Underground Vibration Classification Flow-Chart for Design Verification Testing

Test 1: Vibration Data Acquisition through Arduino microcontroller

I. This test will verify the conversion of analog vibration signals to digital values and extract the distinct features from the signals that will be used to classify them.

II. The quantitative measurements can be broken down into Analog and Digital values.

  A. For analog values, the vibrations we've observed have been within the range of 1 to 10mV depending on the vibration source.

  B. The digital values are correlated to the 12-bit resolution of the Arduino's ADC. We've implemented a DC offset that will capture the entire sinusoidal signals by offsetting the values.

  C. Once converted to digital, several calculations like the mean, standard deviation, variance, and more will be performed on the digital measurements and be placed into a feature array that will serve as the classifiers between different signals.

III. The analog signals will range between 1-10mV and will then be converted into digital values ranging between 0-4095. Due to the DC offset the analog values will converge around 2048, so the acceptable range for the digitized sinusoid signals will be from 2000-3000.

IV. The code for the feature extraction was formulated in the Test 1 code in the Appendix. The digital values were calculated by the formula: ((Vin/(2^12 - 1))*Vref) - DC offset where Vin is the analog voltage value, Vref is the operating voltage of the Arduino (3.3V) and the DC offset would be (2^12 / 2 = 2048). This formula would revert the digital values back to analog and remove the DC offset circuit so the sinusoids are centered at zero.

$\downarrow$

Test 2: Raspberry Pi

I. This test will verify that the Raspberry Pi can run a learning algorithm to classify signals from the received feature data of the Arduino and display it to the user.

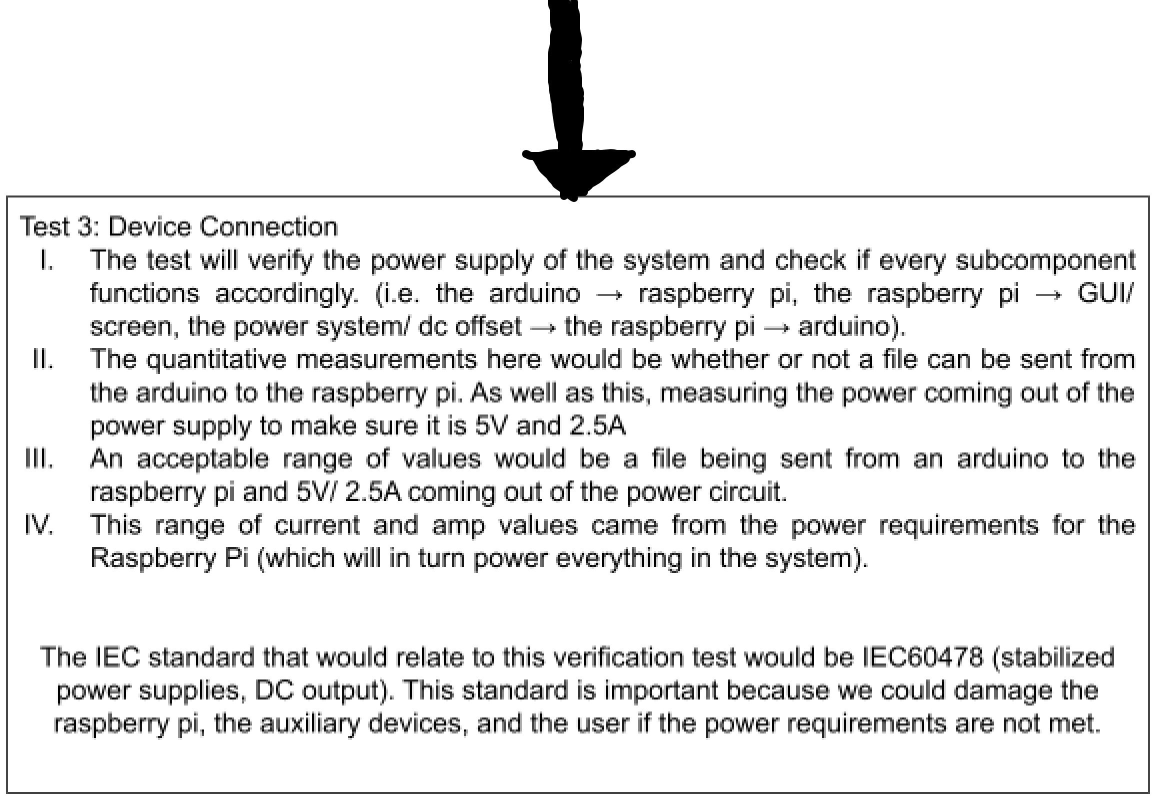
II. The quantitative measurements will be the percent accuracy of the classification model as well as comparing a known signal to the ML's classification prediction. We will also test if the GUI (running on the RPi) is able to display all of this data well enough for the user to understand.

III. An acceptable range of values for this test would be an above 80% accuracy rating for the classification model. As well as this, the GUI will be tested be simply seeing if the correct classification data/ accuracy is displayed or not.

IV. See appendix Test 2 image

An ISO standard relevant to this validation test would be ISO/ IEC 25064: User needs report. This is relevant because the GUI needs to be able to accomodate the needs of the user by displaying the proper information.

$\downarrow$



Test 3: Device Connection

I. The test will verify the power supply of the system and check if every subcomponent functions accordingly. (i.e. the arduino → raspberry pi, the raspberry pi → GUI/ screen, the power system/ dc offset → the raspberry pi → arduino).
II. The quantitative measurements here would be whether or not a file can be sent from the arduino to the raspberry pi. As well as this, measuring the power coming out of the power supply to make sure it is 5V and 2.5A
III. An acceptable range of values would be a file being sent from an arduino to the raspberry pi and 5V/ 2.5A coming out of the power circuit.
IV. This range of current and amp values came from the power requirements for the Raspberry Pi (which will in turn power everything in the system).

The IEC standard that would relate to this verification test would be IEC60478 (stabilized power supplies, DC output). This standard is important because we could damage the raspberry pi, the auxiliary devices, and the user if the power requirements are not met.

Test Reports:

<u>Test 1:</u>

To conduct the verification test for the data acquisition of the Arduino, my team and I plan on testing several days over Spring break. Ideally, Monday March 15th and Tuesday March 16th are the days we plan to conduct this test. The time of testing will be done around Noon in Tyler's backyard. Since we have to conduct our tests outdoors, the weather conditions could alter our plans. We are hoping to be able to test during a sunny day in the early afternoon with reasonable temperature and humidity, so that we can plan to be outdoors testing for several hours. Tyler will be the operator for the test and will be conducting the classification algorithms predictions of vibrations signals. Beforehand, we will have obtained a whole signal library of different footsteps, shovel, and other vibration induced sources to train our classification algorithm. This test will involve obtaining new signals and running their extracted features through the classification algorithm and verifying how accurate its prediction is. Our group has specified that a classification accuracy of 70-80% is the benchmark we are striving for. Being able to reach this accuracy rate will vastly depend on how much data we use to train our model. So if we do not reach this accuracy for the verification test, we know we will have to continue training it until it gets close to this accuracy range.

Test 2:

The second test will be conducted in parallel/ after the verification of the first test. This test will happen over spring break, most likely in the lab space. The first step in this test will be testing the learning algorithm with a test set of data (obtained from the internet). The second step of testing will be testing the algorithm with our own obtained signals and comparing them to predictions in real time. This second part of the test will be conducted after test 1 is completed. We are hoping to already have trained a model that is above 80% accurate and to receive results that correctly identify the test signals to the right classification. This section of testing will be lead by Augustus.

Test 3:

The third test will be conducted over the Spring break once the first two verification tests have been verified. Ideally, we are going to aim for the weekend of March 20th and will test in Tyler's backyard around Noon. To avoid any electrical-water damage, we hope to test during a sunny or cloudy afternoon. Either Tyler, Gus, or Vulindsky will be the operator of the test and we plan to connect the Arduino and Raspberry Pi to the power supply to verify that all components are powered and operating correctly. The arduino should obtain new vibration data, extract its features, and send these to the Raspberry pi to perform the classification algorithm on the data. The data will then be classified to some percentage of accuracy and displayed to the 3" GUI mounted to the GPIO of the Raspberry Pi. The GUI should be responsive whenever new vibration data is obtained. Once we verify that all the subcomponents are working, we will test how long the power supply can keep the system working correctly. We expect after some time, the battery will be drained and not enough power will be supplied to one of the subcomponents. This will cause accuracy of the vibration data to be more inconsistent and even possibly stop displaying data to the GUI.

Appendix

Test 1 Code:

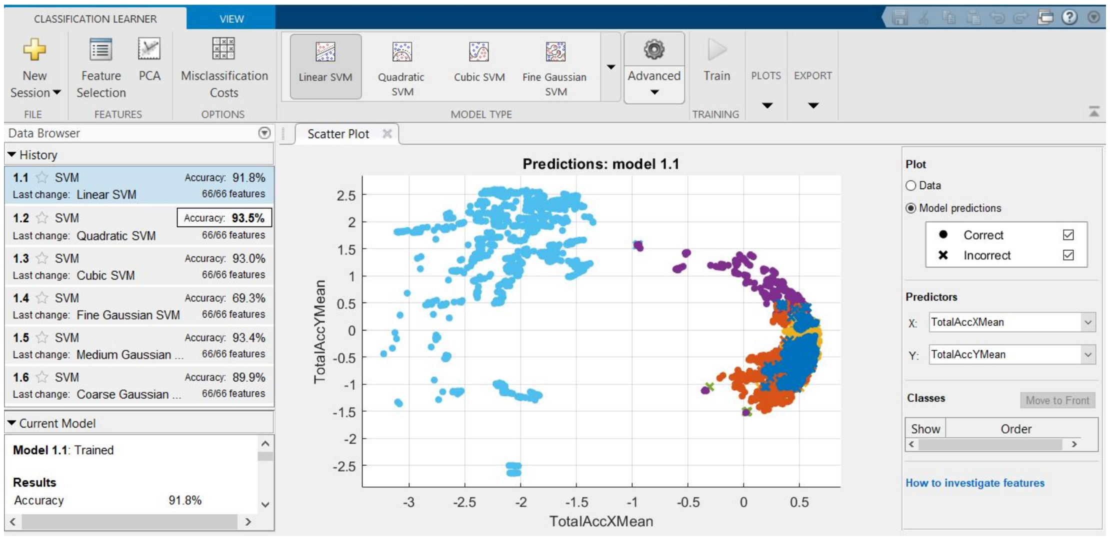
```
function feat = featureExtract(data)
% Extract features of vibrations into array
% Initialize feature vector

feat = zeros(1,22);
fs = 5000;

% Average value in signal buffer for all three acceleration components (1 each)
feat(1) = mean(data,1);
```

```
% RMS value in signal buffer for all three acceleration components (1 each)
feat(2) = rms(data,1);

% Standard Deviation of vibration signal
feat(3) = std(data,1);

% Variance normalizes by N and produces the second moment of the
% sample about its mean.
feat(4) = var(data,1);

% Mean Frequency estimates the mean frequency, FREQ, of the power
% spectrum of the time-domain signal in vector data with sample rate, Fs.
feat(5) = meanfreq(data, fs);

% Median Frequency computes the median frequency, FREQ, of the power
% spectrum of the time-domain signal in vector X with sample rate, Fs.
feat(6) = medfreq(data, fs);

% Peak to Peak measures the difference between the largest
% and smallest element in data.
feat(7) = peak2peak(data);

% Peak to RMS measures the ratio between the largest absolute
% value and root mean squared value in data.
feat(8) = peak2rms(data);

% Calculate the number of peaks in the vibration signal data
feat(9) = numel(findpeaks(data,fs));

% Calculate the number of minimas in the vibration signal data
feat(10) = numel(islocalmin(data,fs));

% Spectral peak features (12 each): height and position of first 6 peaks
feat(11:22) = spectralPeaksFeatures(data,fs);

% Autocorrelation features for all three acceleration components (3 each):
% height of main peak; height and position of second peak
feat(23:25) = autocorrFeatures(data, fs);

% Spectral power features (5 each): total power in 5 adjacent
% and pre-defined frequency bands
feat(26:30) = spectralPowerFeatures(data, fs);

% --- Helper functions
function feats = spectralPeaksFeatures(data, fs)
```

```
feats = zeros(1,12);
N = 4096;

mindist_xunits = 0.3;
minpkdist = floor(mindist_xunits/(fs/N));

% Cycle through number of channels
nfinalpeaks = 6;
for k = 1
   [xpsd, f] = pwelch(data(:,k),rectwin(length(data)),[],N,fs);
   [pks,locs] = findpeaks(xpsd,'npeaks',20,'minpeakdistance',minpkdist);
   opks = zeros(nfinalpeaks,1);
   if(~isempty(pks))
      mx = min(6,length(pks));
      [spks, idx] = sort(pks,'descend');
      slocs = locs(idx);

      pkssel = spks(1:mx);
      locssel = slocs(1:mx);

      [olocs, idx] = sort(locssel,'ascend');
      opks = pkssel(idx);
   end
   ofpk = f(olocs);

   % Features 1-6 positions of highest 6 peaks
   feats(12*(k-1)+(1:length(opks))) = ofpk;
   % Features 7-12 power levels of highest 6 peaks
   feats(12*(k-1)+(7:7+length(opks)-1)) = opks;
end
function feats = autocorrFeatures(x, fs)

feats = zeros(1,3);

minprom = 0.0005;
mindist_xunits = 0.3;
minpkdist = floor(mindist_xunits/(1/fs));

% Separate peak analysis for 3 different channels
for k = 1
   [c, lags] = xcorr(x(:,k));

   [pks,locs] = findpeaks(c,...
      'minpeakprominence',minprom,...
      'minpeakdistance',minpkdist);
```

```
    tc = (1/fs)*lags;
    tcl = tc(locs);

    % Feature 1 - peak height at 0
    if(~isempty(tcl))   % else f1 already 0
        feats((k-1)+1) = pks((end+1)/2);
    end
    % Features 2 and 3 - position and height of first peak
    if(length(tcl) >= 3)   % else f2,f3 already 0
        feats((k-1)+2) = tcl((end+1)/2+1);
        feats((k-1)+3) = pks((end+1)/2+1);
    end
end

function feats = spectralPowerFeatures(x, fs)

edges = [0.5, 1.5, 5, 10, 15, 20];

[xpsd, f] = periodogram(x,[],4096,fs);

featstmp = zeros(5,1);

for kband = 1:length(edges)-1
    featstmp(kband,:) = sum(xpsd( (f>=edges(kband)) & (f<edges(kband+1)), :),1);
end
feats = featstmp(:);
```

Test 2 Image:



History

1.1 ☆ SVM	Last change: Linear SVM	Accuracy: 91.8% 66/66 features
1.2 ☆ SVM	Last change: Quadratic SVM	Accuracy: **93.5%** 66/66 features
1.3 ☆ SVM	Last change: Cubic SVM	Accuracy: 93.0% 66/66 features
1.4 ☆ SVM	Last change: Fine Gaussian SVM	Accuracy: 69.3% 66/66 features
1.5 ☆ SVM	Last change: Medium Gaussian ...	Accuracy: 93.4% 66/66 features
1.6 ☆ SVM	Last change: Coarse Gaussian ...	Accuracy: 89.9% 66/66 features

Current Model

Model 1.1: Trained

Results

Accuracy 91.8%

Plot

Data [✓ Model predictions]

● Correct ☑

✖ Incorrect ☑

Predictors

X: TotalAccXMean

Y: TotalAccYMean

Classes Move to Front

Show Order

How to investigate features