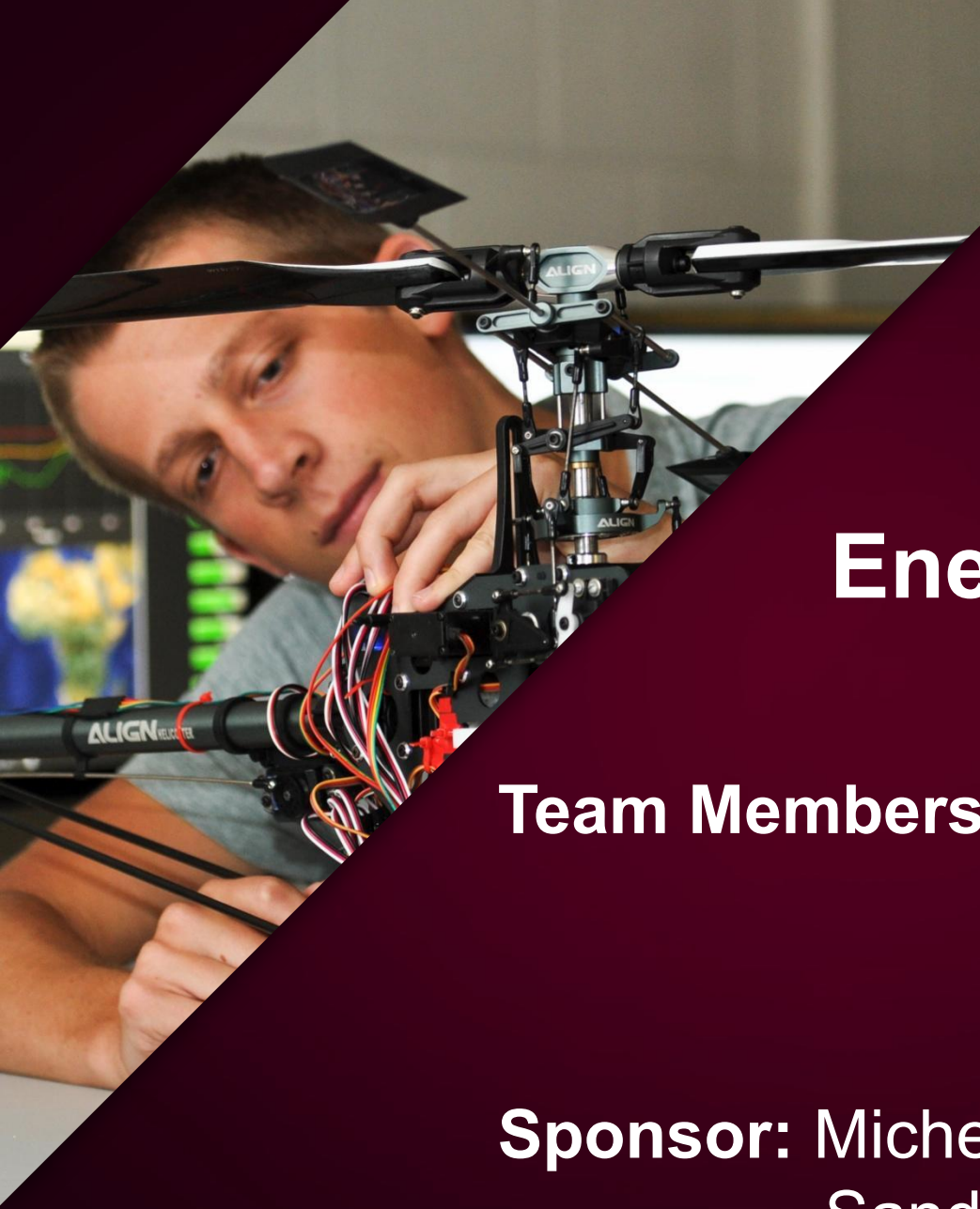


Dwight Look College of ENGINEERING
TEXAS A&M UNIVERSITY

Portable High Energy Experiment (PHEE) DAQ

Team Members: Ethan Barnes
John Sabra
Sang Hoon Chung

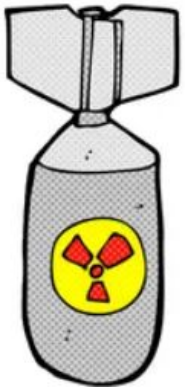
Sponsor: Michelle Chatter
Sandia National Laboratories

TA: Max Lesser

Executive Summary

Problem statement:

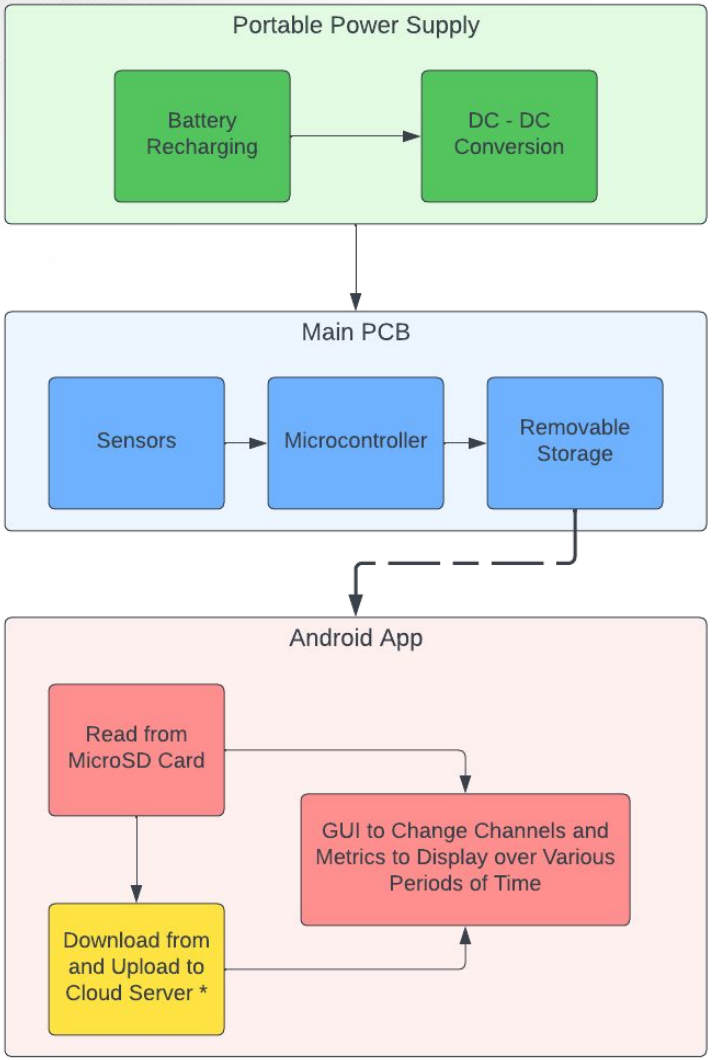
- The United States military possesses about 5500 nuclear weapons in its stockpile
- The security of these weapons and equipment is paramount when they are transported for storage and testing
- Sandia is interested in detecting explosive-type events in sensitive areas
  - Exact application for DAQ system may not be disclosed



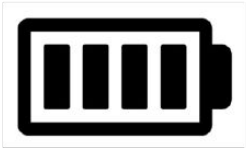
The Portable High Energy DAQ System will:

- Protect government equipment by detecting and identifying explosives within a 100 ft range
  - Utilizes microphone, accelerometer, and pressure sensor to classify if an explosive event occurred
- Write output to removable storage device
  - User will be able to refer to past outputs

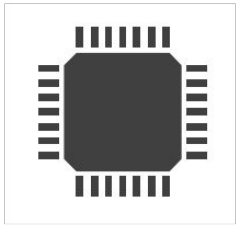
System Overview



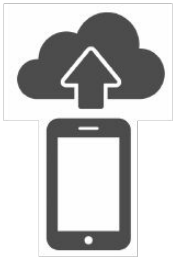
Johnny Sabra



Ethan Barnes



Sang Hoon Chung



Johnny Sabra *

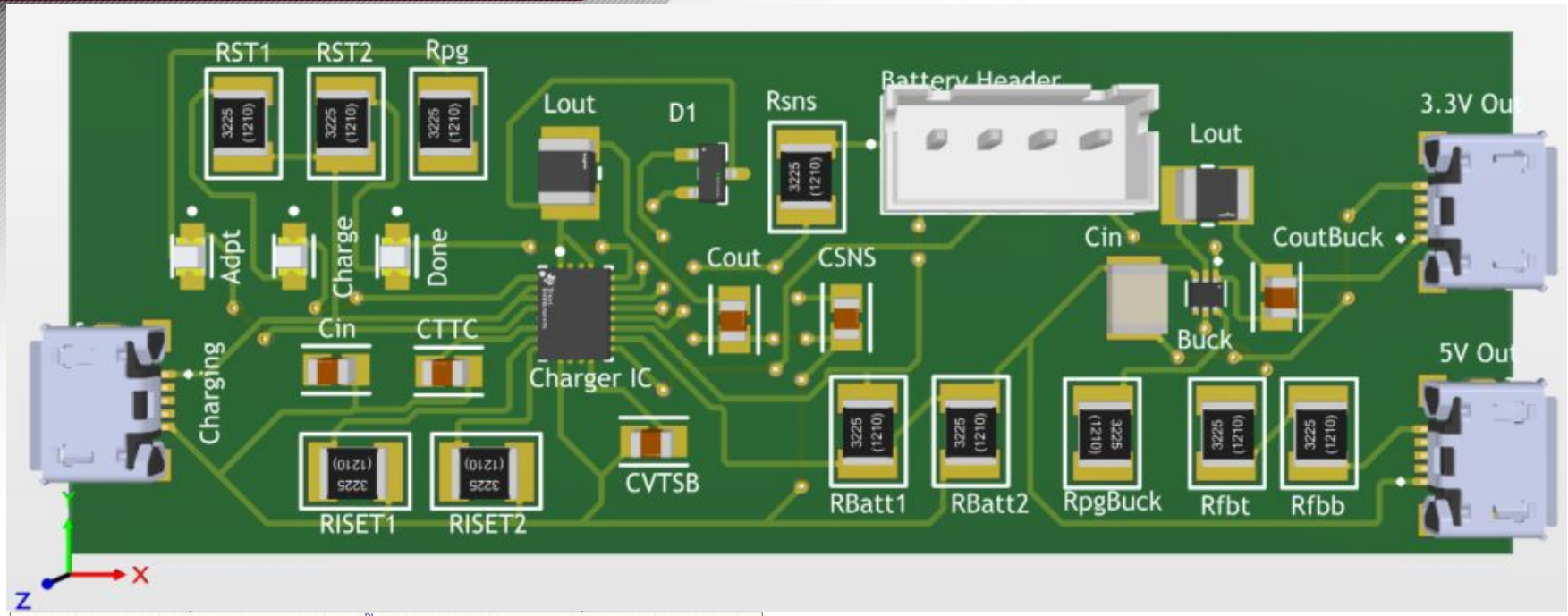
Execution and Validation

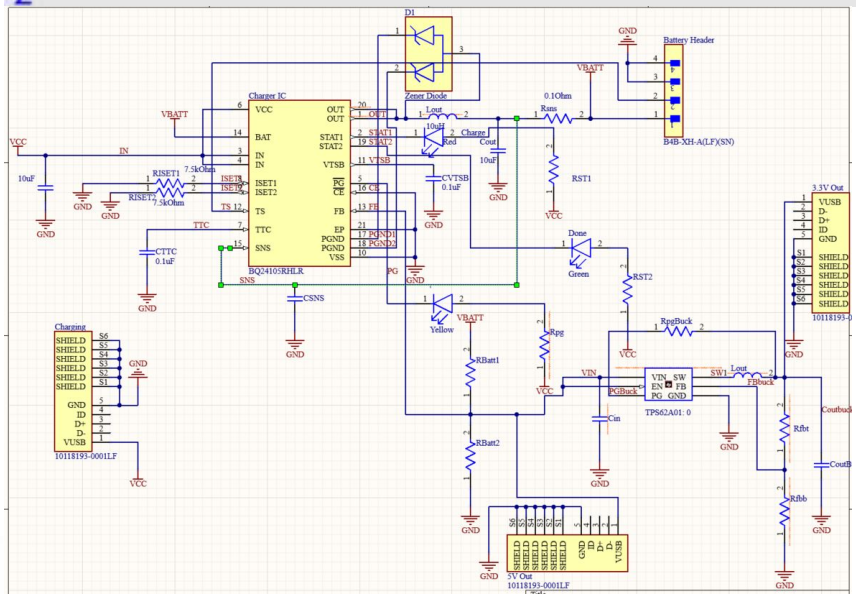
Currently: Outlined needs for each system

	March 20th	March 27th	April 3rd	April 10	April 17
Power Supply	3 V output from DC/DC Buck Converter	Buck Converter steps down 5 V input	5 V output from Power Path IC	Power Path IC can take DC input via Micro-USB charging	Complete PCB design
Microcontroller	Sample at 2 kHz ~~and get signal to noise ratio above 40 dB~~	Detection bit set when certain threshold is passed	Write data to SD card in specified format	Functioning prototype using dev board	PCB design interfaces MCU with sensors and SD card port
Android Application	Develop GUI for users to interact with app	Read data from SD card and compute metrics on data	Set up the cloud server account and file directory	Connect from the cloud server to app	Upload and download the data to and from cloud server

Power Supply Subsystem

Accomplishments since the last presentation	Ongoing progress/problems and plans until the next presentation
• Completed battery charger design • Adjusted battery charger design for 3-cell battery • Completed PCB Design • Began validation circuit for battery charger	• Incorrect breakout adapter footprint for battery charger IC • Complete battery charger validation • Measure a voltage output between 4.5 and 5.5V from the battery charger • Measure the battery charging at a 1C rate



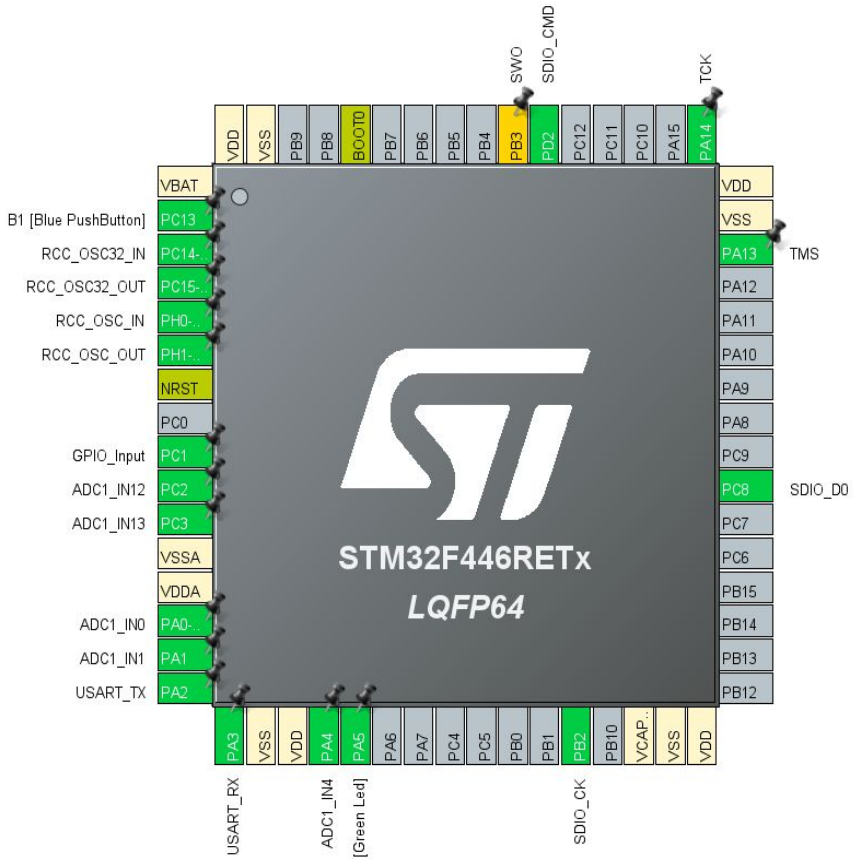


Design Parameter	Battery Charger
Inductor Ripple Current	0.398 A
Peak Current	1.529 A
Output Inductance	6.10048 uH

Microcontroller Subsystem

Accomplishments since the last presentation	Ongoing progress/problems and plans until the next presentation
• Able to read data from all 5 channels of ADC at 100 kHz • Using SDIO 1-bit to write to SD card • Detection bit set when specified threshold is passed	• Read data from signal conditioner - Replace batteries and mount BNC connectors for sensors • Need to begin PCB design - Will begin this week (maybe with help from Johnny) • Date/time information for each sample - Using RTC, but can't currently get time in microseconds • Writing to SD card is too slow - SDIO 4-bit write not working currently, reached out for help and could not fix, could be issue with jumper wires

STM32 Pinout



ADC_DATA.CSV

File Edit View

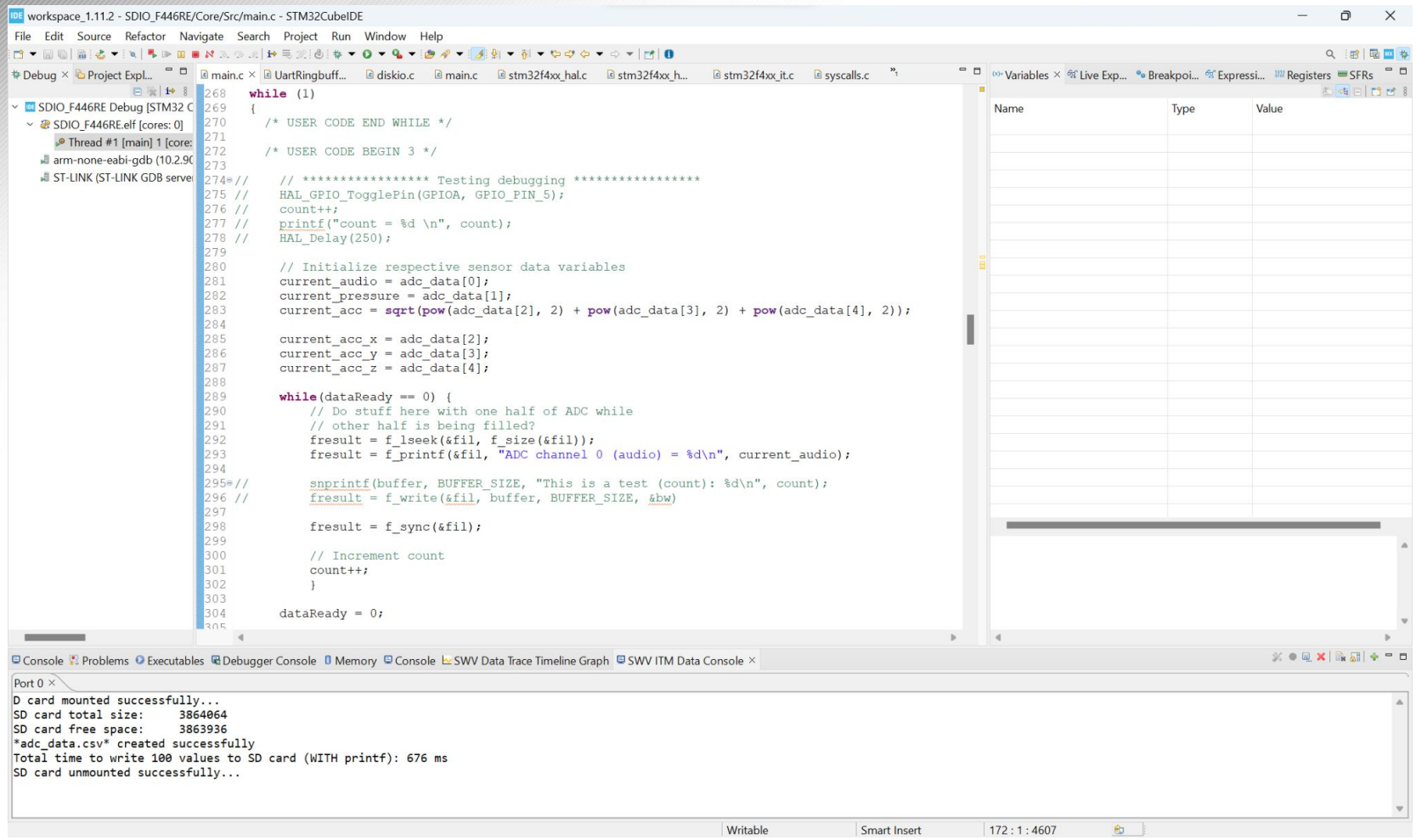
```
time, explosion, audio, pressure, acceleration
0, 1, 1165, 1093, 1802, d_audio = 1165, d_pressure = 1093, d_acc = 1802
0, 1, 1177, 1070, 1763, d_audio = 12, d_pressure = 65513, d_acc = 65497
0, 1, 1107, 1074, 1727, d_audio = 65466, d_pressure = 4, d_acc = 65500
0, 1, 1117, 1050, 1750, d_audio = 10, d_pressure = 65512, d_acc = 23
0, 1, 1152, 1041, 1720, d_audio = 35, d_pressure = 65527, d_acc = 65506
0, 0, 1177, 1060, 1741, d_audio = 25, d_pressure = 19, d_acc = 21
0, 1, 1169, 1065, 1747, d_audio = 65528, d_pressure = 5, d_acc = 6
0, 0, 1188, 1089, 1752, d_audio = 19, d_pressure = 24, d_acc = 5
0, 1, 1165, 1054, 1742, d_audio = 65513, d_pressure = 65501, d_acc = 65526
0, 1, 1176, 1050, 1749, d_audio = 11, d_pressure = 65532, d_acc = 7
0, 1, 1107, 1082, 1777, d_audio = 65467, d_pressure = 32, d_acc = 28
0, 1, 1181, 1074, 1753, d_audio = 74, d_pressure = 65528, d_acc = 65512
0, 1, 1127, 1092, 1794, d_audio = 65482, d_pressure = 18, d_acc = 41
0, 1, 1164, 1076, 1761, d_audio = 37, d_pressure = 65520, d_acc = 65503
0, 1, 1107, 1056, 1746, d_audio = 65479, d_pressure = 65516, d_acc = 65521
0, 1, 1174, 1066, 1768, d_audio = 67, d_pressure = 10, d_acc = 22
0, 1, 1120, 1067, 1755, d_audio = 65482, d_pressure = 1, d_acc = 65523
0, 1, 1163, 1033, 1742, d_audio = 43, d_pressure = 65502, d_acc = 65523
0, 1, 1189, 1092, 1746, d_audio = 26, d_pressure = 59, d_acc = 4
0, 1, 1176, 1054, 1749, d_audio = 65523, d_pressure = 65498, d_acc = 3
0, 1, 1166, 1046, 1726, d_audio = 65526, d_pressure = 65528, d_acc = 65513
0, 1, 1161, 1055, 1756, d_audio = 65531, d_pressure = 9, d_acc = 30
0, 1, 1160, 1042, 1731, d_audio = 65535, d_pressure = 65523, d_acc = 65511
0, 1, 1162, 1031, 1746, d_audio = 2, d_pressure = 65525, d_acc = 15
0, 0, 1186, 1068, 1757, d_audio = 24, d_pressure = 37, d_acc = 11
0, 1, 1164, 1051, 1744, d_audio = 65514, d_pressure = 65519, d_acc = 65523
0, 1, 1111, 1063, 1761, d_audio = 65483, d_pressure = 12, d_acc = 17
0, 1, 1148, 1118, 1784, d_audio = 37, d_pressure = 55, d_acc = 23
0, 1, 1160, 1036, 1750, d_audio = 12, d_pressure = 65454, d_acc = 65502
0, 1, 1163, 1066, 1741, d_audio = 3, d_pressure = 30, d_acc = 65527
```

Ln 40, Col 31

Written ADC values and explosion detection to .csv file

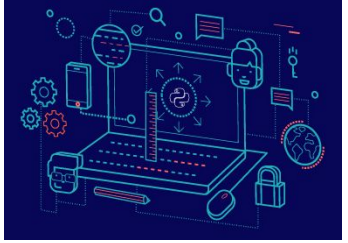
```
205 void processData() {
206 //  for(uint8_t i = 0; i < (ADC_BUFFER_SIZE)/2; i++) {
207     if(1) {
208         uint8_t i = 0;
209         // Initialize respective sensor data variables
210         current_audio = fromADC_Ptr[i];
211         current_pressure = fromADC_Ptr[i+1];
212         current_acc = sqrt(pow(fromADC_Ptr[i+2], 2) + pow(fromADC_Ptr[i+3], 2) + pow(fromADC_Ptr[i+4], 2));
213 
214         current_acc_x = fromADC_Ptr[i+2];
215         current_acc_y = fromADC_Ptr[i+3];
216         current_acc_z = fromADC_Ptr[i+4];
217 
218         delta_audio = current_audio - previous_audio;
219         delta_pressure = current_pressure - previous_pressure;
220         delta_acc = current_acc - previous_acc;
221 
222         // Do explosion detection here
223         if(delta_audio >= THRESHOLD_AUDIO) {
224             explosionDetected = 1;
225         }
226         else if(delta_pressure >= THRESHOLD_PRESSURE) {
227             explosionDetected = 1;
228         }
229         else if(delta_acc >= THRESHOLD_ACCELERATION) {
230             explosionDetected = 1;
231         }
232 
233         fresult = f_lseek(&fil, f_size(&fil));
234         fresult = f_printf(&fil, "%d, %d, %d, %d, %d, d_audio = %d, d_pressure = %d, d_acc = %d\r\n", 0, explosionDetected, current_audio, current_pressu
235         fresult = f_sync(&fil);
236 
237         // Logic for determining when to set explosionDetected back to 0
238         explosionDetected = 0;
239 
240         // The current samples will be the "previous" samples for the next samples
241         previous_audio = current_audio;
242         previous_pressure = current_pressure;
243         previous_acc = current_acc;
244 
245         previous_acc_x = current_acc_x;
246         previous_acc_y = current_acc_y;
247         previous_acc_z = current_acc_z;
248     }
```

Main function to process data from ADC and save to SD card - needs improvement



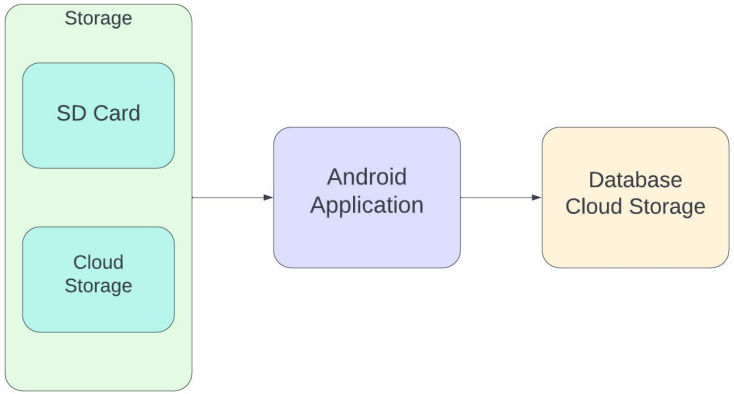
Debug print to console to verify that the SD card was mounted, the file was created and written to, and the SD card was unmounted

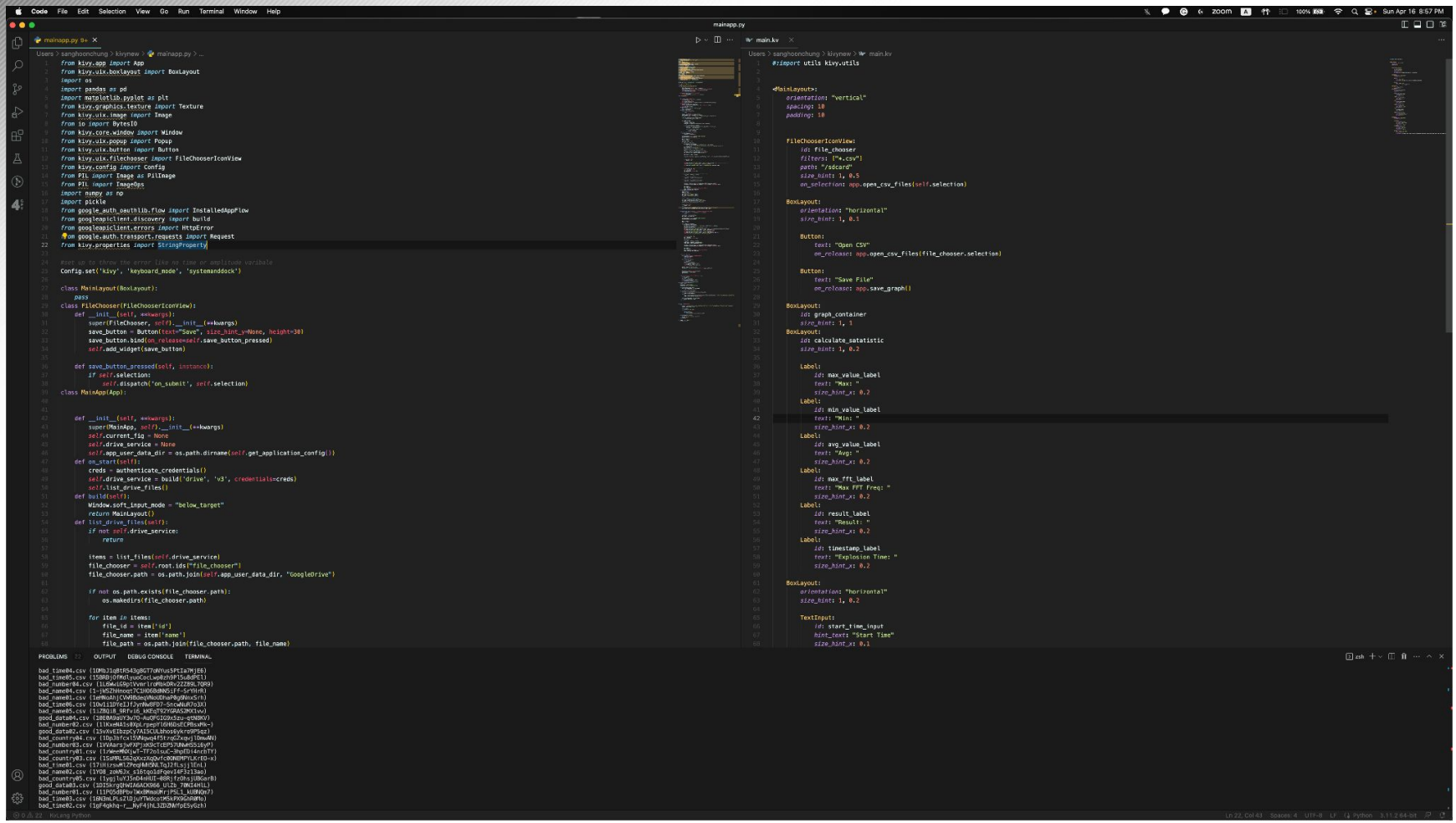
Shows how long it took to write n values to SD card (currently too slow)



Android Application Subsystem

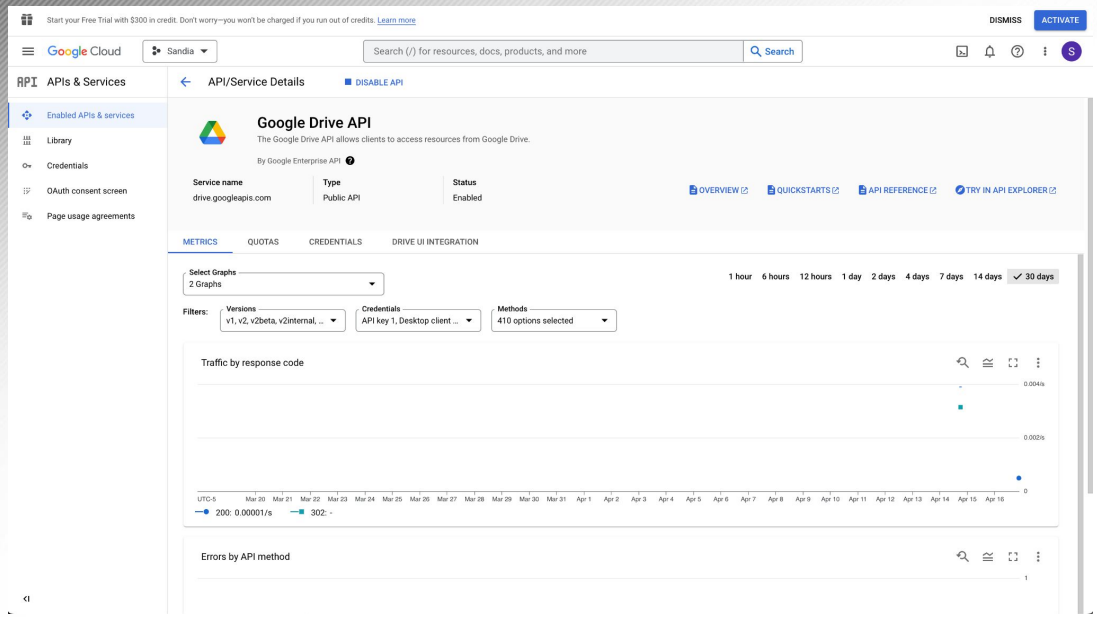
- GUI should select different channels to display a small sample of the data (summary of data collected such as max, min, avg data points, Max FFT, and presence of the explosion)
- View signal data from either cloud server or microSD card and save the file to Cloud Storage.





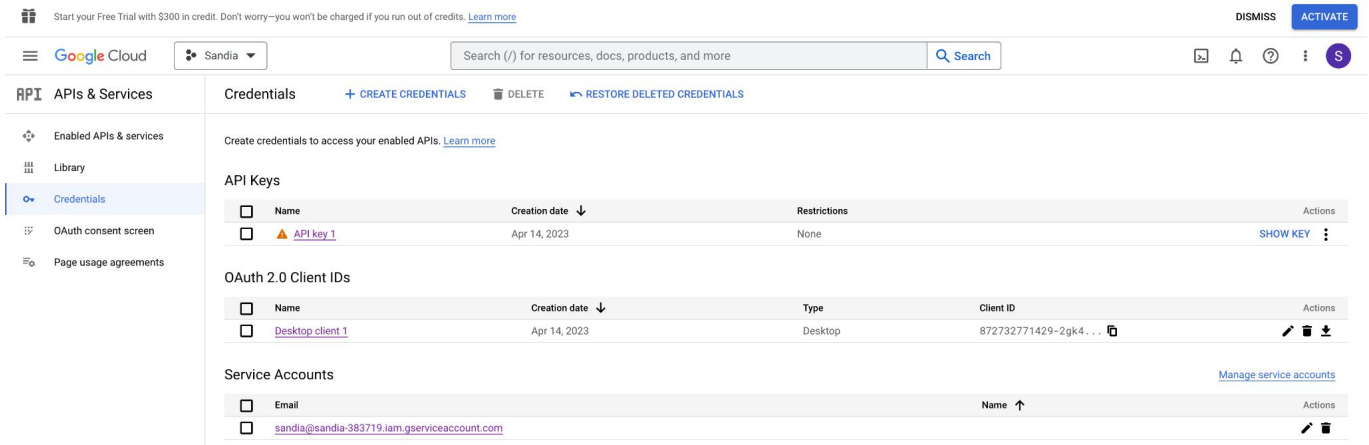
Code to develop GUI for analyzing the signal

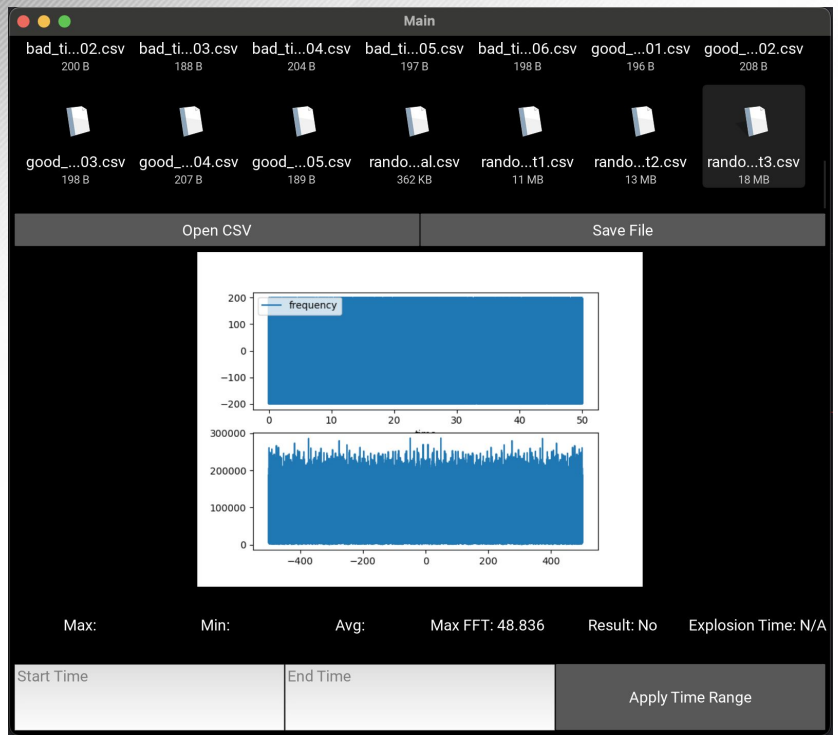
When we start to run the code, the application will read the file from the Google Cloud.



- Setting up the Cloud server account,
- Select Google Drive API for connecting with Cloud server.

Create OAuth 2.0 client IDs to import the "credentials.json" file to connect with app.





The application start to search the csv file at SD card and after click the save file button, it start to open the Google Drive directory

Add more information The app

- If the explosion occur, it shows the explosion time
- The figure of the signal for FFT value vs. FFT frequency

Goals for future - Power Supply Subsystem

403 Goals	404 Goals
• Validate battery charger circuit • Obtain voltage ripple and inductor current ripple values for power supply using E-load	• Interface power supply PCB with MCU PCB • Test power supply PCB • Validate PCB to meet voltage and current ripple requirements

Goals for future - Microcontroller Subsystem

403 Goals	404 Goals
• Complete PCB design • Interface signal conditioner with MCU • Increase SD card write speed	• Improve PCB design to include user interaction to turn on board, reset board, and remove SD card card safely • Interface with power supply subsystem

Goals for future - Android Application Subsystem

403 Goals	404 Goals
• Deploy the app to computer and smartphone • Read more than one signal • Debugging for malfunctioning. • Upload the signal from SD card to Cloud storage	• Validate and test communication between subsystems • Keep debugging for the application