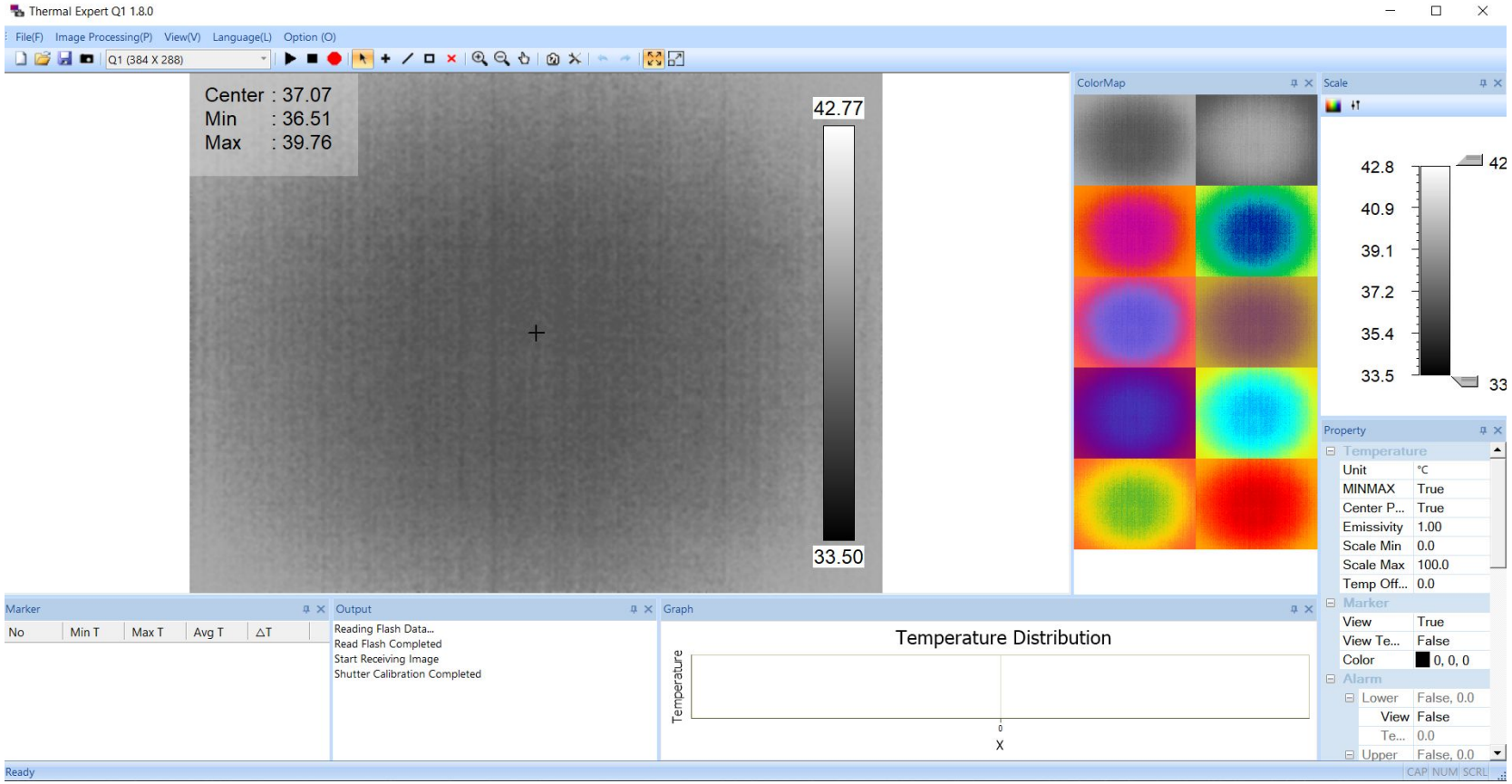
- These current sheet magnetic quadrupoles facilitate research in charged particle beam physics.
- With clever placement of multiple quadrupoles in a row, we can get a net focusing effect on an electron beam

Project Overview: Magnetic Quadrupoles

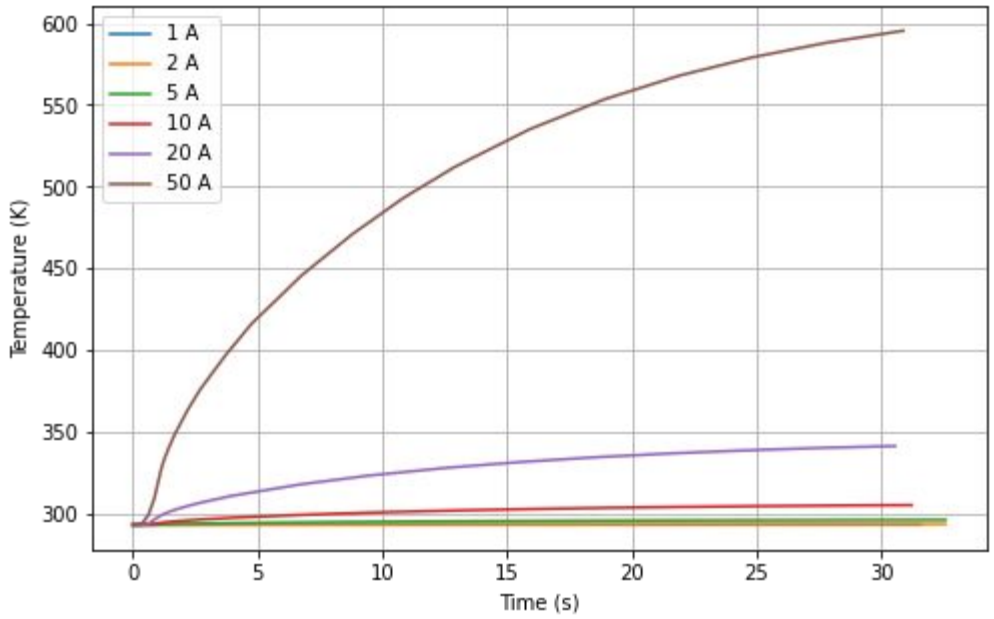


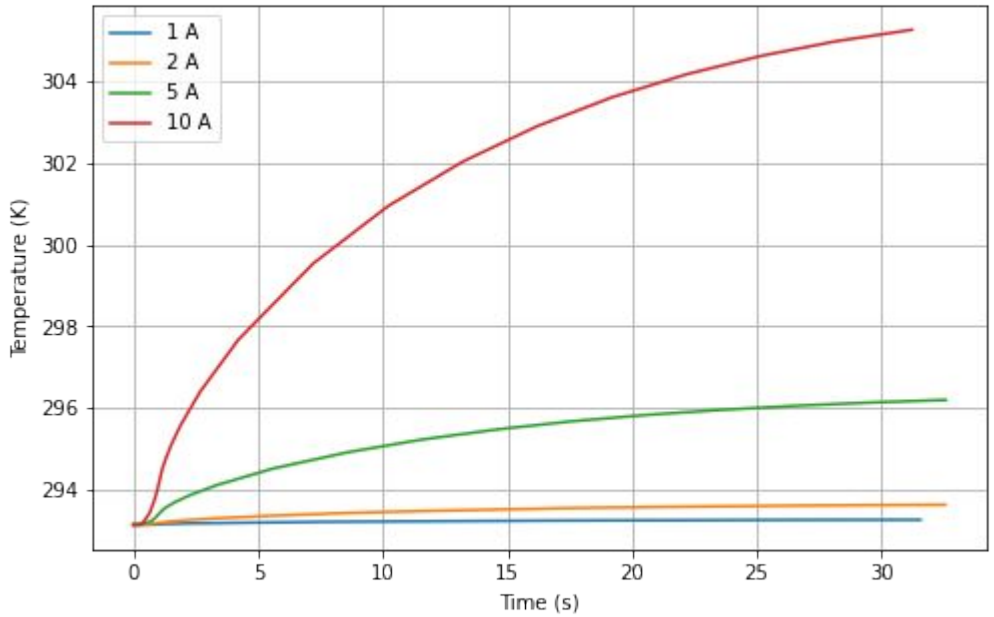


Project Overview: Thermal Camera



Project Overview: COMSOL Sim





Writing Software

- We wanted to write new software to get image data and metadata from our thermal camera.
  - We can then compare this to the COMSOL simulation.
- A Software Development Kit (SDK) was included with the camera, but:
  - Can’t save individual frames– it’s only a live stream
  - Easily crashes
  - Retrieving data is subpar, can’t retrieve after a scan
  - Software bottlenecks
  - Generally incapable of preserving data

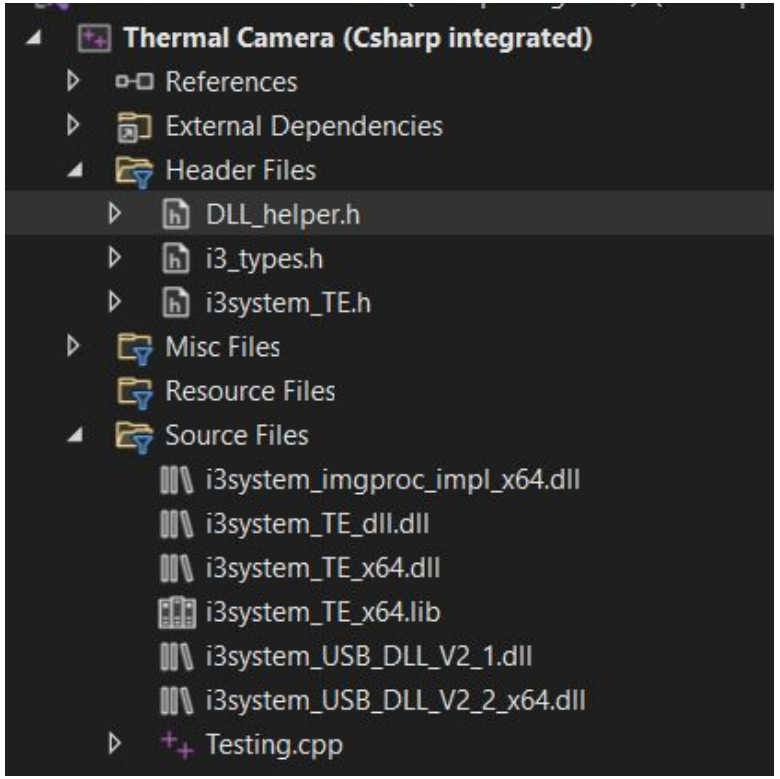
Writing Software: Overview

- We had three main ideas for how to go about writing new software:
  - **Cython**
    - SDK is written in C++
    - Lab uses Python, useful packages for image processing
    - We can use the original header files and .dll files
  - Write a program from scratch using **Python**
    - Less intuitive to interact with the hardware, but simple code
  - Use SDK's lower level files and write high-level code with **C/C++**
    - The program is already written in C/C++ so this is an intuitive option

Writing Software: Overview

- We ended up choosing the third option and using C/C++.

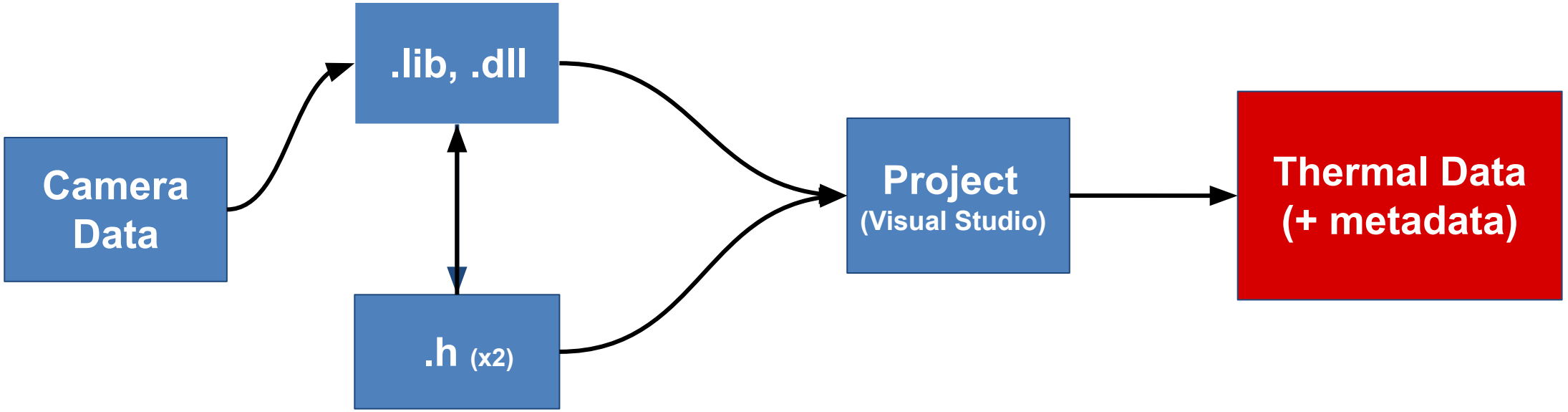
```
namespace PicTaker
{
    class DLLHelper
    {
        static void DLLSetup()
        {
        //Notes:
        // - Some of this stuff might have to be moved outside of a function, because
        // - Check whether we need static
        // - Probably the same functions in the x64 dll, so maybe we can use that
        // - What about the lib file? Might need the non-x64 one if we're not using th


            // Functions from i3system_TE_dll.dll ------------------------------------
            HINSTANCE hinst_mydll = LoadLibraryA("C:\\Users\\zacha\\Documents\\TANMS T

            FARPROC func_ptr [7];
            const char* DLL_FUNCTION_NAMES[7] =
            {
                "RecvImage",
                "RecvImageDouble",
                "CalcTemp",
                "CalcEntireTemp",
                "ReadFlashData",
                "ShutterCalibrationOn",
                "UpdateDead",
            };

            for (int i = 0; i < 7; i++)
            {
                if (hinst_mydll != 0) //Removes the possibility that the function retu
                {
                    func_ptr[i] = GetProcAddress(hinst_mydll, DLL_FUNCTION_NAMES[i]);
                }
            }
```



Writing Software: Process/Challenges

- We now had a framework, but still several logistical challenges
  - Since the SDK is a black box (no implementation files), we had to load the header files and the lower-level (.dll and .lib) files.
  - Lower-level files are cryptic and sensitive to computer architecture, not readable
  - We had to switch back and forth between IDEs due to limitations
- There were several times that we tried switching to a different codebase
  - (C++ → Python → Cython, etc)

Writing Software: Process



Results + Future Work

- We explored all our options and eventually had the makings of improved software
- Logistical challenges + switching codebases led to endeavor being unfinished

Results + Future Work

- We realized that working in Cython is the best option moving forward.
- Python gives us ease and flexibility to process our data
  - Simpler, more modern solutions
  - Robust image processing software
  - Easier communication with USB devices
- We could use a minimal skeleton of the original program → C/C++
  - Neither header files (.h) nor lower-level files (.dll/.lib) would be compatible in pure Python

TANMS Takeaways

- Learning Communities are a fun and engaging way to learn and to interact with other members of TANMS.
- Spending time in a research environment and in the lab has been great experience for both of us.
- Gained experience in reading literature and collaborating as part of a research team.
- The Diversity Module allows us to learn about persistent socioeconomic problems in STEM that need to be addressed.
- We prioritized the research process and scientific learning over the result of the project itself, sparking future interest in undergrad/graduate research in interdisciplinary projects.

Acknowledgements

Andrew Arrieta, Dr. Rob Candler, and the Sensors Technology Lab team

Dr. Pilar O' Cadiz and Dr. Tsai Tsai O-Lee

TANMS Organization and Learning Community teams

National Science Foundation