University of Nevada, Reno

Department of Computer Science Engineering

Software Requirements Specification

November 7th, 2016

**GE Machine Learning:**

Patrick Kelley

Brian Gaunt

Harrison Stanton

Samson Haile

**Instructors:**

Eelke Folmer

Devrin Lee

Table of Contents:

Introduction 2

Summary of Stakeholders’ Interviews 3

High Level Business Requirements 4

Technical Requirements Specification 4

Use Case Modeling 5

Requirement Traceability Matrix 9

Potential Legal Issues 9

Initial Snapshots 9

Glossary 14

List of References 14

Contributions 15

**Introduction:**

For this Software Engineering class our group will be working with General Electric(GE). The problem as presented by GE is to detect and predict error modes for turbines. Currently, the process is handled by GE’s employees: They have employees who go out, pull the data, and then provide information on why errors are happening. The purpose of our project is to design and implement a machine learning algorithm to predict and classify failure modes on their turbines. Having a system in place to monitor the turbines greatly reduces the amount of time that GE has to spend on maintenance.

Our system will be able to monitor for error modes and be able to alert the client without the need for one of GE’s employees to manually comb over the data. Our plan to achieve this is as follows: First we will gather data from their rotor testing units for various levels of failure modes as well as proper operation. This will be done through usage of the API they are providing to us. More details on the API can be found below. After we obtain the data, we must then parse the data in a way that allows for usage in supervised learning techniques. This step will include labeling the data(in order to include the state of the turbine along with the corresponding data for that time period). Depending on the models that we employ after this step, we may need to further parse the data in order to develop more robust features.

We will then attempt to create the model using the data we gathered: Our intention for this stage is to take full advantage of existing libraries and software to help us build our models; some such software we have looked at to help us at this stage include sklearn, OpenCV, Keras and TensorFlow. We will experiment with several different models to find the most apt tool for the job. Although the robustness and popularity of neural networks is prominent, we may choose to use other models since the amount of data we generate may fall short for training. One such model that may work well with a limited amount of data is random forests. If the results from our initial attempts are lacking, we modify the model and gather more data if needed. We will continue to modify our model until we achieve a relatively low error rate. After we reach this stage we will then continue to add on functionality for other error modes. At the end of this project, it is our goal to be able to accurately predict and classify different error modes for GE’s turbines.

**Summary of Stakeholders’ Interviews:**

1. Roughly how much does GE spend on repairs that could by this protection system? (i.e. in dollars per year, unless there is a better or more accessible metric) If there is no way of knowing this, maybe a rough estimate of the Reno area’s repair spending?
   * GE itself doesn’t spend any money on repairs, although they are invested in their clients not having to either.
2. How accurate are employees at predicting failures using the information we will be using? Is it likely that a well-trained employee would misclassify a potential risk to the system being monitored?
   * The employees are extremely accurate, and this is what their business is primarily built on. It is very unlikely that an employee would misclassify a risk to the machinery being monitored.
3. Do you see the value of this project lying more in improvement of accuracy or in the ability to have employees working on more demanding/important tasks? Or in something else?
   * The main value in the project is that it expands the resources employees have to diagnose problems. In particular, our system is meant to point the employees in the right direction without them having to sift through tons of sensor data.
4. Is there any other information GE might be able to gain from more accurately knowing when these machines are close to an error condition? Note: If not, a simple “No” is fine here.
   * No
5. How would you measure the accuracy of the algorithm? Percent of correct classifications? Precision/Recall? Something else? We expect false negatives are worse than false positives, and maybe there is a better-suited method than adjusting the threshold accordingly?
   * Percentage of positive classifications
6. How accurate do you expect the algorithm to be with respect to that measure of accuracy?
   * Ideally 100%
7. How likely do you think it is that GE will use our software if it works with a high level of accuracy?
   * The team’s advisors are not sure how likely this is.
8. For senior projects in the past, has GE typically used the software the teams made? If so, to what degree?
   * They have used the projects before, but many of them use various languages/platforms/APIs that are incompatible with GE’s standards.
9. If GE does decide to use the software, does that require any extra work on our part to integrate the software with the systems it would operate on? If so, what might that entail?
   * The team’s advisors do not foresee any additional work from us on the project
10. If the software works well, how likely do you think it will be that the team will get the chance to expand on the project with GE? (e.g. adding new classes, improving accuracy, etc.)
    * Again, the team’s advisors do not foresee any additional work unless we seek employment from GE.
11. How well do you expect the results from the rotor kit to generalize to all of the machines GE would use the software on? If this is a concern, is there anything we can do to make it generalize better?
    * The team’s advisors expect the data to generalize well, but if it does not they will take on the responsibility to generalize the system.
12. What information can we expect to get from the Fleet API? The JSON schema would be ideal, but if that information is unavailable, what is the current purpose intended for the API? (So we can have a general idea of what to expect)
    * The team’s advisors do not know much about the Fleet API yet, as it is a new system that we will be beta testing as we use it.

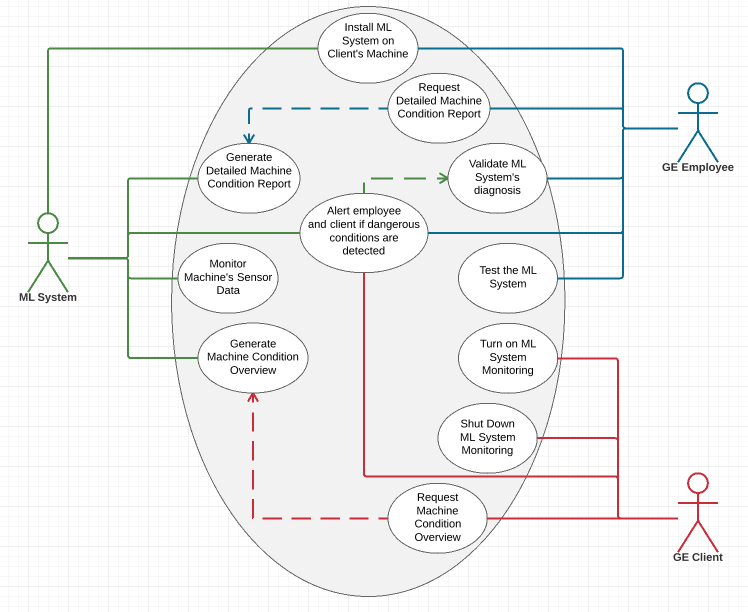
**High Level Business Requirements:**

* The team shall generate rotor kit data(i.e. sensor outputs)
* The team shall retrieve and analyze data from the Fleet API
* The team shall test different types of classifiers to find out which approach gives the best results.
  + The results of these tests will be documented so the team at GE will know how the team came to their conclusion.
* The system shall use sensor data to detect and classify conditions that are potentially dangerous to machinery.

**Technical Requirements Specification:**

* *Functional Requirements:*
  + First Priority:
    - The system shall detect when either of two dangerous conditions occur.
    - The system shall classify which of the two dangerous conditions occurred when it detects such a condition.
  + Second Priority:
    - The system shall detect two additional dangerous conditions.
    - The system shall classify two additional dangerous conditions.
  + Last Priority:
    - The system shall detect/classify more dangerous conditions.
    - The system shall give a confidence level for the chosen classification.
* *Non-functional Requirements:*
  + First Priority:
    - The system shall be as accurate as possible.
      * The team shall provide benchmarks of the system’s performance.
  + Second Priority:
    - A classifier shall be chosen to minimize the amount of data needed to train for new error conditions.
    - There shall be thorough documentation regarding the team’s decision on an algorithm.
  + Third Priority:
    - The system shall run as efficiently as possible in regard to both computation and memory.

**Use Case Modeling:**



* Generate Detailed Machine Condition Report: This detailed report will include how likely each error scenario currently is, accompanied by relevant sensor output . This information is meant for a GE employee who is trained to understand this sensor output, and is meant to give more information about the current state of the machine than the overview that is meant for GE’s clients.
* Alert employee and client if dangerous conditions are detected: This is the main function of the protection system. The machine learning algorithm will be trained to recognize a number of potentially dangerous conditions in the machinery; if one of these conditions is detected, it will inform GE and their client about the error.
* Monitor Machine’s Sensor Data: The ML System will be constantly taking in sensor data from the machine so that it can detect dangerous conditions at any time.
* Generate Machine Condition Overview: This overview will simply state the status of the machine; in particular, this report will include how likely each monitored condition currently is. This information is available on request so that the client can check on the status of the machine.
* Install ML System on Client’s Machine: One of GE’s employees will have to install the system on the client’s machine.
* Request Detailed Machine Condition Report: GE’s employees can request a detailed report on the machines condition, including how likely each error scenario currently is, accompanied by relevant sensor output.
* Validate ML System’s Diagnosis: When the Machine Learning System detects a dangerous condition in the machine, it will send relevant sensor information along with its diagnosis. GE employees will be able to check the algorithm’s diagnosis to be sure it is correct.
* Test the ML System: An employee can give the machine test data, or manually simulate error conditions using a rotor kit to see if the protection system is functioning properly. Errors could arise over time due to problems in sensors, and the system should ideally be tested regularly.
* Turn on ML System Monitoring: After the protection system has been turned off, it should be simple to turn it back on again. Again, this should be the default behavior for the system.
* Shut Down ML System Monitoring: By default, monitoring should remain on, but for the purposes of testing the system it might be beneficial to turn off the protection system, especially if it is linked to automatic reactions(such as automatically turning the system off) in the future.
* Request Machine Condition Overview: The client can request an overview of their machines condition. This overview will include how likely each error scenario currently is.

|  |
| --- |
| **Use Case: Generate Detailed Machine Condition Report** |
| **ID: UC1** |
| **Actors:**  Machine Learning System  GE Employee |
| **Preconditions:**   1. The system is installed on the machine 2. The machine is running |
| **Flow of events:**   1. The use case is triggered by the GE Employee requesting the report 2. The system checks how the machine is operating with respect to each of the tracked error conditions 3. The system generates a report including how likely each error condition is and the relevant sensor information |
| **Postconditions:**   1. The GE Employee who made the request receives a report with the likelihood of each error condition and the corresponding sensor data |

|  |
| --- |
| **Use Case: Generate Machine Condition Overview** |
| **ID: UC2** |
| **Actors:**  Machine Learning System  GE Client |
| **Preconditions:**   1. The system is installed on the machine 2. The machine is running |
| **Flow of events:**   1. The use case is triggered by the GE Client requesting the report 2. The system checks how the machine is operating with respect to each of the tracked error conditions 3. The system generates a report including how likely each error condition is |
| **Postconditions:**   1. The client who made the request receives a report with the likelihood of each error condition |

|  |
| --- |
| **Use Case: Alert employee and client if dangerous conditions are detected** |
| **ID: UC3** |
| **Actors:**  Machine Learning System  GE Employee  GE Client |
| **Preconditions:**   1. The system is installed on the machine 2. The machine is running |
| **Flow of events:**   1. The sensors send data to the Machine Learning System 2. The system analyzes the data using the Machine Learning Algorithm 3. The Machine Learning Algorithm returns a set of values corresponding to likelihoods for each error condition    * If any of the values are above a threshold, the system will notify the GE Employee and the GE Client      + Note: The notification sent to the GE Employee will include relevant sensor data so the employee can validate the system’s diagnosis |
| **Postconditions:**   1. If the machine is in a dangerous condition, the GE Employee and GE Client will receive a notification. Otherwise nothing will happen |

|  |
| --- |
| **Use Case: Validate ML System’s Diagnosis** |
| **ID: UC4** |
| **Actors:**  Machine Learning System  GE Employee |
| **Preconditions:**   1. The system has sent the GE Employee a notification about an error condition in a machine |
| **Flow of events:**   1. The GE Employee analyzes the data sent with the system’s diagnosis, and verifies that the diagnosis is correct 2. The GE Employee will notify the client about whether or not the diagnosis was correct |
| **Postconditions:**   1. The client will know if they need to fix the problem reported by the system |

**Requirement Traceability Matrix:**

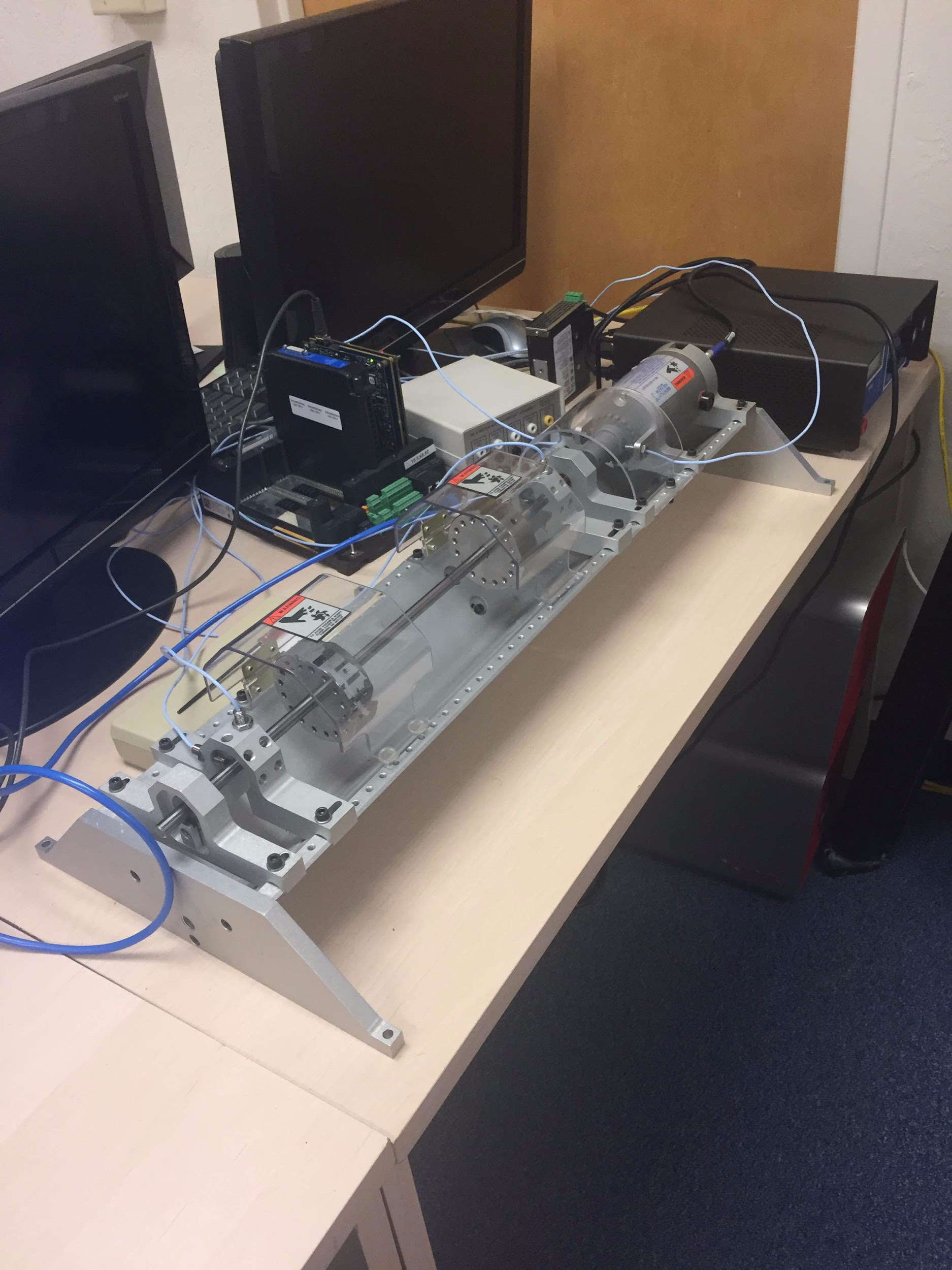
|  |  |
| --- | --- |
| SR ID | Functional Requirement |
| SR 1 | The system shall generate a detailed condition report at the request of a GE employee |
| SR 2 | The system shall generate a condition report at the request of a client |
| SR 3 | The system shall alert users to any and all dangerous conditions |
| SR 4 | The system shall be able to process test data |
| SR 5 | The system shall be able to be turned off by the user |

**Potential Legal Issues:**

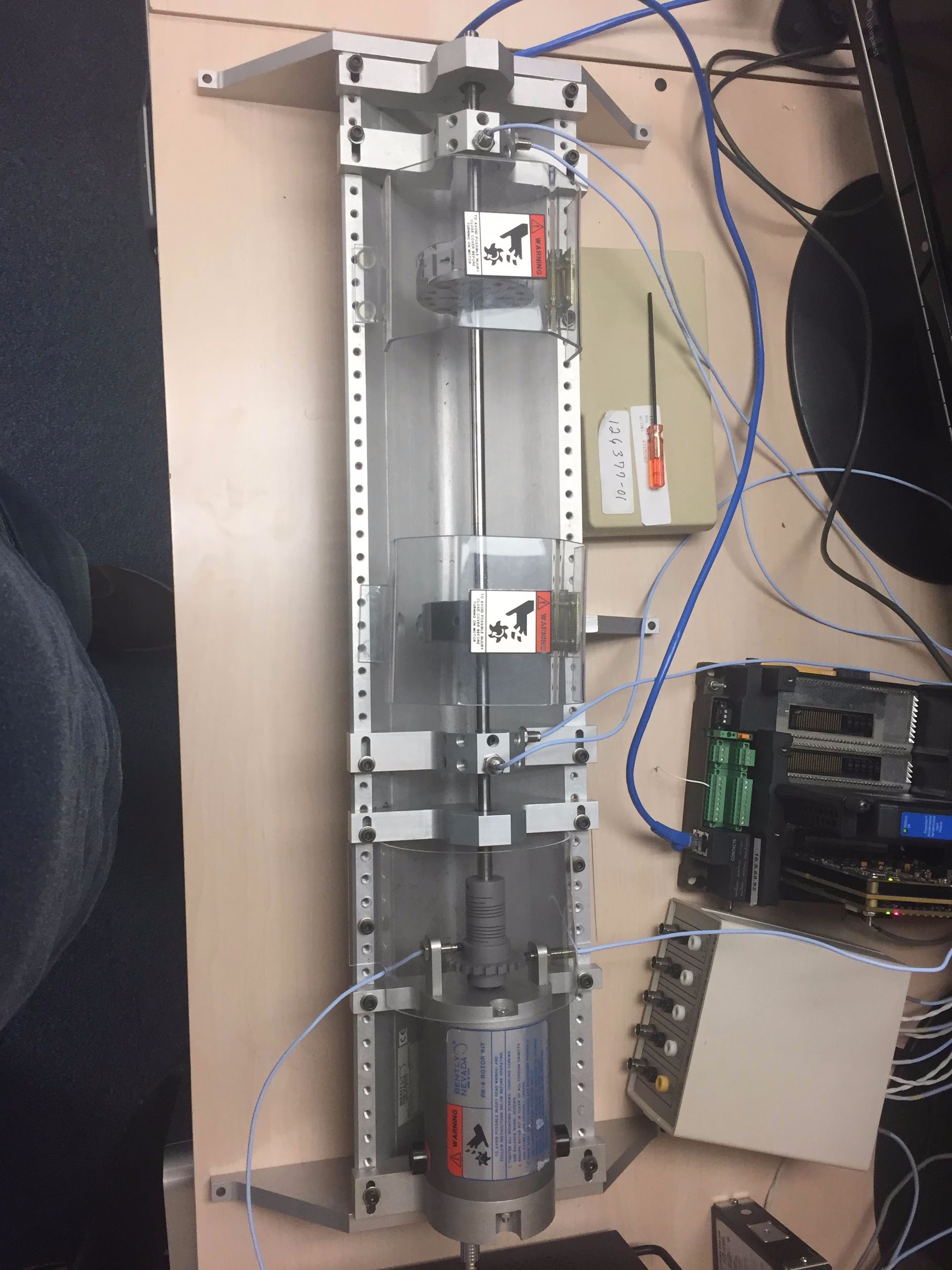
* A non-disclosure agreement was signed by each member of the group to resolve the following issues:
  + Specifying the prohibited sharing of the Fleet API and System 1 with other individuals
  + Specifying the prohibited sharing of equipment provided for use with the project
  + Maintaining confidentiality of trade secrets and documentation provided for use with the project.

**Initial Snapshots:**

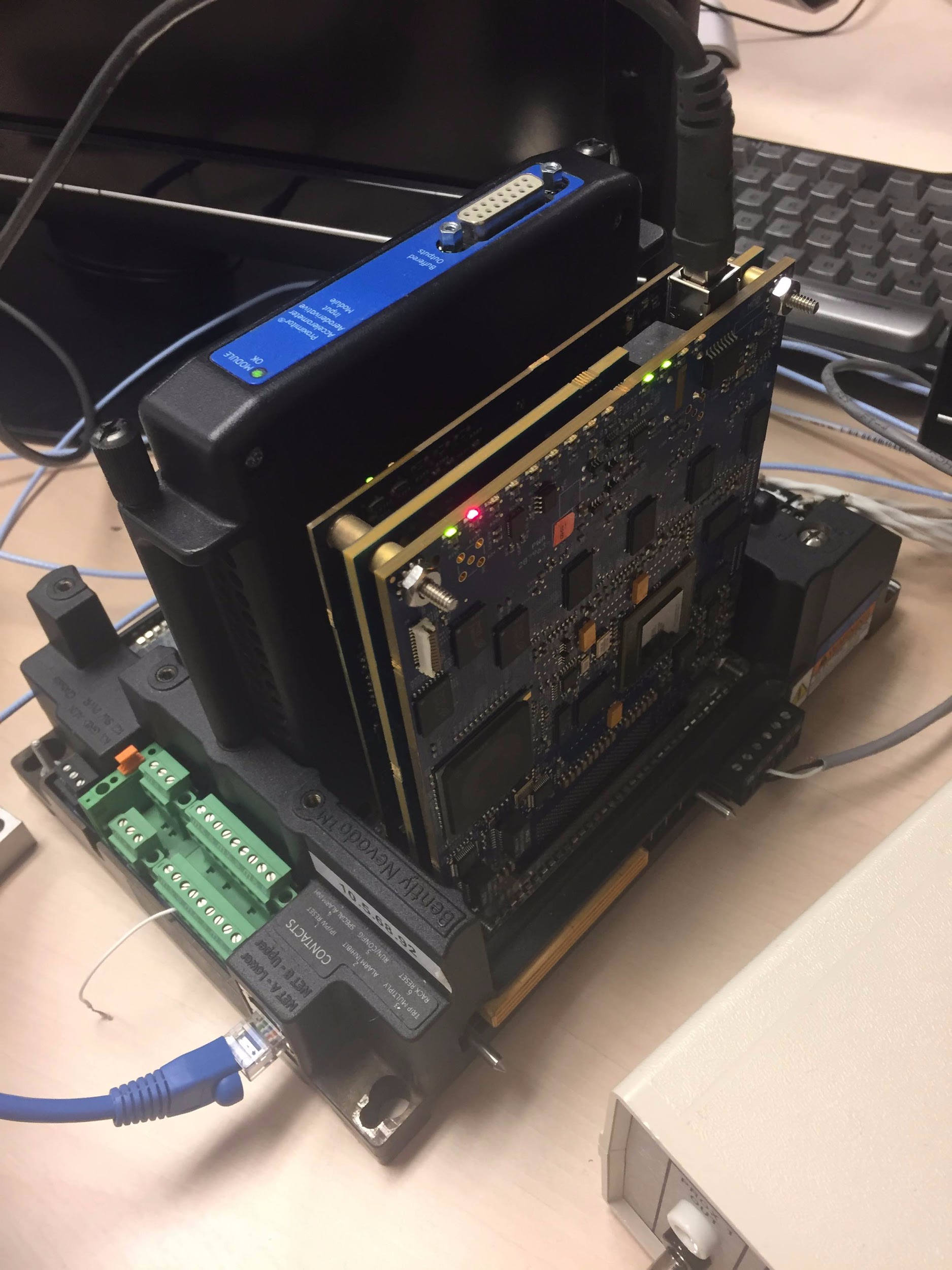
These snapshots include both the setup of our rotor kit and different components, including the programs we will be using to monitor the performance of the rotor kit. We are waiting on GE to deliver the Fleet API component before we can use everything as a whole, meaning the program snapshots will be minimalistic.



*Figure 1: Displays the entire rotor kit components.*



*Figure 2: Rotor Kit Turbine that the team will spin to test for failure modes.*



*Figure 3: Power Supply and Ethernet Connection*



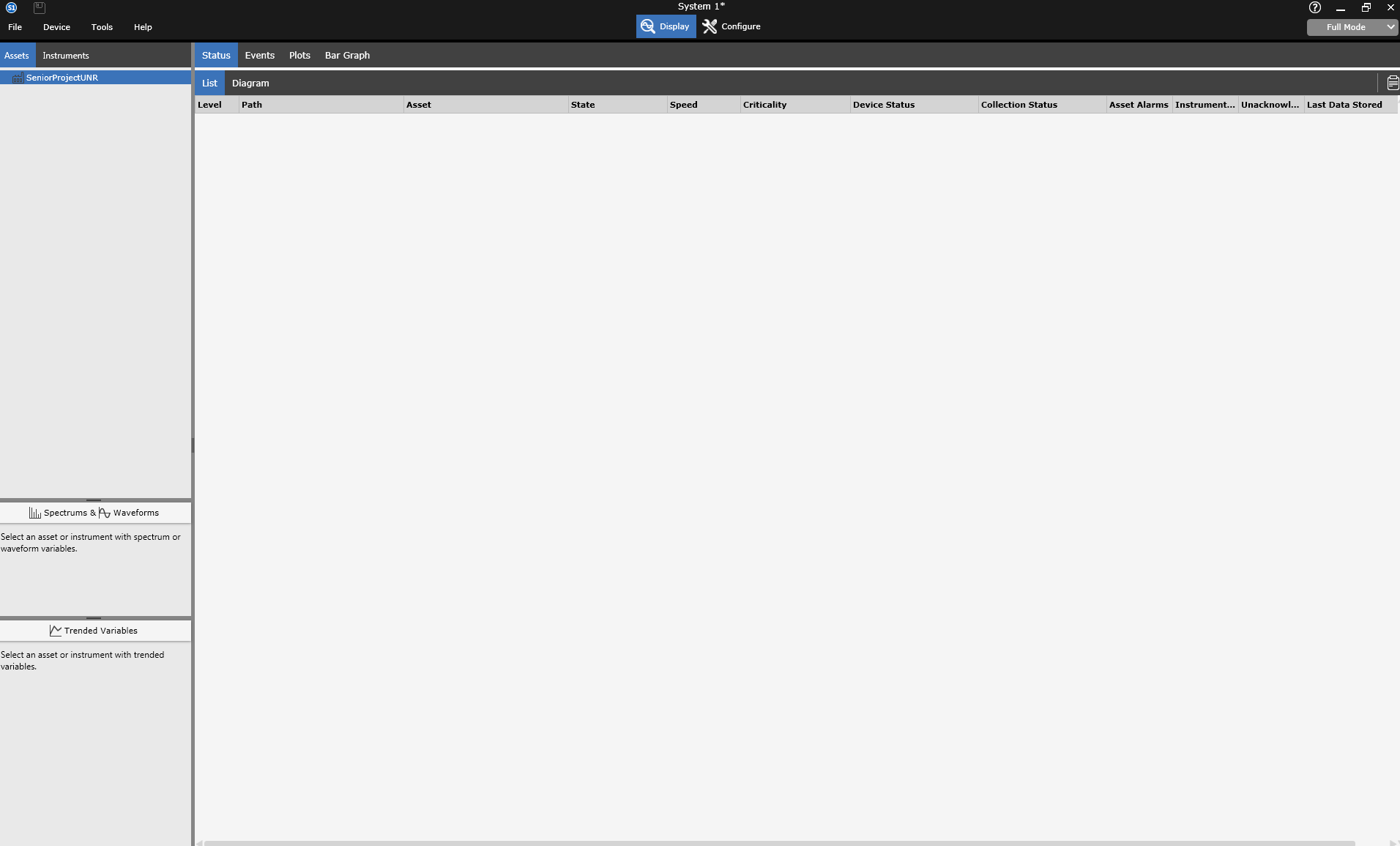
*Figure 4: Proximitor Assembly*



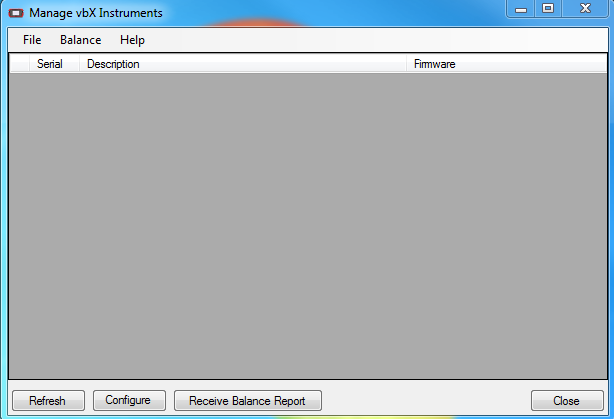
*Figure 5: Rotor Kit Motor Speed Control*



*Figure 6: System 1 is the program we will use to model the behavior of the rotor kit graphically*



*Figure 7: The interface for System 1 that we will use, which will contain graphs of how the turbine is moving.*



*Figure 8: VBX Instruments manager so that System 1 can communicate with the rotor kit instrument*

**Glossary:**

* Machine Learning(ML): A branch of artificial intelligence where the software is “trained” by processing data and adapting itself to operate better on that data.
* JSON: Stands for JavaScript Object Notation. This is a simple data format that is commonly used for sending information on the internet.
* API: Application Program Interface; the set of routines, protocols, and tools that denotes how software components should interact.
* Neural Network: A machine learning classifier that uses multiple layers of functions chained together to train a model to map given inputs to an output layer.
* Random Forest: A machine learning classifier that constructs decision trees to divide and classify the data.
* Decision trees: A technique which classifies data by subdividing it into smaller parts.
* Supervised Learning: A machine learning method of training models from labeled training data.

**List of References:**

* <http://usecase-traceabiltymatrix.blogspot.com/> - This article explains and outlines how a requirement tracibility matrix should look as well as the different components that can potentially be added for effectiveness. It also outlines the properties of use cases such as the flow of events, preconditions and postconditions.
* <https://www.gemeasurement.com/condition-monitoring-and-protection/software/ges-system-1-condition-monitoring-and-diagnostic> - This website describes in the purpose and performance of the System 1 Conditioning Monitoring Software that we will be using for our testing. The website outlines the System 1’s in terms of user experience, capability, accessibility, and embedded expertise.
* <http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/> - This is a tour of machine learning algorithms that describes the some of the different algorithms we may develop for this project, including supervised learning, decision trees, random forests, and neural networks which are defined in the glossary.
* <http://www.commtest.com/what_s_new/product_releases/index> - This describes the purpose of the VBX manager and how it will relate to our system. The instruments we are using are SCOUT instruments, and we can use VBX manager to configure and communicate between the instruments and the System 1 software.
* <https://coreos.com/fleet/docs/latest/api-v1.html> - This outlines the documentation and functionality of the Fleet API that we will be using in our project. It contains information such as how to create, modify, list, and delete unit entities, as well as describe how error communication will work.
* <http://www.w3schools.com/js/js_json_intro.asp> - This contains an introductory course into the inner workings of the JSON file format. The website contains information on what a JSON is, the different components of it, as well as different examples that flesh out the syntax of how the JSON file will look.

**Contributions:**

Patrick Kelley:

Summary of Stakeholder’s Interviews, High-Level Business Requirements, Technical Requirements Specification, Use Case Modeling

Brian Gaunt:

Initial Snapshots, List of References

Harrison Stanton:

Introduction and relevant glossary terms

Samson Haile:

Requirement Traceability Matrix, Potential Legal Issues