2024

Software Suite

THE CREATION OF AN INTERNAL SAAS PROJECT

STUDENT 220491901

Contents

1 Introduction

Metrology, the science of measurement, has many interesting aspects to the field but it is probably best known for the calibration of equipment which involves two steps: a verification and, if necessary, an adjustment. The verification process is what happens 90% of the time. A piece of equipment will come to the lab and either a technician or engineer will follow set procedures to verify that the equipment is performing as optimally as it should be. If the measurements are within the specified manufacturer's tolerances, then there is no need to adjust, or repair, the equipment.

That may seem simple, but in reality there are a few more steps to consider what would pass or fail the verification process besides the manufacturer's tolerance limits. Because this is the science of measurement, all aspects of the process are considered such as room temperature, time since last auto-test or auto-calibration, length of the cables, accuracy of the display, number of measurements taken at each setpoint, or even if the battery is new. In a perfect world, most of these processes would be repeatable and reproducible, but that is where the science part comes in with the uncertainty calculation or budget.

2 Presentation of the problem

That brings me to the main issue with the calibration process: not everything is automated… yet. Granted, even with automated procedures there is still some manual intervention, whether that be to change the cable connections or change equipment settings or whatever else the automated procedure asks the user to do. As long as the equipment has the ability to be remotely controlled or communicated with, then there is no reason not to have an automated procedure. Why? Well for one, it could cut down calibration times from 6-8 hours down to 1-2 hours, which is very important when you work in a service industry that focuses on output and turnaround times. Another good reason to have equipment automated is for repeatability and/or reproducibility when it comes to the measurements.

When it comes to calculating the uncertainty of a measurement, there are two types of measurements that are considered which are Direct and Indirect. Electrical equipment only deal with direct measurements, making the calculations much easier since we do not have to worry about room temperature having any affect on the warping of the instrument, operator bias, computation error, or reference vs transfer standard aka the instrument used for the calibration.

The following equation is what an uncertainty boils down to, simplified to only having one standard, where u means Uncertainty, $u_{\varepsilon_{\bar{x}}}$ means Combined Uncertainty, STD means standard and UUT means Unit Under Test.

$$u_{\varepsilon_{\bar{x}}} = \sqrt{u^2{}_{Std\_bias} + u^2{}_{Std\_resolution} + u^2{}_{UUT\_resolution} + u^2{}_{repeatability}}$$

Std_bias, Std_resolution, and UUT_resolution all come from the manufacturer's specifications, for example ±10V, 0.0001V, and 0.01V respectively, but the repeatability is taken at the time of

measurement which is calculated based on the number of readings (n). The following is the equation for the standard deviation of the sample:

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Where x_i is the i^{th} reading and the mean value ($\bar{x}$) of the sample is computed from:

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + \ldots + x_n)$$

Which we then use in the following calculation for the repeatability uncertainty:

$$u_{\bar{x},repeatbility} = \frac{s_x}{\sqrt{n}}$$

As you can see, having a high standard deviation will greatly affect the uncertainty of the measurements, thus creating a need to make sure that there are as many steps as possible to reduce that number. There is mention of reproducibility in the introduction, which is another source of error along the lines of repeatability if not done consistently. To clear things up: repeatability is the number of readings taken per measurement, aka can we repeat this result multiple times, whereas reproducibility asks if we reproduce the same results with different operators or personnel [1].

3 Aims and Objectives

Although there are automated procedures and/or software already, all are owned by different companies in which we are not able to fix if issues arise nor can it be modular by adding new equipment. This has the technician or engineer search every program to see which equipment is compatible with the correct software. When it does not exist then the last option is to create a procedure ourselves. If push comes to shove, then everything can be done manually, but time is the biggest factor for the industry.

My goal, this is ultimately long term and therefore the project will be the beginning steps of the final vision, is to have a software "suite" where all the self-created software resides together. There will be the inevitable searching of each bought software and now this suite for compatibility, but it is way better than our current solution of having standalone software for individual equipment (with potential of not knowing those individual programs exist).

With that, this project is meant to help my employer out by not only creating a new calibration program, but also to update the process in which future engineers can follow with better standards and practices. Most engineers write quick and dirty scripts to get the job done, but there is the potential for technicians who do not know how to code or are tech savvy to know how to properly use the scripts, thus having the need to have a frontend/GUI is huge. This GUI will also need to be able to be flexible enough to account for varying screen sizes as well as resolutions. The program will be planned to be

used just within my department starting out, but hopefully I can spread it around to the rest of the company and the other facilities.

4 Methodology

There have been programs created in the past by some interns, which have mostly been written in C#, but end up as just hard coded solutions. So, if anything needs to be changed, then the source code needs to change and turned into an executable again. The issue is that the way the interns published their programs make that impossible these days and partly created this need to change our practices, coupled with the fact that C# is not as popular as other languages such as Python.

So, we start from there, picking a new language that is easier to understand or learn for beginners as well as easier to write with (given the fact you can code basic commands from within the Command Prompt or Terminal). It also helps when testing out commands to the instrumentation without needing to open an IDE or compile a program multiple times for minor adjustments. Python, the chosen language for our case, provides more flexibility regarding the libraries that can be installed and supported throughout version updates. One of the major things the software needs to do is interact with an Excel file, and the way Python handles that interaction vs how C# handles it is a lot more simplified and does not hog the resource in case other files need to be opened.

With the backend doing some interaction with a database, there needed some code that could handle SQL, which Python does seamlessly. One of the previous engineers did create an API to get live temperature and humidity data for the lab, but they also wrote a Python package the database interaction since there is need to not only retrieve instrument information but also to write measurement data to the database. Instead of reinventing the wheel, using Python for the software suite for this part was a no-brainer.

Coming from an arts background, I wanted to have the software be more coherent with the company's brand. All previous software from the interns tried their best to match it by using a similar color scheme from the logo, but the buttons and design were all over the place and looked like it was something out of the late 90's early 2000's. Luckily companies have a branding toolkit hidden somewhere, so all you have to do is some searching to find it and follow along. In our case, since the front is planned to be more flexible, it will be written using HTML/CSS but more specifically using the Jinja2 library. Why? The company already has a Django template that can be downloaded and modified. Although Django was considered as a framework, but it was doing too much that was not needed so then considered using Flask (a lighter version of Django), but that was also too much for our needs. I then found a GitHub repo with a lighter version of Flask called Eel [2]. Granted, I do not think I am using the framework properly, pushing it past it limits, so there are still hiccups that happen but overall it has worked out quite well for the needs of the software.

One of the main complaints about the older programs, and the ones we get from other vendors such as Fluke's MetCal [3] and Northrup Grumman's SureCal [4], from the technicians using them is that

there is not much flexibility if something goes wrong like forgetting to switch connections from one channel to another. Sometimes when an instrument fails and we decide to limit the range of use, we like to test different setpoints to see which point it no longer fails, and now that will have to be done manually once the program is done running. The technicians would love to be able to change the program's setpoint so that they do not have to type in all of the settings on the different instruments (they normally get reset back to default at the end of the program). With that in mind, the software will include individual pages for each test that is done that will show which setpoints are being test and the settings being applied that are editable fields so that users can change whichever they like to correctly dial it in.

Let me backtrack and explain how exactly the software is going to handle these calibration procedures. All of this could not be done unless an engineer has already figured out how to perform the verification and adjustment procedures, which standards are needed to meet the needs, and created a datasheet and report to save all the results via an Excel file. In the end, the software is reading the setpoints from the Excel file and following the steps from what the engineer has established. Finally, the software then saves the Excel file to the correct location within our server and open it up so that the user can double check everything is correct then ultimately convert it into a PDF to provide for the end user of the instrumentation.

5 Plan for Development

I am not the best at judging how long things will take, plus this project was started some time ago so it is not something that is happening from scratch for the next couple of months. Doing this for the project allows me to have a reason to work on it during the evening or over weekends, not just trying to complete it during my down time in the office. Also, since this will be an ever-evolving platform, I had no set goal myself on when it should be completed. By having the project end in September, this will give me a firm date of when to have a working product that would hopefully be in an executable format, even if it is only for one instrument. I would like to have at least two instruments done so that it gives a clearer idea of how the program will work, but when we do not see specific instruments more than once a year for only a week at a time (if that), then it is hard to fully test if things are working properly.

Since there is the need to be connected to a physical unit as well as having the API connections that require connection to the company WiFi/Internet, the submitted product will be created with a "simulation" mode. Some of the features will be disabled (just commented out but will still be included to show what is meant to happen) and there will be fake results included that will represent different aspects of the functionality of the software (Fails and Passes).

6 References

[1] NASA Satern Training. (2020). Measurement Uncertainty Analysis. [Online]. Available: https://satern.nasa.gov/

[2] Eel. (2023). Python-eel. Accessed: February 13, 2023. [Online]. Available: https://github.com/python-eel/Eel

[3] MetCal Software. (2024). Fluke. Accessed: June 20, 2024. [Online]. Available: https://us.flukecal.com/products/calibration-software/met-cal%C2%AE-software

[4] SureCal Calibration Software. (2024). Northrup Grumman. Accessed: June 20, 2024. [Online]. Available: https://www.northropgrumman.com/cyber/surecal-calibration-software