SIP Report_25204_Shahidan bin Idris_EE

by Shahidan Bin Idris 0025204

Submission date: 08-Dec-2020 01:07PM (UTC+0800)
Submission ID: 1468357905
File name: SIP_Report_25204_Shahidan_bin_Idris_EE.pdf (3.25M)
Word count: 22304
Character count: 116765

STUDENT INDUSTRIAL PROJECT (SIP) REPORT

SEPTEMBER 2020 – DECEMBER 2020

DATA COLLECTION SCRIPTS AND CONTACT RESISTANCE (CRES)/SPRING RATE MEASUREMENT TOOL FOR IN-CIRCUIT TESTER (ICT) FIXTURE PROBE HEALTH MAINTENANCE AND LIFE EXTENSION

at

INTEL PRODUCTS (M) SDN BHD

by

SHAHIDAN BIN IDRIS

25204

BACHELOR OF ELECTRICAL & ELECTRONICS ENGINEERING

Appendix I

VERIFICATON STATEMENT

I hereby verify that this report was written by SHAHIDAN BIN IDRIS (Student's Name) and all information regarding this company and the projects involved are NOT CONFIDENTIAL / CONFIDENTIAL (strikethrough not relevant).

Host Company Supervisor's Signature & Stamp	B/P: Intel Product (M) Sdn. Bhd. (365735-X) Lot 8, Jalan Hi-Tech 2/3 Kulim Hi-Tech Park 09090 Kulim Kedah, Malaysia
Name:	GOH FONG HRIA
Designation:	TEST MANAGER
Host Company's:	INTEL PRODUCT (M) SDN BHD
Date:	7/DEC/2020

ACKNOWLEDGEMENTS

Firstly, I would like to thank the Career Development Office, Centre for Student Development, Universiti Teknologi PETRONAS (UTP) for the opportunity to undergo Student Industrial Project (SIP) as part of the graduation requirement of my undergraduate studies in Electrical & Electronic Engineering. The experience has provided valuable knowledge and exposure in the technical aspects of the industry, as well as given the opportunity to apply the theoretical knowledge gained from university and the host company to contribute back in the form of a project.

My deepest appreciation also goes to Intel Corporation and Intel Products (M) Sdn Bhd, the host company in which I have undergone my Student Industrial Project with. The valuable training and knowledge provided by the company has equipped me with the necessary skills to not only complete tasks and assignments at the company, but to also successfully complete the internship project. A special thanks goes to my host company supervisor, Ms Goh Fong Hsia. Her supervision, guidance, encouragement, and feedback throughout the project execution has greatly eased the process, as well as helped with the completion of the project. I would also like to thank my work buddy, Cavan Tan Chu Siong, who has devoted his time to providing guidance and feedback not only for the project, but also the training and teachings given throughout the internship period with regards to PCBA Parametric test and in-circuit-tester (ICT) technology. The knowledge gained from the training has helped in completing the project to contribute back to the ICT team.

Finally, I would like to thank my internship supervisor from UTP, Dr Cheng-Kai Lu, for his guidance. Dr Lu has provided valuable support, especially in the final stages of my Student Industrial Project. The advice he has provided has greatly helped me in producing the internship reports, evaluations and documentations needed, as well as in carrying out my internship presentations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 ABSTRACT

1.1 Introduction

This report describes the project 'Data Collection Scripts and Contact Resistance (CRES)/Spring Rate Measurement Tool for In-Circuit Tester (ICT) Fixture Probe Health Maintenance and Life Extension'. The project covers a series of tools used to enable the health maintenance and life extension of ICT fixture probes used in SIMS. For the data collection scripts, the tools that have been developed include the ICT Fixture Preventive Maintenance (PM) Report Generation Tool, Probe Cycle Count Harvesting Script, and Probe Replacement Tracking Tool. As for the measurement tool, a device is proposed and designed to enable measurement of contact resistance (CRES) and Spring Rate (Spring Force over Spring Travel) of ICT fixture probes.

For ICT in the factory, much of the effort has been placed on improving the test performance. Test performance is related to how well the tester is able to detect true defects within a PCBA, while not false-calling healthy PCBA. Through the investigations, it has been found that a majority of No-Defect Found (NDF) is caused by poor contact between fixture probes and the PCBA test points. This is largely due to the probe failures, caused by damages, defects, and probes being worn out. To investigate and develop and action plan to overcome the probe failure issues, while improving the test performance for ICT (by reducing NDF rate), it is necessary to collect engineering data regarding the ICT fixture probes, such as measurement of lifespan of the probes.

The scripts and tool developed for the health maintenance and life extension of ICT fixture probes play an important part in gathering the necessary data for probe health measurement. The variables in which these tools measure include probe cycle count, failure mode causing probe replacements, probe contact resistance (CRES), and probe spring rate. The data gathered allows SIMS, specifically Parametric Test team, to quantitatively understand the health and lifetime of ICT fixture probes in production, while allowing development of new approaches and innovations to extend the lifetime of ICT fixture probes beyond existing probe cycle count limits.

To enable the gathering of probe cycle count data, the tools that are developed are the ICT Fixture Preventive Maintenance (PM) Report Generation Tool and Probe Cycle Count Harvesting Script. These Python-based tools have been developed to help gather the data needed for probe health measurement, while also serving additional functions for the ICT technicians and engineers. From the Report Generation Tool, the hardware probe cycle count (probe cycle count that is displayed on the fixture itself) is collected into a separate database, after it is extracted from the reports generated by PM crew after maintenance activities. As for the Probe Cycle Count Harvesting Script, the tool is developed to collect the software probe cycle count (probe cycle count that is generated through test program into a counter file) from all the products in the factory, then displaying the cycle counts in a combined table.

Another important data feature that is needed is the probe ID, probe specifications, and failure modes that are causing the probe replacements. This is done by developing a tool (Probe Replacement Tracking Tool) to track the probe replacements each time this troubleshooting approach is conducted. After the replacements are done, the technicians shall record the probe ID that has been replaced with the respective failure modes, then the script will auto-populate the inputs with the related probe specifications (probe size, probe location, coordinates within the fixture, etc) from the probe lists. This provides ample engineering data for the engineering team to investigate probe health and root cause for probe failures within ICT.

While the probe cycle count is able to estimate the health of the ICT fixture probe, a direct measurement of electrical and mechanical properties of the probe itself serves as the best approach in measuring the lifespan of an ICT fixture probe. As such, exploration and designs in developing a contact resistance (CRES) and spring rate measurement device have been done to pave the way for enabling this capability within ICT. By allowing the measurement of CRES and Spring Rate, it is possible to understand the trend and relationship between these measurements, along with the probe cycle count, failure modes, and probe specifications, with the duration that the ICT fixture probes are run in production (the lifespan).

1.2 Problem Statement

During the start of the internship period, it is found that the Parametric Test team is unable to store and retrieve ICT fixture preventive maintenance (PM) reports after PM activity. There is no trigger for PM crew to conduct Level 2 PM (fixture reprobing) when probe cycle

count exceeds the cycle count limit (probe is no longer healthy to be used for production). Furthermore, ICT fixture PM crew has no tool to record and track software probe cycle count and display fixtures that need Level 2 PM (reprobing). With regards to handling probe replacements during production, Parametric Test team is unable to analyse the relationship between ICT fixture probe replacements and failure modes when repeated failure or NDF occurs. While there is estimation that can be done using the probe cycle count, Parametric Test team is unable to quantitatively measure and study the health of ICT fixture probes after probe replacements are conducted.

1.3 Objective

Once the project is completed, the 'Data Collection Scripts and Contact Resistance (CRES)/Spring Rate Measurement Tool for In-Circuit Tester (ICT) Fixture Probe Health Maintenance and Life Extension' project should cover a few objectives, namely, to develop a tool that can generate ICT fixture PM reports faster and easier and to develop a report generation tool that is dynamic and adaptable to changes made in the factory specifications (addition or removal of steps in activities). From the fixture PM reports, the project aims to harvest/collect the hardware probe cycle count from ICT fixture PM reports generated by PM crew and conduct data visualisation from the probe cycle count. Aside from hardware probe cycle count, the tool will gather the software probe cycle count from all fixtures running in the factory by reading the individual counter files and display in a single dashboard. When there is probe replacements conducted during production, the tool will help in recording and tracking the probe replacements done during production as troubleshooting approach during repeated test failures or NDF. After referring to probe lists for each fixture, the tool will auto-populate probe replacement data with engineering data, such as probe specifications, probe location, and coordinates within the fixture. For a more accurate measurement of ICT fixture probe lifespan and health, the project aims to enable measurement of electrical and mechanical properties (contact resistance and spring rate) of ICT fixture probe.

1.4 Methodology

The data collection tools were developed mainly using Python. A number of Python libraries were used to support the functions of the tools, such as *PySimpleGUI* for Graphical

User Interface (GUI), *pandas* for data handling and DataFrame building, *matplotlib* for Pareto Chart and bar chart plotting and many more.

When developing the tools, UI/UX design principles were implemented in order to build an appealing and easy-to-use software for ICT. At the same time, the tools were developed to allow easy editing and modifications to be done to the contents of the software, which is accomplished by reading steps and questions from an Excel file, in order to allow flexibility and adaptability to factory specification changes.

Since data collection is the main purpose of the project, it is necessary to implement data visualisation within all of the software tools. For the majority of the tools, the Pareto Chart and/or bar chart of the top features are produced and displayed to the user. Aside from data visualisation, this raw data is stored as CSV file to allow the user to conduct their own study and analysis.

As for the CRES and Spring Rate Measurement Device, exploration and studying was done to identify the electrical and mechanical properties to be measured. The first designs of the device, as well as the mechanism for measuring the ICT fixture probe CRES and spring rate were produced. The project is further continued with communication and negotiation with vendor to receive measurement range and specifications, higher level device designs, as well as quotation of developing the device. The execution of development and measurement using the CRES and Spring Rate Measurement Device is to be done after the internship period.

1.5 Results and Conclusion

From the data collection tools, technicians are able to generate ICT fixture PM reports more quickly and easily, compared to the previous condition of editing Word template and emailing or printing the reports. The tools have also provided a centralised database in the form of a network drive to store and keep track of the inputs and records. This allows users, especially engineering team, to retrieve and study historical reports and records for further engineering study.

While the CRES and Spring Rate Measurement Device is not developed and used for measurement during the internship period, the scope of work and contributions during the

internship period have allowed the Parametric Test team to explore the capability once the Return of Investment (ROI) has been solidified. The first designs have been provided to the vendor, as well as the confirmation on the necessary measurement range as well as device specifications. The vendor has provided their designs and quotation for the device.

In total, the data collection tools, and CRES and Spring Rate Measurement Device has enabled the collection of various data features that will help in the investigation of root cause for high NDF rates caused by probe failures, study of ICT fixture probe health, as well as the potential extension of probe lifespan.

2.0 INTRODUCTION

2.1 Background of study

At SIMS, printed circuit boards (PCBs) go through board assembly, followed by board testing, and finally board integration. Board testing is located middle-of-line in the production process. Board testing, handled by Test Engineering department, is divided into Parametric and Functional test. Parametric test involves detection of manufacturing defects before functional test. Parametric test includes measurement of electrical parameters of the PCBA. This involves measuring for shorts, opens and analogue values (resistance, capacitance). Through parametric test, it is possible to identify component defects (faulty resistors, capacitors), or defects such as lifted pins, missing components or poor soldering.

In Parametric Test, there are two types of testing conducted, which are In-Circuit Test and Flying Probe Test. In-circuit testers implement "bed-of-nails" technology to do board testing. This technology has the PCBA placed onto a pad with multiple probes, giving a graphical description of the technology. Practically, in-circuit testing relies on these probes to contact with the PCBA's test points. The probes then send input signals and receive output signals to conduct different tests. Examples of the different tests conducted in ICT machines include pin test, short test, analogue test, digital test, Testjet and Boundary Scan.

In test engineering, much effort is placed in improving test performance, increasing test coverage, and enhancing test capabilities. When it comes to test performance, optimally, a tester machine should detect all faults or failures on a board correctly. Some of the scenarios which highlight a poor test performance include the tester not being able to detect a true failure (no-defect-found, or NDF) or that the tester is being "overzealous" and calls out a failure on a healthy test point – a "false-positive" (in production, this is also classified as NDF, albeit the defect is found when there is no defect). At the time of writing, majority of the efforts in ICT is put towards studying NDFs, identifying root cause, and reducing NDFs (thereby increasing test performance).

One of the root causes of NDF is contact issue. This is caused by the ICT probe having a poor contact and connection to the test point on a PCB. Contact issues introduce errors or fluctuations in the parametric testing process of driving input and receiving output from test

points. At severe levels of contact issue, the received output at test points may stray far from the upper and lower limits of a test procedure, thus causing NDF. There are many reasons that contribute to the contact issue, two of the reasons include foreign materials (FM) coating the surface of the ICT fixture probe or test point, as well as unhealthy or damaged ICT fixture probes.

As of the time before the projects are implemented at the factory, SIMS does not have a quantitative method to measure the health of ICT fixture probes. While it is identified (or estimated, hypothesised) that an NDF is caused by contact issue or a damaged fixture probe, there is no existing tool or approach to measure the health of probes. In production, engineers and technicians rely on the probe cycle count to ensure only healthy probes are being used. The probe cycle count is the count of the frequency a fixture is run with a board test. When a fixture is loaded onto a tester machine and tests a board, the cycle count goes up by one count. Effectively, the probe cycle count can give an estimate of how many test-runs, and thus how long have the probes been used in production. Different factories and companies employ different limits for the probe cycle count for the probes to be considered healthy and usable in production. When the probe cycle count exceeds this limit, the ICT fixture probes are replaced (this process is known as reprobing) with new probes.

In a production setting, it is important to ensure probes are regularly maintained and replaced when the probe cycle count exceeds the cycle count limits. By complying to the cycle count limits and replacing the fixture probes on time, NDF is greatly reduced, as there is a less likelihood that the ICT fixture probe is damaged or unhealthy. The probe cycle count is displayed and obtained in two places: the probe cycle count display on the ICT fixture (hardware cycle count), and the counter file generated and updated after each test-run in the tester PC.

For the hardware cycle count, the probe cycle count is displaced on a screen which is located on the ICT fixture. The preventive maintenance (PM) crew records the probe cycle count during each Level 1 PM (general PM activity, such as cleaning and basic replacement of parts). Once the probe cycle count is identified and recorded to be above the cycle count limit, the fixture is scheduled to undergo Level 2 PM (enhanced PM activity, includes fixture reprobing). At the time of writing, there is no existing method or automation to trigger Level 2 PM for the ICT fixtures. While the fixtures are mostly scheduled for maintenance and

maintained in a timely manner, the lack of a trigger or dashboard, as well as a database for collecting and displaying probe cycle counts of ICT fixtures used in production gives room for error and non-compliance to the existing procedures. As a result, some ICT fixtures do exhibit contact issues due to unhealthy or damaged probes, caused by a high probe cycle count on the fixtures.

The software cycle count is obtained from special files that are generated by the tester PC. The tester PC retrieves the probe cycle count from the fixture, generates or updates a file (that has no file extension) containing an integer value of the probe cycle count, and stores this file at the respective board folder in the production server. While the PM activity involves observing and recording the hardware cycle count displayed on the ICT fixture, there are instances whereby the display is damaged, the battery of the probe cycle count display is faulty or dead, or that the hardware cycle count has been reset (either accidentally or intentionally) by tampering with the circuitry. While it is possible to tamper with the hardware cycle count, it is slightly more difficult to do the same for the software cycle count. Introducing an automation tool to collect or harvest the software cycle count from the counter files is a useful approach to ensure consistency and genuineness of the recorded probe cycle count during PM activity.

However, relying solely on probe cycle count only gives an estimate of the health of a probe. The cycle count merely gives the number of test-runs the probes have been used in. There are two major problems that occur when relying solely on probe cycle count for probe health:

1. ICT fixture probes that have exceeded the cycle count limit may still be healthy. This may be due to the limits set by the factory or probe manufacturer being too low (underestimating the lifetime of a probe) or that the probes could still be used in testing and produce an error or deviation in measurement that does not exceed the upper or lower limits of the test procedures.

2. ICT fixture probes are damaged prematurely, whereby the probes are unusable in production, although their probe cycle count is below the cycle count limit. This could be due to many reasons, such as foreign materials (FM) damaging the probe contact surface, board manufacturing defects (warpage, bent or misplaced components)

damaging the probes, or mishandling of tester machine, ICT fixture or probes during production.

As such, there needs to be a more direct means of measuring the health of ICT fixture probes, rather than using the probe cycle count. From the research and study conducted (which is described in detailed later in this report), it is viable to measure the physical quantities and electrical parameters of the ICT fixture probe to determine the health of the probes. These quantities and parameters are the contact resistance, or CRES, of the probe, as well as the spring rate, which is the quotient of spring force and spring displacement of the probe.

An ICT fixture probe should have a very low contact resistance (CRES). The presence of CRES is what introduces additional impedance in the measurements during PCB testing, thus affecting the measurement value and test performance. During PCB testing, there are a number of impedances that exists between the input signal source (point at which tester machine starts introducing the test signal through the device-under-test, or DUT) and the output signal receiver (point at which tester machine reads the measurement value). In other words, many impedances exist within the testing circuit, aside from the DUT. These impedances add up together and cause errors or deviations in test measurement. When discussing the CRES, we put focus on the internal resistance of the ICT fixture probe, as well as the contact resistance between the probe contact surface and the test point on the PCB. The task of CRES measurement is a very delicate matter, as the values of electrical resistance for this measurement come in milliohms (mΩ). As such, very specialised tools and equipment designs are needed to achieve CRES measurement of ICT fixture probes.

An ICT fixture probe contains a spring. This spring allows the probes to pressed down, either by a pneumatic/mechanical press or vacuum, to contact the test points on the PCB. An important parameter for the health of an ICT fixture probe includes the health of this spring. Should the spring be unhealthy or malfunctioning, this would cause the probes to have improper contact with the test points, thus introducing contact issues and NDFs. An example of a severe case of spring damage is a jammed spring. Not only would the ICT fixture probe have a poor or improper contact onto the test point, the high contact force from the probe due to a jammed spring may damage the test point, the PCB itself, or even the fixture. As such, the probe spring rate, which is the measure of the quotient of spring force and spring displacement of the ICT fixture probe, is included as a measurement parameter for the health measurement.

Similar with the CRES measurement, the measurement of spring force and spring displacement is an equally challenging and delicate task, as the values for these measurements are extremely small. Specialised tools and equipment designs are needed to achieve spring rate measurement of ICT fixture probes.

2.2 Problem Statement

Before the project, there have been a number of problems that have been identified through the briefing provided by the Host Company Supervisor (HCSV), with regards to the data collection and measurement of ICT fixture probe health. Aside from that, there are also some challenges that can be solved concurrently, while seeking to produce engineering data needed for maintenance of probe health and extension of probe lifespan.

The problem statement for the project is as below:

- Parametric Test team is unable to store and retrieve ICT fixture preventive maintenance (PM) reports after PM activity. There is no trigger for PM crew to conduct Level 2 PM (fixture reprobing) when probe cycle count exceeds the cycle count limit (probe is no longer healthy to be used for production)
- ICT fixture PM crew has no tool to record and track software probe cycle count and display fixtures that need Level 2 PM (reprobing)
- Parametric Test team is unable to analyse the relationship between ICT fixture probe replacements and failure modes when repeated failure or NDF occurs
- Parametric Test team is unable to quantitatively measure and study the health of ICT fixture probes after probe replacements are conducted

3.0 OBJECTIVE

The main objectives of the project 'Data Collection Scripts and Contact Resistance (CRES)/Spring Rate Measurement Tool for In-Circuit Tester (ICT) Fixture Probe Health Maintenance and Life Extension' are:

i. To develop a tool that can generate ICT fixture PM reports faster and easier
ii. To develop a report generation tool that is dynamic and adaptable to changes made in the factory specifications (addition or removal of steps in activities)
iii. To harvest/collect the hardware probe cycle count from ICT fixture PM reports generated by PM crew and conduct data visualisation from the probe cycle count
iv. To gather the software probe cycle count from all fixtures running in the factory by reading the individual counter files and display in a single dashboard
v. To develop a tool that can record and track the probe replacements done during production as troubleshooting approach during repeated test failures or NDF
vi. To auto-populate probe replacement data with engineering data, such as probe specifications, probe location, and coordinates within the fixture
vii. To enable measurement of electrical and mechanical properties (contact resistance and spring rate) of ICT fixture probe in order to study the lifespan of probes

3.1 Scope of Work

From the objectives mentioned above, there is exploration, research, and studying done in order to accomplish the tasks for the project. While the objectives of the project are highlighted to showcase the overall goals of the project, the following is the scope of work covered during the internship period for the project:

- Analysed the available variables (data features) that are related to the ICT fixture probe health
- Studied the probe cycle count limits within the factory as well as the behaviour of probe cycle count for healthy and unhealthy probes
- Analysed the different datapoints that can yield probe cycle count data (hardware probe cycle count and software probe cycle count)
- Studied and understood the current conditions of producing ICT fixture PM reports, as well as the weakness of using the current conditions
- Investigated different programming tools and framework to execute the project tasks, such as studying website development languages and tools, common scripting languages, such as Python, Perl and Visual Basic for Applications (VBA)
- Explored UI/UX design principles in producing an appealing, interactive, dynamic, and easy-to-use tool for ICT
- Explored and researched on different electrical and mechanical properties used to identify ICT fixture probe health, as well as the method of measurements for these properties
- Communicated and worked with vendor to establish scope of work for the measurement device, provided and received designs on the device, as well as negotiated the features and quotation of the device development

4.0 LITERATURE REVIEW

4.1 Feinmetall Contact Technologies – Life Cycle Test of Contact Probes

Part of the process of exploring the measurement of electrical and mechanical properties that affects the lifespan of ICT fixture probes involves knowing what some of the parameters and techniques used by other organisations and institutions are. The documentation provided by Feinmetall Contact Technologies, which is a company focused on developing ICT fixture probes as well as other testing probes, helps in understanding some of the existing measurements and tools.

The documentation used is a catalogue or review of the various products and services available from Feinmetall. In the catalogue, the company discusses on the various types of probes that are available for sale. While they discuss the quality and durability of the ICT fixture probes available, Feinmetall also gives a detailed explanation on the life cycle test that the company conducts on their probes.

From the catalogue, it is found that Feinmetall mainly measures the following, as part of their life cycle test:

- Spring Force against Spring Travel (this is known as the Spring Rate)
- Contact Resistance (CRES) of the probe
- Spring Force against Contact Cycles (known as Probe Cycle Count in the factory)
- Contact Resistance (CRES) against Contact Cycles

Feinmetall also conducts these tests under specific conditions, which are ambient temperatures of 20ºC to 30ºC, relative humidity of 40 to 60%, and within a dust-free environment.

One of the important relationships or trends used and shown by Feinmetall is the graph of spring force and CRES measured as a function of spring travel. The graphs are later combined in a diagram, showing the whole life cycle of the probe.

From the catalogue, the project has taken references from the parameters measured by Feinmetall as the viable electrical and mechanical properties to be measured for lifespan of ICT fixture probes. These are the contact resistance (CRES) and spring rate, which is a measure of spring force over spring travel. At the same time, the documentation has also helped in suggesting good correlations that can be done, especially with the probe cycle count of the fixtures.

While the measurement of the life cycle by Feinmetall is done in specific conditions highlighted above, it was later decided that the device will not require to keep the temperature, humidity, and dust-free conditions as conducted by Feinmetall. This is to reduce the difficulty and complexity in developing the product, as well as the increase in cost and time-to-lead in producing the end product. Regardless, the documentation provided by Feinmetall has proven to be critical in exploring the properties that need to be measured for lifespan of ICT fixture probes.

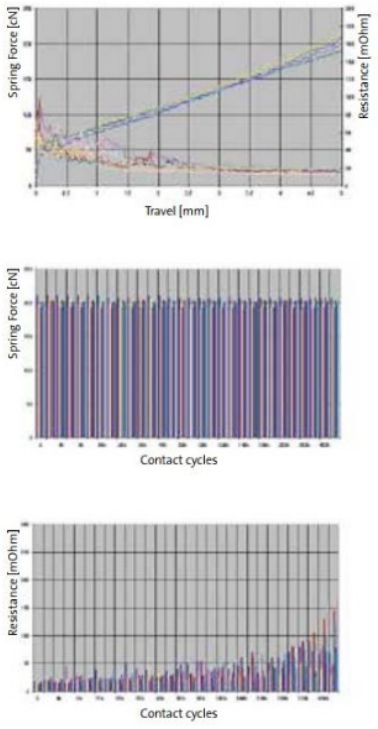


Figure 1: Trends that are used by Feinmetall to study life cycle of probes

4.2 Lessons In Electric Circuits, Volume I – DC, by Tony R. Kuphaldt

After understanding the necessary parameters that need to be measured by the device, it is required to know how to go about measuring these properties. The quantities that need to be measured are electrical resistance, force, and displacement.

When measuring electrical resistance, especially in low resolutions and sensitivity, it is sufficient to conduct measurement directly using probes and an electrical resistance measurement tool (such as an ohmmeter). However, when conducting the project for the CRES and Spring Rate Measurement Device, we find that the items to be measured (ICT fixture probes) come in very small values. The approximate range of electrical resistance of an ICT fixture probe is between 1 mΩ and 10,000 mΩ. Using conventional measurement techniques, such as using 2-wire measurement methods, will cause the electrical resistance measurement to include internal resistance of the measurement wires. This introduces additional errors to the measurement of the probes.

As such, it is important to explore and study the 4-wire Kelvin measurement, in order to enable a more accurate and consistent measurement for small values of electrical resistance. The book, 'Lessons In Electric Circuit, Volume I – DC', touches on many different aspects of DC circuit theory and measurement. Our focus for the project is with the 4-wire Kelvin measurement, the theory behind the configuration, as well as how to set up the circuit configuration for the CRES and Spring Rate Measurement Device.

From the book, it is found that at small values of electrical resistances, the internal resistance of the measurement probes and wires also contribute to the overall electrical resistance measurement of the ICT fixture probes. By using the 4-wire Kelvin measurement, the voltage measurement of the ICT fixture probe is done with the voltmeter as close as possible to the fixture probe. This is to ensure only the voltage across the fixture probe is taken in the measurement, without including the voltage drops on the measurement wires. The diagram below shows the 4-wire Kelvin measurement configuration, as well as the formula related to the measurement of electrical resistance using this configuration.

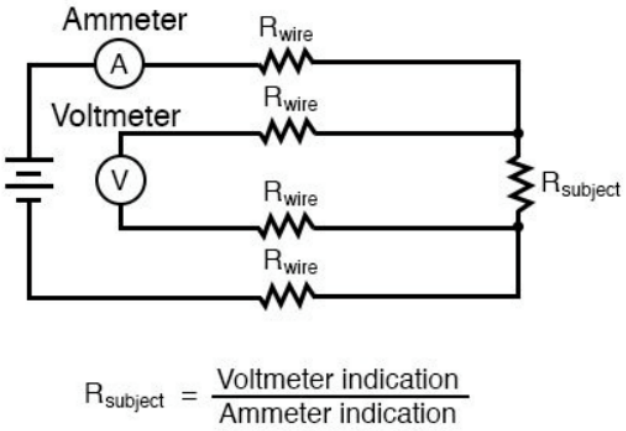


Figure 2: 4-wire Kelvin measurement configuration, with equation of electrical resistance

It is later found that the 4-wire Kelvin measurement is a setting that is available on high resolution source measure units (SMUs), such as those used within the factory. The user of the SMU would only need to use the device along with 4 wires, while enabling the settings, in order to conduct a 4-wire Kelvin measurement. Having said that, understanding the circuit configuration of 4-wire Kelvin Measurement, as well as the theory and operating principles behind the configuration, helps in understanding the criteria that should be taken into consideration during the development of the CRES and Spring Rate Measurement Device.

4.3 Are scripting languages any good? A validation of Perl, Python, Rexx, and Tcl against C, C++, and Java

Part of the exploration of tools and alternative solutions for developing the data collection tools involves validating which programming languages and tools are to be used to execute the project. Some of the possible programming languages that were thought of for the project include Perl, Python and Visual Basic for Applications (VBA). In order to confirm which language is to be used, some studying and research is needed to weigh the pros and cons of each language.

The document, 'Are scripting languages any good? A validation of Perl, Python, Rexx, and Tcl against C, C++, and Java', which is a chapter of a book ('Advances in Computers,

Volume 58') describes the different programming languages that are used for scripting. The history of the founding and development of the programming languages are explained in the document, along with the empirical study conducted on each of the programming languages. The results are explained in the chapter, with the measurements done on the execution time, memory consumption, program length and amount of commenting, program reliability, and programmer work time.

While the results of the programming performance does not show Python to be a more formidable programming language to be used for scripting, as compared to languages like C++, Java, and Perl, Python has more flexibility in terms of the available libraries and tools that are available. At the same time, Python has proven to be much simpler in syntax compared to the other languages. From the document's empirical study conducted on the scripting languages, the differences between the execution performance of Python and other programming languages are minimal. Since the execution time and performance for the data collection tools is not critical, it is therefore preferred to use Python as the programming language, since the simpler syntax and large range of libraries allow for faster programming time and project completion.

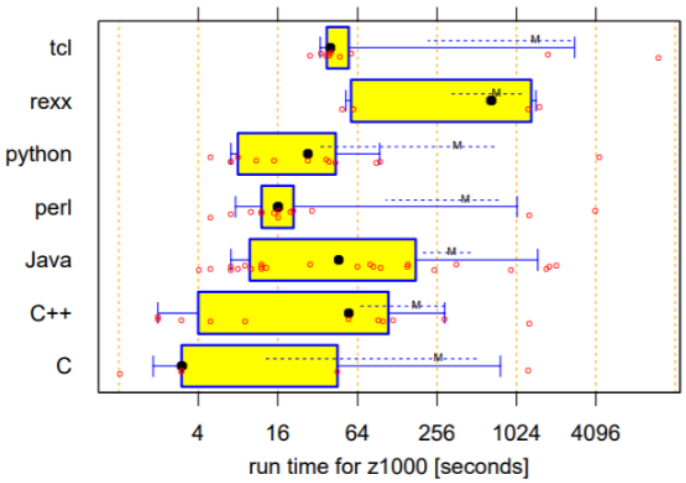


Figure 3: Execution Time in seconds on the z1000 dataset for different scripting languages

4.4 pandas: a Foundational Python Library for Data Analysis and Statistics

When building the data collection tools, the fundamental task that all the tools work on is data handling. Since the programming language used for the development of the data collection tools is Python, it is necessary to seek and understand the Python library needed for handling data.

The *pandas* Python library is a well-known and powerful tool used for handling data, building DataFrames from lists of data, generating output CSV and HTML files, and many more. In order to adequately understand the library, for the purpose of developing the software tools, the conference paper 'pandas: a Foundational Python Library for Data Analysis and Statistics' has helped in explaining the available tools and functions that can be used in *pandas* for the project.

The conference paper illustrates some real-life scenarios, namely in data analysis and statistics, that require *pandas* to operate. The majority of the conference paper talks about the available functions within the *pandas* library. These functions range from the basic functions that enable the creation of DataFrames (e.g., pandas.DataFrame), as well as more complex data handling and manipulation functions, such as data grouping, data sorting, and pivoting.

While most of the project execution relies on referring to the official documentations of *pandas*, as well as forums such as Stack Overflow, the referencing and reading of the conference paper has greatly helped in understanding the library that is being used.

4.5 A brief demonstration of some Python GUI libraries

In order to create more visually appealing and easy-to-use software for ICT, Graphical User Interfaces (GUIs) are incorporated into the project. Within the data collection tools, the GUIs are mainly implemented in the uploading of images, as well as input of probe ID and failure modes. Using a GUI for the software allows the users to easily upload files and images that are needed for reporting, as well as providing action buttons that help in navigating and editing the data tables that have been created using the software. Python has many GUI libraries that are available for development needs.

During the project execution, a number of GUI libraries have been explored and studied, such as *tkinter*, *PySimpleGUI*, and *Kivy*. From the research paper, 'A brief demonstration of some Python GUI libraries', it is possible to see the available functions and widgets for different GUI libraries. The GUI libraries that are explored by the author includes *tkinter*, *PySimpleGUI*, *Flexx*, and *Ipywidgets*. The author has demonstrated the various functions of each GUI library. Some GUI related applications were also demonstrated, such as visualisation of equations.

After studying the document, it is decided that *PySimpleGUI* is the best tool to be used for the purpose of the project. One of the main reasons for selecting *PySimpleGUI* is due to its function-based syntax, which relies on widgets and infinite loops to operate. This is different from other GUI libraries, which implement an object-oriented programming (OOP) approach in building the GUI windows. *PySimpleGUI* is also more flexible for drawing out the menus and text needed in the GUI window. The library has a powerful technique to loop widgets on the window, which allows the reading of text and questions from separate text files (implemented in the Probe Replacement Tracking Tool).

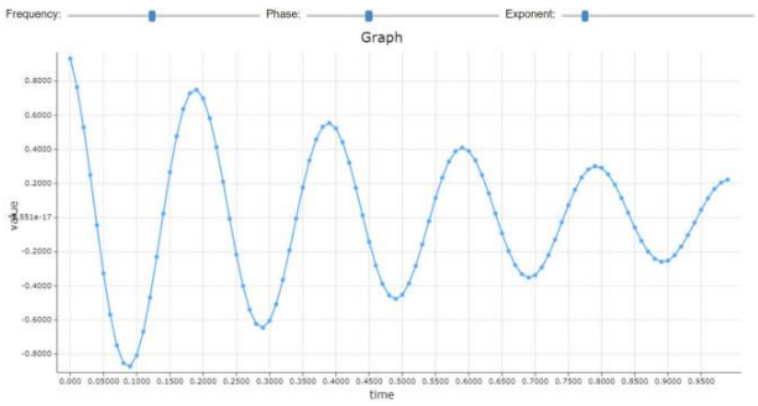


Figure 4: Damped Sinusoid Example using Flexx

5.0 METHODOLOGY

5.1 Methods and Tools

i. ICT Fixture Preventive Maintenance (PM) Report Generation Tool with Probe Cycle Count Pareto Chart

In order to collect the hardware cycle count (probe cycle count displayed on the fixture), it is necessary to collect the data from the reports produced during ICT fixture Level 1 PM activity. During this PM, the probe cycle counts displayed on the fixture are recorded by the PM crew in the report. Before the project is implemented in production, PM reports are either emailed, kept in local computers or printed and stored in cabinets. This makes it difficult to keep track of PM activities and the reports, as well as to gather the probe cycle counts for each ICT fixture running in production. As such, a report generation tool, equipped with a probe cycle count collection and pareto chart is developed and deployed for ICT fixture PM activities.

ii. Probe Cycle Count Harvesting Script

Aside from the probe cycle count that is displayed on the fixture (hardware probe cycle count), the probe cycle count is also recorded and stored within the product folder in the tester PC. This probe cycle count, known as the software probe cycle count, is stored in files which either are located within the individual product folder, or a separate counter file folder. These files are generated by the test programme that is run by the ICT to test the boards. Whenever a board is loaded onto the ICT fixture and ICT machine to be tested, the probe cycle count is recorded, and a file is generated. In order to gather the probe cycle count from these files, the current condition is to manually open the counter files and record the integer values within the file. As such, a Probe Cycle Count Harvesting Script is developed to retrieve the software probe cycle count from all the counter files within the tester PC.

iii. Probe Replacement Tracking Tool

When there are repeated test failures or NDF occurring, the first approach is to check if there are contact issues related to the testing. These are hinted by certain phenomena, such as high resistance failures and low capacitance failures. As a troubleshooting step, the common

approach is to replace the probes that have detected the failure, as these probes are suspected to be failing or are damaged. While this may solve the issue of repeated failures or NDF, there is lack of traceability or data collection done related to the probe replacement. In other words, there is no way of knowing what factors provide impact to a probe failing. Some of the factors or data features that are not tracked include the probe ID, failure modes, probe specifications, probe locations, and node names that are being measured by the probe. With the development of the Probe Replacement Tracking Tool, it is possible to track the probe replacement, while generating the necessary engineering data to help in investigation for root cause of repeated failures or NDF, as well as help in the study of probe lifespan.

iv. Contact Resistance (CRES) and Spring Rate Measurement Device

The probe cycle counts serve as an estimate for the measurement of probe lifespan and probe health. Rather than relying on the probe cycle count, it is best to measure electrical and mechanical properties of the ICT fixture probe in order to determine the health of the probes. Some of the parameters or properties that are measured by organisations and institutes outside are the contact resistance (CRES) and spring rate (spring force divided by the spring travel) of the ICT fixture probe. After exploration and designing, a CRES and Spring Rate Measurement Device is proposed to enable this capability for measuring the probe health and lifespan.

5.1.1 Tools

a. Python programming language

Python was used to develop the report generation tool and probe cycle count harvesting script. Python is a powerful scripting language used widely in software development. Python is heavily employed to build automation tools and various data handling workloads.

Figure 5: Python Programming Language logo

Python was selected as the programming language for this project due to its strength in string and file handling. While building a report generation tool and scripts, a lot of the tasks involved creating directories, generating output files, and converting files into different formats. Python allows the developer to easily "walk" through directories and accomplish file handling tasks for the project.

Furthermore, an important feature in the project is the ability to collect probe cycle counts recorded by the PM crew and produce a pareto chart of the probe cycle counts. This task involves storing the data into a database or comma-separated values (CSV) file. While Python is well-known for data science and machine learning applications, the handling of data and CSV files, as well as generation of plots and graphs from the data collected, makes Python the most-preferred tool to develop the project.

A very powerful quality of the Python programming language is the ability for developers to import various libraries that are required for the software. There is an extremely large option for libraries available for the development of the project. Some of the libraries used during the development of the project include libraries for data handling, graph plotting, graphical user interface (GUI) and many more.

For the project, the Python version used is version 3.6.8. In Python development, there are two major distinctions for Python versions, which are Python 2 and Python 3. Python 2 is the older version, which has an older and slightly different syntax compared to Python 3. The tasks handled by the report generation tool and probe cycle count harvesting script is relatively advanced and requires modern and more powerful libraries, which are not supported in Python 2. Furthermore, Python 2 is no longer supported as of 1 January 2020. As such, Python 3 is used for the project.

The latest version of Python at the time of writing is 3.9.0. As the project has required the use of many different libraries, some newer and some older than others, there have been instances whereby the libraries have malfunctioned for newer versions of Python (e.g, *matplotlib* has caused bugs when using Python 3.7 during development). After numerous experimentation and trial-and-error, it has been found that Python 3.6.8 has worked the best for the project, being compatible with all the libraries used, as well as being modern enough to support latest tools required.

b. *pip* package-management system

Majority, if not all, of the Python libraries used in the project have been obtained and installed from the *pip* repository. *pip* is the go-to package-management system for installing libraries and software packages used in Python. *pip* contains installation files needed to run the Python libraries, as well as links redirecting to the project homepage and documentations for the libraries. Aside from installing the software directly from the *pip* website (https://pypi.org/project/pip/), *pip* can also be called as a command in the Command Prompt or Terminal Window.

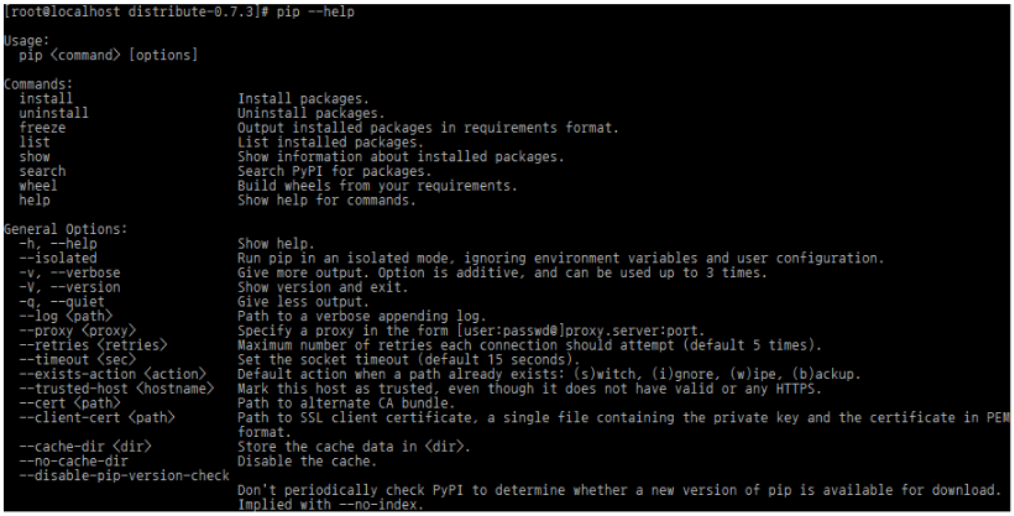


Figure 6: Example of help menu from pip and execution on Terminal window

pip is pre-installed and readily available along with the Python installation. Once Python has been installed on the PC, running *pip* commands is sufficient to install all the necessary libraries for the report generation tool and probe cycle count harvesting script.

c. Anaconda distribution *(IDE and virtual environment)*

Anaconda is an open-source distribution of Python and R programming languages. Anaconda distribution is mainly used for scientific computing and data science applications. Two of the

important tools within the Anaconda distribution used for this project are the Spyder Integrated Development Environment (IDE) and the Anaconda environment.

Figure 7: Anaconda Distribution Logo

Spyder IDE is an open-source integrated development environment heavily used for scientific computing in Python. Spyder IDE can be easily configured to run both Python 2 and Python 3, whichever preferred by the software developer. A powerful tool that has been used heavily in Spyder is the Variable Explorer. Variable Explorer allows the developer to trace the code execution line-by-line, observing the step-by-step changes made in the code variables. This is a useful tool for debugging and tracing loops and recursions when developing the project.

Figure 8: Spyder IDE logo

Another important tool from Anaconda distribution is the Anaconda environment. While it is possible to install Python libraries natively in the PC, using a virtual environment allows for easier handling of the libraries, while giving the option for the developer to do a fresh installation of libraries needed for a project, by creating a new environment. By developing the project within an Anaconda environment, it made it easier to upgrade and downgrade libraries when bugs occurred (especially for the matplotlib-Python incompatibility

incident), as well as to make a clean installation of Python and libraries by creating new environments for different projects.

d. Git and Github *(version-control system)*

Figure 9: GitHub logo

Figure 10: Git logo

Git is a version-control system used to track software versions throughout the software development. Github provides hosting for software development and version-control using Git. When developing the report generation tool and probe cycle count harvesting script, version-control is important to allow development on different machines, while also allowing version reverting when a bug occurs in a build.

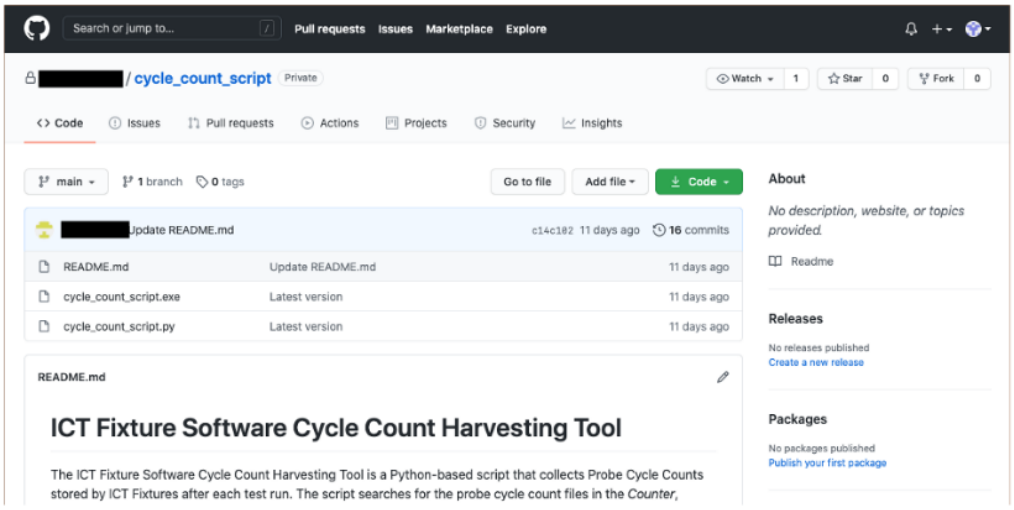


Figure 11: GitHub Repository for Probe Cycle Count Harvesting Script

While it is common practice in the industry or software development to create development branches and making pull requests to merge the updates with the main branch, the software projects were developed mainly by an individual. As such, the software revisions and updates are pushed directly to the main branch.

e. *pandas* library *(data handling and output file generation)*

One of the most heavily used libraries for the project is *pandas*. *pandas* allows developers to handle datasets in Python. Some of the data handling capabilities provided by *pandas* are creating DataFrames from lists or arrays, merging, joining, or appending data onto existing DataFrames, as well as generating output files from DataFrames, such as CSV and HTML files. The capabilities provided by *pandas* has enabled the collection of data for the report generation tool, gathering of both hardware and software cycle count values, as well as producing CSV files and HTML reports from the collected data.

Figure 12: pandas logo

At the time of writing, the latest version of *pandas* is version 1.1.4. However, the version used to develop the project is version 1.1.3.

f. PySimpleGUI *(GUI tool)*

Graphical User Interfaces (GUIs) are used a lot within the data collection tools. The implementation of GUIs helps in creating a more visually appealing and easy-to-use software for ICT. By implementing GUIs in the software, the concepts of UI/UX design can be implemented accordingly.

Figure 13: PySimpleGUI logo

While there are a number of GUI libraries available from Python, such as *tkinter* and *Kivy*, *PySimpleGUI* is selected out of all the GUI libraries available. The reason for selecting *PySimpleGUI* is due to its simplicity in implementation. The library creates its GUI window through widgets and infinite loops. This is different from other GUI libraries, which implement an object-oriented programming (OOP) approach.

Another benefit and reasons for using *PySimpleGUI* is its ability to be applied within loops to display widgets, menus and text. This allows the script to dynamically read from external text files and display the menus and widgets accordingly in the GUI window. This is particularly implemented in the Probe Replacement Tracking Tool for reading the failure modes from a text file.

g. Pyinstaller *(generates executable file from Python scripts)*

Since the tools are developed using Python and the generated scripts that run the software are Python scripts, it is only possible to execute the scripts by running the main script using Python. In other words, it is necessary for the PC that runs the scripts to have an installation of Python.

However, the target PCs that will be running the tools do not have Python installed natively. It is therefore not possible and not practical to install Python for every target user that will be using the tools. As such, the Python scripts are converted into executable files (.exe files) in order to run successfully on all targeted PCs.

The library used for this task is *Pyinstaller*. Using *Pyinstaller*, it is possible to process a Python script and generate an executable file from the scripts. The library also allows addition of external software and tools that are required by the script. This is particularly important for the conversion of HTML files to PDF files for the data collection tools, which make use of the *wkhtmltopdf* tool.

It is noted, however, that the targeted PCs are running Windows operating system. As such, the *Pyinstaller* executable file generation is done to produce a Windows executable file. Should the script be running on an operating system aside from Windows, the tool will not be able to operate accordingly. This has been consulted with the project team and found to be not of major concern, since the software will be mainly used on Intel devices and tester PC, which are running Windows.

5.1.2 Methods

5.1.2.1 ICT Fixture Preventive Maintenance (PM) Report Generation Tool

When developing the report generation tool, there are two categories of the software, each used for one level of PM. The software is categorised into 'ICT Fixture Level 1 PM Report Generation Tool' and 'ICT Fixture Level 2 PM Report Generation Tool'. The key difference between Level 1 and Level 2 PM in terms of the tool's capabilities is the collection of hardware cycle count, which is only done in Level 1 PM.

The ICT Fixture Preventive Maintenance (PM) Report Generation Tool is a Python-based script that allows Preventive Maintenance (PM) crew to generate reports upon conducting ICT Fixture PMs. The tool allows uploading of images (Before and After conducting PM), form to enter report data, and generates CSV, HTML and PDF of the report. The hardware cycle count of each ICT fixture provided in the report after PM activity is harvested into a separate CSV, and a Pareto Chart of Top Probe Cycle Counts is produced, indicating to the PM Crew on which fixtures require reprobing (probe cycle count exceeding the cycle count limit).

The report generation tool is made up of 4 individual Python scripts, the execution follows this order:

1. menu.py
2. images.py
3. dataframe.py
4. cyclecount_pareto.py

Report Form (menu.py)

The program starts with the Terminal displaying description of the program. The description is made up of the software title, the developer of the software, and the date the software was developed. A large ASCII art text of the software title is displayed using the *art* Python library.

The form starts with requesting the user to input their employee ID, the name of the fixture that had PM conducted on, and the fixture ID.

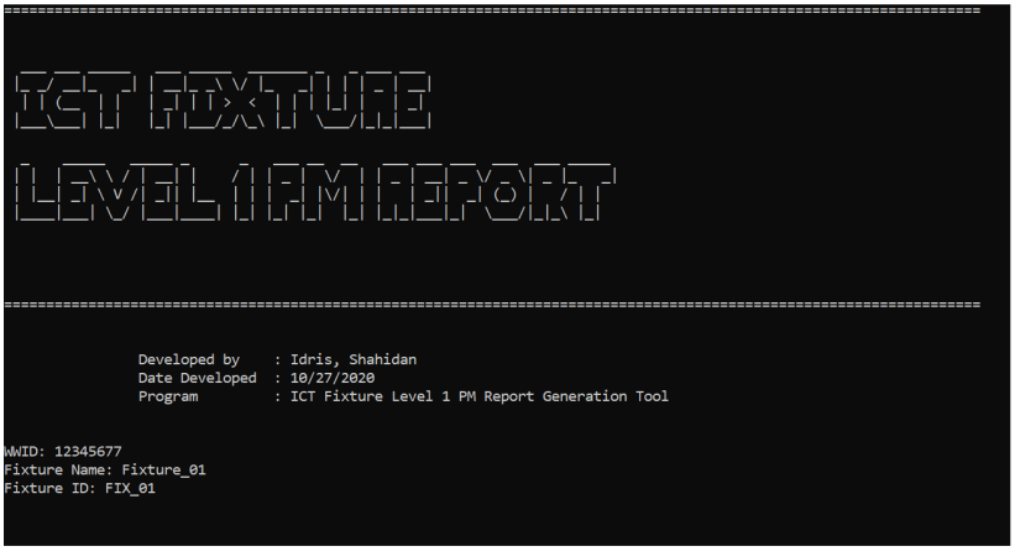
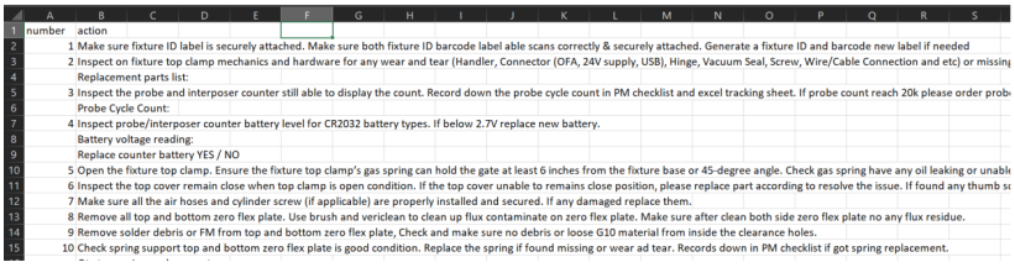


Figure 14: First page of Report Form requesting for employee ID, Fixture Name and Fixture ID

Reading Steps from Excel file

Using the *xlrd* Python library, the program reads from an Excel file named 'steps_Level1.xlsx'. This Excel file contains the steps of conducting Level 1 PM on the ICT fixture, as set by the factory specifications. By allowing the program to read and retrieve the steps from an Excel file, this makes it easier for the tool owner to make amendments to the existing steps and specifications, as well as update the report generation tool whenever there are changes to the steps, without needing to reopen and edit Python code.



	number	action
2	1	Make sure fixture ID label is securely attached. Make sure both fixture ID barcode label able scans correctly & securely attached. Generate a fixture ID and barcode new label if needed
3	2	Inspect on fixture top clamp mechanics and hardware for any wear and tear (Handler, Connector (OFA, 24V supply, USB), Hinge, Vacuum Seal, Screw, Wire/Cable Connection and etc) or missing
4		Replacement parts list:
5	3	Inspect the probe and interposer counter still able to display the count. Record down the probe cycle count in PM checklist and excel tracking sheet. If probe count reach 20k please order prob
6		Probe Cycle Count:
7	4	Inspect probe/interposer counter battery level for CR2032 battery types. If below 2.7V replace new battery.
8		Battery voltage reading:
9		Replace counter battery YES / NO
10	5	Open the fixture top clamp. Ensure the fixture top clamp's gas spring can hold the gate at least 6 inches from the fixture base or 45-degree angle. Check gas spring have any oil leaking or unabl
11	6	Inspect the top cover remain close when top clamp is open condition. If the top cover unable to remains close position, please replace part according to resolve the issue. If found any thumb s
12	7	Make sure all the air hoses and cylinder screw (if applicable) are properly installed and secured. If any damaged replace them.
13	8	Remove all top and bottom zero flex plate. Use brush and vericlean to clean up flux contaminate on zero flex plate. Make sure after clean both side zero flex plate no any flux residue.
14	9	Remove solder debris or FM from top and bottom zero flex plate, Check and make sure no debris or loose G10 material from inside the clearance holes.
15	10	Check spring support top and bottom zero flex plate is good condition. Replace the spring if found missing or wear ad tear. Records down in PM checklist if got spring replacement.

Figure 15: Excel File containing steps and questions for conducting PM activity

After reading the steps from the Excel file, the program displays the questions one-by-one to the reporter. The reporter, or user of the program, needs to hit 'Enter' or enter text input wherever the steps require. The program stores the questions or steps of the PM, as well as the input entered by the reporter into lists.

Figure 16: Report Form executing according to Excel File provided steps and questions

At the end of the form, the program executes the image retrieval tool, which runs the graphical user interface (GUI).

Image Retrieval Tool (images.py)

On the Image Retrieval GUI (*images.py*), PM crew will upload images of the PM activity conducted. The images are uploaded according to these categories, sourced from historical ICT Fixture PM reports in the factory:

- Top Cover (1) Before
- Top Cover (1) After
- Top Cover (2) Before
- Top Cover (2) After
- Bottom Cover (1) Before
- Bottom Cover (1) After
- Bottom Cover (2) Before
- Bottom Cover (2) After

Not all PM activities would have all 8 images to be presented. Users would only need to upload whichever images that are applicable to the PM activity conducted.

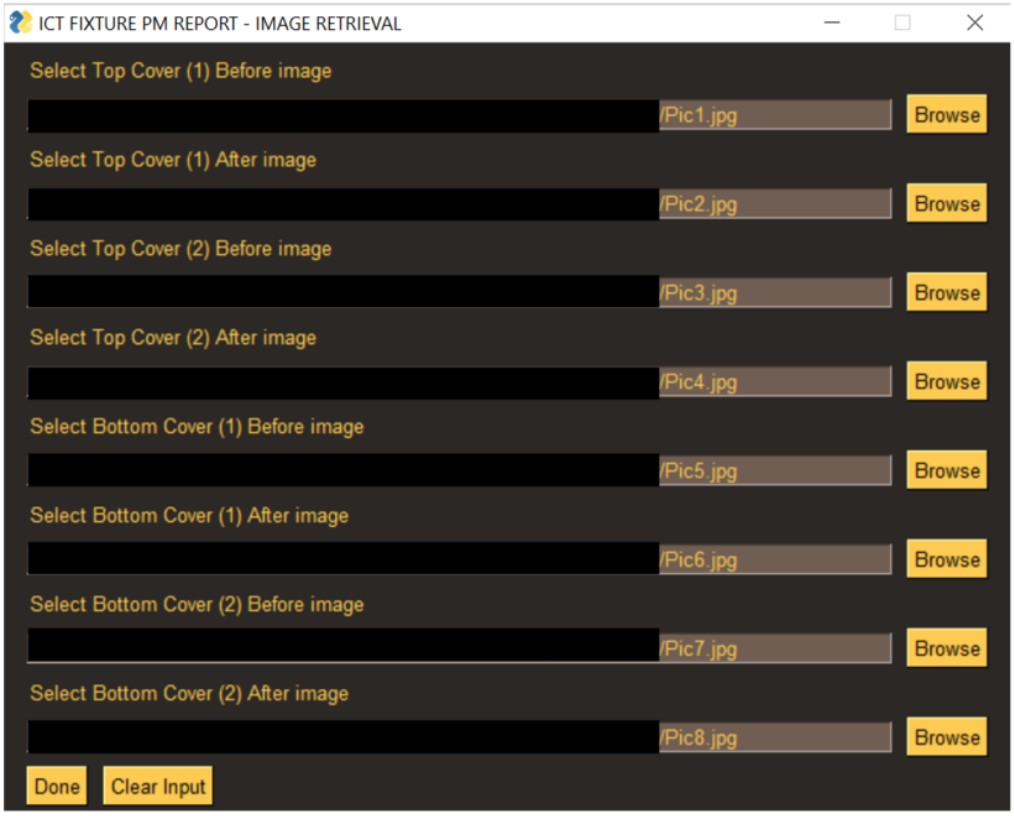


Figure 17: GUI to collect Images developed using PySimpleGUI

The GUI was built using the *PySimpleGUI* library. Different from other GUI tools that are developed using an object-oriented programming (OOP) approach, such as Kivy and Tkinter, *PySimpleGUI* relies on straightforward functions and loops to run the GUI.

The GUI starts with loading the selected colour theme for the GUI, which is 'Dark Amber'. The layout is then build using a number of widgets, made up of functions. Some of these widgets include Text, Input, and FileBrowse. At the end of the GUI, 'Done' and 'Clear Input' buttons are added for added functionality.

If the user selects 'Clear Input', the inputs and images uploaded onto the GUI is cleared, and users can reupload again. Once the images are uploaded and confirmed by the user, the 'Done' button terminates the GUI (by breaking the infinite loop running the GUI window). A new list is created and added with the file locations of the images uploaded by the user. This list is added onto the answer list previously created in *menu.py*.

The program then executes *dataframe.py*, which runs the data handling and report output file generation.

DataFrame, Report Output Generation, Probe Cycle Count Harvesting (dataframe.py)

The next script that is run within *menu.py* is *dataframe.py*. This part of the script collects the list of questions and answers from the Report Tool and creates a DataFrame. From the DataFrame, a CSV and HTML file is generated to show the report.

The program starts with creating a *pandas* DataFrame, using the lists of questions and answers created and populated from *menu.py* and *images.py*. The program drops the last 8 data entries (which are the image captions and image file locations) from the DataFrame, as these entries do not need to be shown in the report table. As the user or reporter may provide empty entries, or perhaps '0', indicating no items or quantity of items, the program converts empty entries and 0 into 'None'.

The program creates a folder named as the fixture ID provided earlier, if it is not previously created. This folder is used to store the CSV, HTML, and PDF report files. A CSV file is then generated, named with the fixture ID and the date of the report being generated.

Using HTML and CSS, a title and header is written for the HTML report. For example, the header for the Level 1 PM report is 'ICT Fixture Level 1 PM Report for <Fixture ID>'. The DataFrame is dropped with the first 4 data entries, which are the employee ID, fixture name, fixture ID, and date reported. The DataFrame is then printed into the HTML file in the form of a table.

ICT Fixture LEVEL 1 PM Report for Fixture_01 (FIX_01)

Reported by: 12345677
Fixture Name: Fixture_01
Fixture ID: FIX_01
Date Reported: 2020-12-06

Items	Contents
1. Make sure fixture ID label is securely attached. Make sure both fixture ID barcode label able scans correctly & securely attached. Generate a fixture ID and barcode new label if needed	Done
2. Inspect on fixture top clamp mechanics and hardware for any wear and tear (Handler, Connector (OFA, 24V supply, USB), Hinge, Vacuum Seal, Screw, Wire/Cable Connection and etc) or missing parts replace according. Record down in PM checklist.	Done
Replacement parts list:	None
3. Inspect the probe and interposer counter still able to display the count. Record down the probe cycle count in PM checklist and excel tracking sheet. If probe count reach 20k please order probe, follow the probe part number and qty at probe part list attach on top of fixture. Plan schedule for re-probe activity	Done
Probe Cycle Count:	20000
4. Inspect probe/interposer counter battery level for CR2032 battery types. If below 2.7V replace new battery.	Done
Battery voltage reading:	None
Replace counter battery YES / NO	None

Figure 18: Sample Output of Generated ICT Fixture Report

At the end of the HTML file, the image files, designated by the file locations provided in the GUI earlier, are encoded in Base64 and rendered as images in the HTML report. After printing each image, the respective caption is printed and centre-aligned (using CSS).

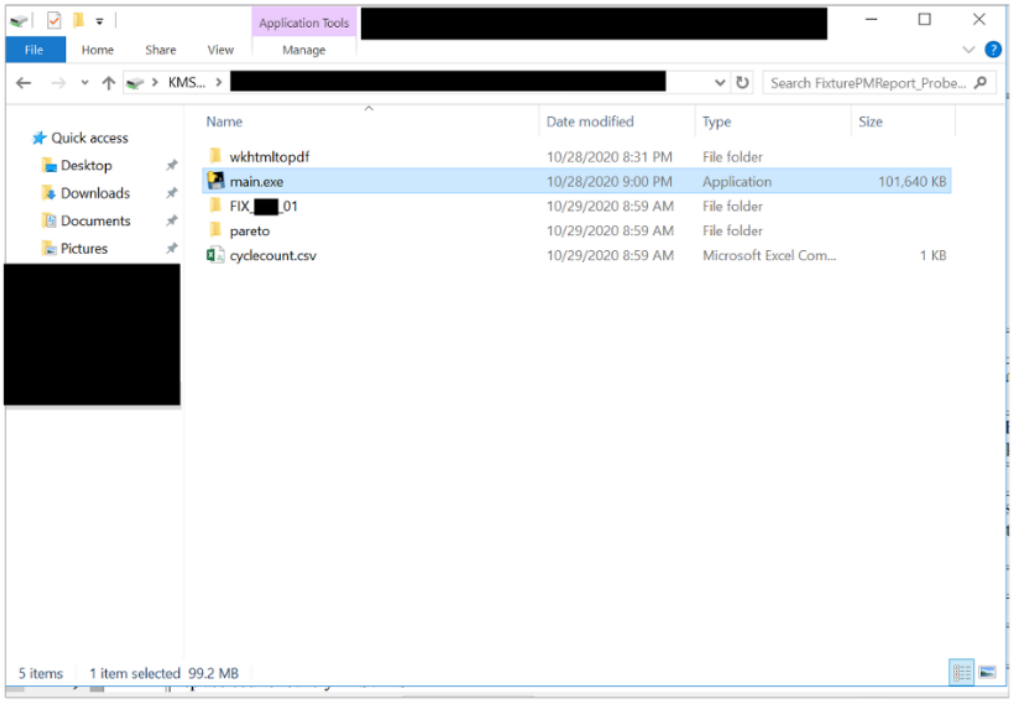


Figure 19: Generated folders and files – Fixture ID folder, Probe Cycle Count CSV and Pareto folder

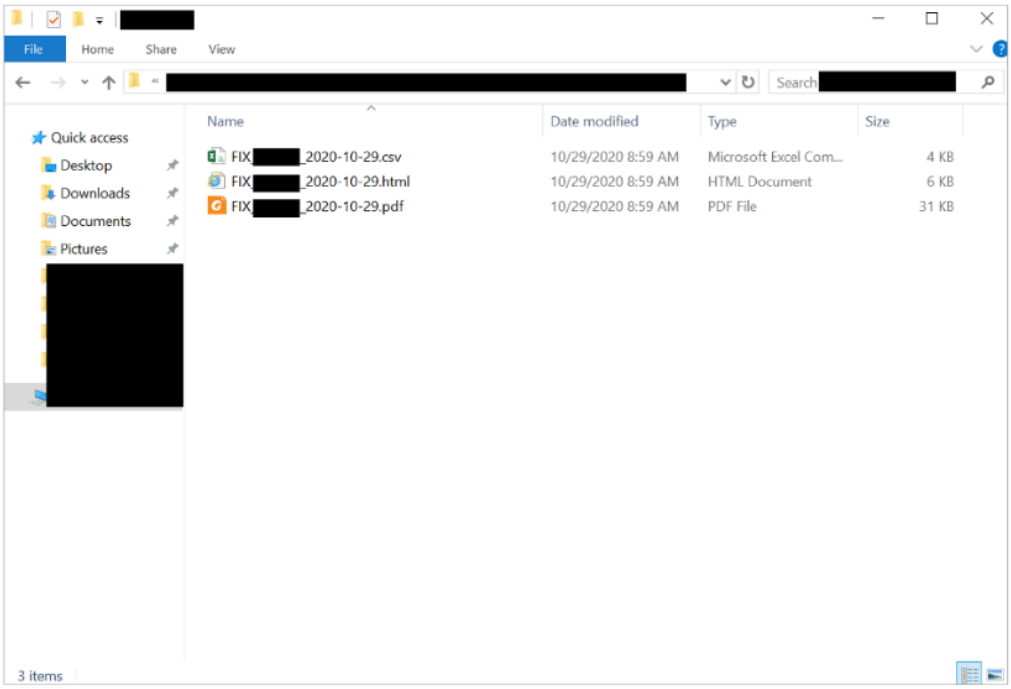


Figure 20: Report Data CSV file, HTML and PDF reports generated in Fixture ID folder

Using the *pdfkit* Python library, which makes use of a program named *wkhtmltopdf*, the program converts the HTML report generated earlier into a PDF. The program opens the PDF file to be shown to the user or reporter.

For the Level 1 PM, the final part of the *dataframe.py* script collects the reported Probe Cycle Count and stores the data into a separate CSV file using a DataFrame. The DataFrame data is built using timestamp of the report generated, fixture name, fixture ID, and the Probe Cycle Count. The program then creates a CSV file from the DataFrame.

This CSV file is later read and processed to generate the Pareto Chart. As for Level 2 PM, the hardware cycle count recorded is not collected into the CSV as the procedures for Level 2 PM only involve collecting the probe cycle counts before the reprobing is conducted. This data is not very useful to us.

Pareto Chart (cyclecount_pareto.py)

The program then calls *cyclecount_pareto.py*. This script reads the CSV file containing the reported Probe Cycle Counts of each fixture and produces a Pareto Chart. The Pareto Chart shows the top Probe Cycle Counts for all fixtures in the CSV file (top 10 if there are more than 10 fixtures recorded). Within the chart, there are limit lines indicating the PM Crew needs to start purchasing new probes, and when reprobing needs to be conducted.

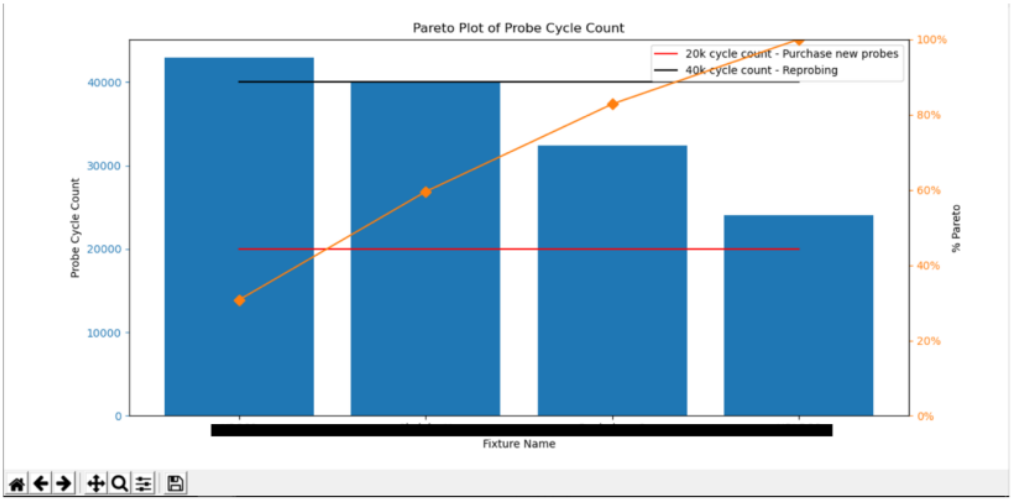


Figure 21: Pareto Plot of Top Hardware Cycle Counts retrieved from Level 1 PM reports

The Pareto Charts generated from this script are saved as a PDF and stored in a folder named ‘pareto’, which is generated if it does not exist.

5.1.2.2 Probe Cycle Count Harvesting Script

The Probe Cycle Count Harvesting Script is a Python-based script that collects Probe Cycle Counts stored by ICT Fixtures after each test run. The script searches for the probe cycle count files in the 'Counter' and all board folders, retrieves the integer value contents within the file and stores the data into a *pandas* DataFrame. A CSV and HTML file reporting the DataFrame is generated.

When running the executable file, the script first looks into the Counter folder. This folder contains cycle count files for newer boards or boards with updated testplan (to generate cycle count files in Counter folder instead of respective board folders). The cycle count files contain an integer of the probe cycle count. The file name and the contents are stored in lists.

Next, the script looks into the board folders. The script looks for files starting with "fix_cntr_", which indicates the cycle count file. Once the file is found, the full path and contents of the file are stored in lists.

The lists are added into a *pandas* DataFrame. The script checks if there is an existing 'cycleCount.csv' file. If the file does not exist, a new CSV file is generated with the newly gathered data. If the file exists, the old data is read into a DataFrame, the new data is joined with the old data, and is then generated into the CSV file.

A HTML file is generated with some HTML formatting (for title of the HTML report) and appended with the DataFrame. The CSV and HTML files are generated at the 'SoftwareCycleCount' folder, which is created if it does not exist.

Software Cycle Count Report (2020-11-25)

BoardName	CycleCount_2020-11-09	CycleCount_2020-11-11	CycleCount_2020-11-18
[redacted]	7151	7696	9413
	8466	8765	9620
	2352	2457	2543
	2853	2874	2970
	1000	1000	1000
	51048	51465	52602
	432	432	470
	5082	5082	5082
	29246	29449	29928
	29411	29427	29432
	17358	17825	18284
	23641	23641	23641
	30050	30050	30050
	80498	80502	80513
	14135	14172	14238
	29755	29755	29755
	34583	34583	34583

Figure 22: Generated HTML file with Table of Software Probe Cycle Counts from running tool

5.1.2.3 Probe Replacement Tracking Tool

While the ICT Fixture PM Report Generation Tool is able to collect data on the hardware cycle count of fixtures, and the Probe Cycle Count Harvesting Script gathers the software cycle count, it is important to understand the failure modes that are called when a probe is replaced. Aside from that, the data should also contain information on the physical specification of the probes replaced, such as the probe size, coordinates of the probe in the fixture, and the location of the probe (top or bottom of the fixture).

Aside from Level 2 PM, probes are also replaced during production, as a troubleshooting approach for repeated ICT test failures. Instead of replacing all the probes on the fixture, only the probes that are suspected to be causing the repeated failure are replaced. In order to study the health of ICT fixture probes, it is necessary to have a tool to track the probe replacements.

The Probe Replacement Tracking Tool is a Python-based software that allows technicians to record details of the probes replaced during production. The tool allows the user to key in the fixture ID in which the probes are being replaced, the probe ID of the probes replaced, and the failure modes causing the replacement. The tool searches the probe list, which is a document provided from the fixture manufacturer that describes each probe's ID, net name, probe size, and other information, to then populate the data recorded by the user. At the end, the tool will show information on the probe IDs being replaced for each fixture, as well as the failure modes that frequently cause probe replacements. All the data is collected into master CSV files for further engineering study.

The software is built from multiple Python scripts, which follow the execution order as below:

1. main.py
2. failure_parser.py
3. gui.py
4. table.py
5. query.py
6. graph.py

Main script (main.py)

The script starts by displaying the header of the program. The header contains the title of the tool, written in ASCII art using the *art* Python library. The following information on the header includes the developer name, the date in which the software is developed, and the title name.

The tool first asks for the employee ID of the user (in this case would be the technician recording the probe replacement). The tool then asks for the Fixture ID. The employee ID will be included in the data file later for reference, while the Fixture ID is used to find the correct probe list file and store the data files into the correctly organised folders.

The main script then calls the GUI menu, which is used to gather the data of the probes replaced by the user.

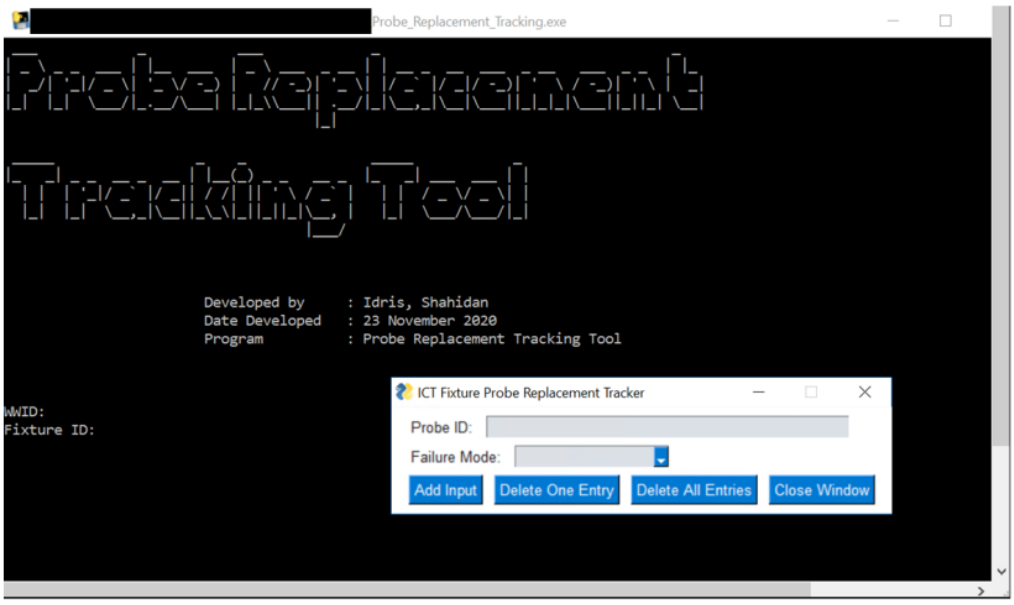


Figure 23: Menu requesting for employee ID and fixture ID, with GUI appearing requesting for probe replacement data

Failure Mode parser script (failure_parser.py)

Before the GUI menu appears, the tool will first execute the parser script to read the available failure modes that can be entered by the user. The parser script reads a text file, which contains the list of failure modes, and presents the list in a dropdown menu widget in the GUI tool. The reason for adopting this approach is to allow flexibility and room for expansion in the future for ICT. The tool owner or engineer can directly edit or add on to the failure mode list, whenever there are new test capabilities (and new failure modes) added to the factory's ICT.

The parser first looks for the text file containing the list of failure modes. The script then removes the whitespaces and linespacing between the full text string, followed by splitting the full text string into individual failure modes (split by linespacing between failure modes). The parser then returns the failure modes in a list to the GUI.

GUI menu (gui.py)

The GUI menu is built using the *PySimpleGUI* Python library. The library makes use of functions and infinite loops to run the GUI, rather than adopting an object-oriented programming (OOP) approach.

The GUI is made up of an InputText (standard text input) widget and a Combo (combo box) widget. The InputText will allow the user to enter the Probe ID for the probes being replaced. For each Probe ID, the user will select the Failure Mode associated with the replacement. In other words, each Probe ID will have its own Failure Mode, whereby the Failure Mode was parsed and retrieved from the text file by the failure_parser.py script.

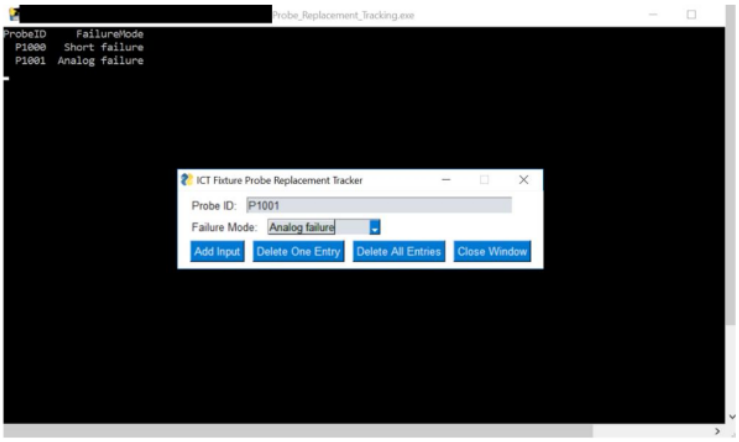


Figure 24: GUI menu requesting for Probe Replacement Data, with dynamic table updating according to input

There are a number of buttons at the end of the GUI, which are 'Add Input', 'Delete One Entry', 'Delete All Entries', and 'Close Window'. After entering the Probe ID and selecting the respective Failure Mode, the user selects 'Add Input' to add the entry to the GUI. The GUI processes the input into lists, which are fed to the table.py script for DataFrame building. 'Delete One Entry' allows the user to remove the latest Probe ID and Failure Mode input, in the case that the user mistakenly enters data. 'Delete All Entries' clears the data table, should the user find the need to do so. 'Close Window' is selected once the user has completed the data entry of all Probe ID and Failure Mode. The full data table is then built in table.py and returned to the main script.

The GUI works together with the table.py script. The input received from the GUI translates to table.py, where the DataFrames are built and displayed to the user each time a new entry is added.

DataFrame building (table.py)

This script has a number of functions that support the GUI. This script is able to build a table from the data input, delete one entry from the table, delete all entries from the table, as well as store and return the DataFrame back to the main script.

The GUI collects the inputs from the user in the form of lists. The lists are entered as arguments to the function for building the table (makeTable function). The makeTable function takes in the lists and builds a DataFrame using *pandas*. The DataFrame is then printed or displayed onto the Terminal window. Since the makeTable function is called within the GUI loop, the DataFrame is cleared and displayed each time a new input is added. This makes the table display to be dynamic for every input that the user enters.

The buttons used in the GUI has their functions tied to the table.py script. The function associated with 'Delete One Entry' button in the GUI is deleteOneEntry function, whereas the function associated with 'Delete All Entries' button in the GUI is deleteAllEntries function. The function, deleteOneEntry, removes the latest entry in the lists (that the user populates with data) provided by the GUI. The function, deleteAllEntries, clears the lists provided by the GUI.

Finally, once the GUI window is closed (either by directly closing the window or selecting the 'Close Window' button), the storeTable function is called, which builds the final DataFrame from the lists of data in the GUI. The function then returns the DataFrame to the GUI, which returns it back to the main script.

Populating the DataFrame with Associated Data (query.py)

The DataFrame generated and built using the GUI and table.py contains the Probe ID and Failure Mode of the replacements made. However, there is still a lack of information relating to these attributes. While the user or technician conducting the probe replacement has the Probe ID and Failure Mode information at hand during the activity, from an engineering perspective, it is necessary to know additional information, such as the probe size (width of the probe, usually comes in 39mil, 50mil, 75mil and so on), location of the probe (either at the top or bottom of the fixture), coordinates of the probe (x-y location of the probe within the fixture), net or node name associated with the probe (the node within the circuit of the board which the probe is connected to), the probe part number, and the probe BRC number. The information will help in studying the pattern of the failures, with regards to the parameters being measured by the probe (node name gives indication of voltage-current values and measurement type), as well as physical attributes such as probe size. Given the DataFrame containing Probe ID and Failure Mode, the query.py script will populate the existing table with the additional

information. The script obtains the information from the Probe List of each fixture (named according to their Fixture ID) mentioned previously.

The script starts with finding the folder containing the Probe List. Once the folder is located, the script runs through all the Probe List Excel files within the folder to find the Fixture ID associated with the probe replacement. Once the correct Probe List is found, the script then opens the file to read the contents.

The formatting of the Probe List files is different, depending on the supplier of the fixtures. In other words, each supplier produces a different style of the Probe List Excel file. Based on the formatting and column names provided in all the Probe List files that exist to date, the script is programmed to parse the column names correctly (for all supplier file formats) to retrieve the correct information needed.

Since the Probe List files are in Excel format, the *xlrd* Python library is used. The script reads the first sheet of the Excel file, then loops across the column names (row number 0, moving horizontally). While reading the column names, the script also parses the name strings by removing the whitespaces, underscores (_), front-slashes (/), hash symbols (#, indicates 'number') and lowercases the complete string. This is done to standardise the column names read by the script across different formats and styles of Probe List files. For each column name that the script is able to identify (e.g. 'probesize' is found within the Excel file), the script then loops vertically down the column (in other words, runs through all the existing probe sizes for that fixture). The script then appends all the data into a list. This process is repeated for all the necessary columns or attributes needed for the DataFrame population.

From the DataFrame built from the GUI, the Probe ID is read into a list. The script runs a loop and compares each Probe ID input from the GUI data to the Probe ID obtained from the Probe List file. Once a match in Probe ID is identified, the script collects the respective information (probe size, probe location, coordinates, and so on) for that Probe ID and collects the data into lists.

At the end of the script, the lists are appended as new columns to the DataFrame built from the GUI (which is added as an argument to the function of this script). The newly

populated DataFrame, containing the additional information relevant to engineering, is returned from the script.

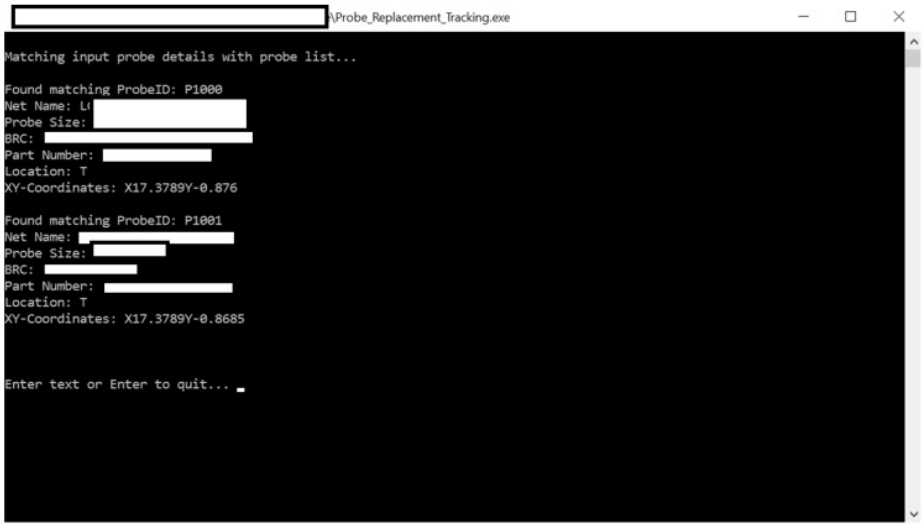


Figure 25: Terminal Display of matching probe ID with engineering data from Probe List

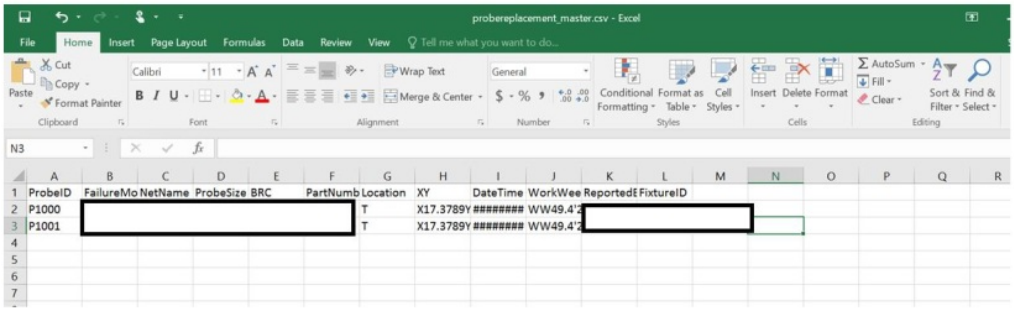


Figure 26: Master CSV file containing recorded Probe ID and respective engineering data

DateTime and WorkWeek Calculations, CSV File Output

Before setting the file organisation and generating the CSV files from the data, the DateTime (timestamp) of the probe replacement activity is appended to the DataFrame. Aside from the timestamp, a work week calculation is added, and the resulting work week is appended to the

DataFrame. As timelines are mostly communicated in work weeks at the factory, it is useful to include the work week automatically for easier referencing.

Work weeks at Intel start on Monday (WWx.1) and end on Sunday (WWx.7). The general rule of how work weeks are calculated is that the work week starts at 1 for the beginning of each year. This means that the first week of January is WW1, and the week of 30 November 2020 to 6 December 2020 is WW49. It is not always the case that the last week of the year is the last work week of the year. For the case of the year 2020, the dates 30 – 31 December 2019 are considered WW1 of 2020.

With the general rule and various exceptions of how work weeks are calculated, an algorithm for the calculation is built to summarise and simplify the complete generation of work weeks, given the date. The algorithm is as follows:

1. Given the date, with components *todayyear* (the year), *todaymonth* (the month), and *todayday* (the date number, e.g. 30 November, 30 is the *todayday*).
2. *lastday* is the day of the week for the last day of the year (i.e. 31 December of *todayyear*). From the Python code, day of the week starts as 0 for Monday, 1 for Tuesday, and so on. Therefore, *lastday* and all day of the week values generated are added by 1 (so that Monday is 1, Tuesday is 2, and so on, which matches the calculations at the factory).
3. *thisnewyearww* is the work week of this year's New Year's Day (i.e. the work week of 1 January of *todayyear*). Work week in Python code is calculated as 0 for the first week of the year. As such, this needs to be added by 1, to ensure the first work week is 1. However, this is not always the case. If the first work week has a Monday within the new year, the work week is considered as 1, instead of 0. Therefore, the addition of 1 to the work week is conditional.
4. *newyearday* is the day of the week for *nextyear*'s (*todayyear + 1*) New Year's Day. This number is added with 1 to accurately indicate the days of the week (Monday as 1, Tuesday as 2, and so on).
5. If *lastday* is 7 (Sunday), set *lastday* as 0 (this is for calculation purposes).
6. If *newyearday* is 7 (meaning falling on Sunday), get *workweek*, or the work week of the date (using *todayyear*, *todaymonth,* and *todayday*), using the built-in Python

code. If *thisnewyearww* is 0 (due to first week not having a Monday), add *workweek* by 1.

7. If *newyearday* is not 7 (not Sunday), check if *todaymonth* is 12 and that *todayday* is more than or equal to (31 – *lastday*). The work week calculation is conditional for December, and the condition for whether the last few days of December should be WW1 or WWx (where x is the last work week of the year), depends on the (31 – *lastday*) calculation.
    a. If *todaymonth* is 12 and *todayday* is more than or equal to (31 – *lastday*), set *workweek* as 1. This means that *todayday* is within the WW1 of the next year.
    b. If *todaymonth* is 12 and *todayday* is less than (31 – *lastday*), get *workweek* using Python code. If *thisnewyearww* is 0 (due to first week not having a Monday), add *workweek* by 1.
    c. If *todaymonth* is not 12 and *todayday* is less than (31 – *lastday*),), get *workweek* using Python code. If *thisnewyearww* is 0 (due to first week not having a Monday), add *workweek* by 1.

Using the above algorithm, the script adds the work week for the DateTime produced and appends it to the DataFrame. The DataFrame is also appended with the employee ID of the reporter, as well as the fixture ID.

The file organisation is done by having a folder to contain all the file outputs for the Probe Replacement Tracking Tool. This folder, named 'ProbeReplacement', will contain the master CSV file, fixture ID folder, and the 'plots' folder. The master CSV file contains all the entries for probe replacement done on all fixtures in the factory. This CSV contains the full DataFrame, including the appended fixture ID. The fixture ID folder contains the CSV for the fixture that had probes replaced. This CSV only contains data from probe replacements done on that fixture ID. Since the fixture IDs are already highlighted by the folder and CSV file name, the fixture ID column appended earlier is dropped before outputting to the CSV. The 'plots' folder contains the Bar Chart of the number of each Probe ID replaced and the number of Failure Modes for each replacement. The 'plot' folder has folders named by fixture ID (organizing the plots into respective fixture IDs), with the plot PDFs named with the date of the file generation.

After the CSV files have been generated into the designated directory locations, the main script then calls the graph.py script to plot and generate the Bar Chart from the data.

Plotting Pareto Chart (graph.py)

The script takes in the CSV file of the fixture ID (not the master CSV file) and the fixture ID as the input arguments. After the file organisation has been set for the plots (mentioned earlier for the 'plots' folder and its files), the script converts the CSV file data (taking only Probe ID and Failure Mode) to lists. The lists are counted for the quantity (or frequency), and a bar chart is generated for each of the list. As a result, a plot (with 2 subplots, one for each bar chart) is produced with bar charts of the number of each Probe ID replaced and the number of Failure Modes for each replacement. The plot window is produced using *matplotlib* Python library.

Aside from displaying the plot window at the end of the tool, the plot is also saved and exported to a PDF, which is stored into the file organisation for 'plots' folder, mentioned above.

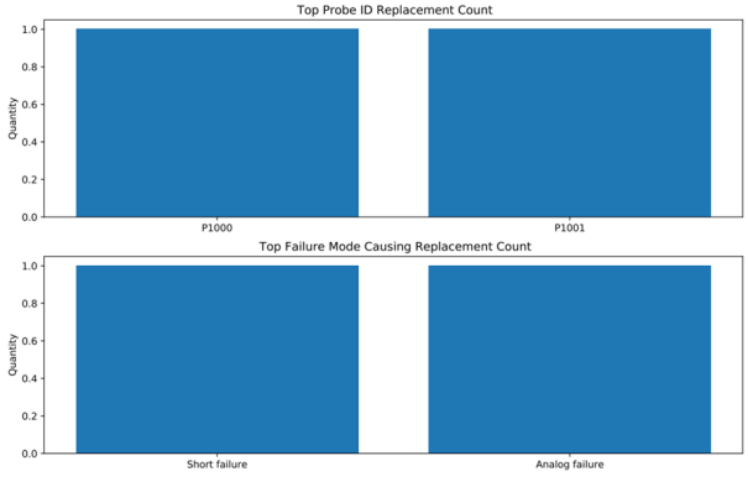


Figure 27: Bar Chart plot of Top Probe ID and Top Failure Mode

5.1.2.4 Contact Resistance (CRES) and Spring Rate Measurement Device

During the time of writing, we are able to make use of the Probe Cycle Count (hardware and software), probe ID and its specifications (probe size, probe location, node name), as well as failure modes related to the probe replacements to identify the health of an ICT fixture probe. However, as mentioned before, the probe cycle count is good as an estimate on the health of the probe. As such, the viable quantities of an ICT fixture probe that needs to be measured are the contact resistance (CRES) and the spring rate.

The measurement of these quantities is quite a challenging task, as the quantities come in very small values. The measurement of the quantities should also have a range that is greater than the range of measurement, as is the best setting for all measurement devices. The measurement of CRES and spring rate requires a specialised device to be built.

First Designs and Exploration

From the first research and designs, it is identified that the parameters that need be measured are the Spring Force against the Spring Displacement (this is Spring Rate), as well as the CRES of the probe. The necessary measurement devices of circuit configurations that enable these measurements are:

1. Electronic Load Cell – Strain Gauge (force sensor) with Wheatstone Bridge Configuration ***(measures Spring Force)***
2. Linear Variable Differential Transformer (LVDT) ***(measures Spring Travel or displacement)***
3. 4-wire Kelvin Measurement ***(measures Contact Resistance)***

For the first designs, the mechanisms for the measurements are proposed. The spring rate measurement is designed to have a load pushing down onto the probe vertically (force applied to the probe, and thus the spring). This load is equipped with the strain gauge, which senses the reactive force that the spring exerts in response to the force applied. The strain gauge is then wired to a Wheatstone Bridge Configuration, which makes up the Electronic Load Cell.

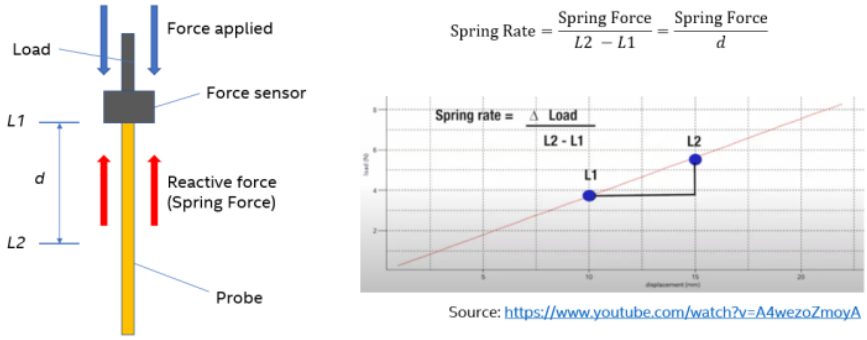


Figure 28: First designs of Spring Rate Measurement Mechanism

While measuring the spring force, the device also measures the spring travel as the force is applied. This is accomplished by the LVDT. Once the device has measured the spring travel, along with the spring force obtained from the Electronic Load Cell, the spring rate is then calculated using the equation:

$$\text{Spring Rate} = \frac{\text{Spring Force}}{L2 - L1} = \frac{\text{Spring Force}}{d}$$

The measurement of CRES uses the 4-wire Kelvin Measurement method. This is applied due to the very sensitive and high-resolution nature of the measurement. By using the 4-wire Kelvin Measurement, the effects of electrical resistance of the measurement probes (used to measure the CRES of ICT fixture probe), can be reduced. This will improve the accuracy of the CRES measurement.

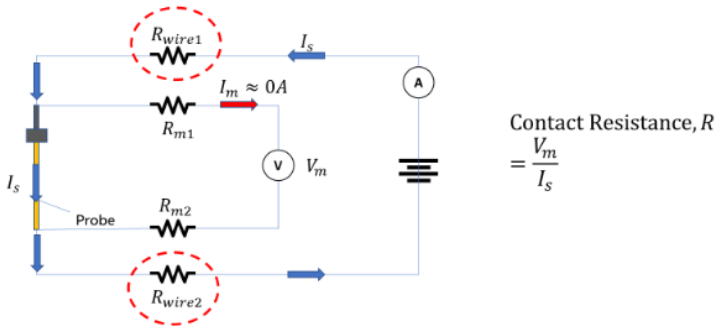


Figure 29: First Designs of CRES Measurement Mechanism

Using the 4-wire Kelvin Measurement, a high current is pumped through the ICT fixture probe using a pair of probes. A separate pair of probes is then used to measure the voltage across the ICT fixture probe using a voltmeter. The CRES is then obtained from the calculation of Ohm's Law.

$$\text{Contact Resistance, } R = \frac{V_m}{I_s}$$

Vendor Support and Designs

Once the first designs have been confirmed, it is possible to know the direction of how the measurement device should be built and what parameters that need to be measured.

At the factory, the specifications of all probes used for all fixtures can be found in probe lists. The probe lists, along with specification documents obtained from the fixture suppliers were studied to obtain the required range of measurement for the device. After studying the documents, the following measurement ranges have been obtained:

- Spring Force (in oz):
    - Required Range: 0.00 ~ 50.00
- Spring Travel (in inches):

- Required Range: 0.00 ~ 0.75

- Contact Resistance, CRES:
  - Required Range: 1mΩ - 10,000mΩ

ICT fixture probes have different pin head types, selected depending on different test point types. While there is a variety of pin head types used in the factory, the full pin list is compiled and provided to the vendor.

The design proposal provided from the vendor resembles the first designs provided. The vendor's design differs in the introduction of motor drivers and linear stage to allow the precise movement of ICT fixture probe during the measurement. Furthermore, the measurements for CRES employs a Source Measure Unit (SMU) to enable high resolution measurement. There is also an included software that helps in the measurement automation and data collection for the measurements made. This helps for engineering study of the probe health in a centralised database and dashboard.

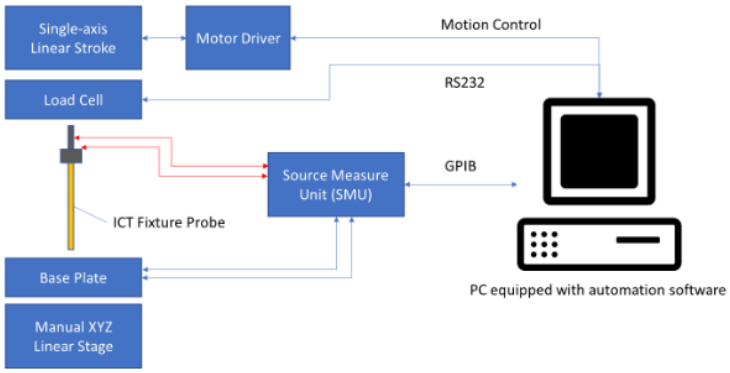


Figure 30: Proposed Designs (Functional Block Diagram) from Vendor

Test Setup

The test setup comprises of the functional block for housing the fixture probes to be measured, the connection of the test probes to the fixture probe, as well as the linear stages and motor drivers that enable the proper contact during measurement.

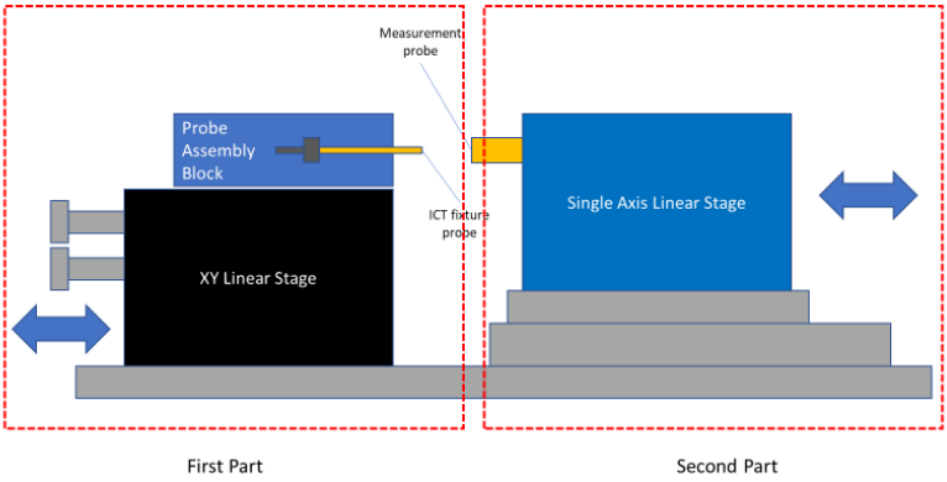


Figure 31: Test Setup Diagram

The first part of the test setup contains the fixture probe assembly block and the XY linear stage. The fixture probe assembly block houses the fixture probes that are to be measured for CRES and spring rate. The blocks are custom designed either to be based on different pin head types (in this case, there would be many assembly blocks, one for each pin head type), or to be based on the probe sizes and lengths. The assembly block design according to pin head types has base plates in the shape that allows best contact with the specific pin head type, whereas the block design according to probe sizes and lengths implements a generic base plate to contact the ICT fixture probes.

With the XY linear stage, it is possible for the user to move and adjust the probe position on the test setup before measurement is conducted. The design for the stage can be motorised, to allow automation and pre-setting for best ICT fixture probe contact with the measurement device. Alternatively, the stage can be designed without motors, whereby the user would need to manually tweak and adjust the ICT fixture probe to come in contact with the measurement device.

When adjusting the position of the fixture probe, it is important to have the fixture probe come into contact with the measurement device test probe, without compressing the fixture

probe spring or applying pressure to either the fixture probe or device test probe. Any mechanical changes that exist within the fixture probe or test probe, as well as pressure and differences in contact orientation may introduce deviations and errors in measurement of CRES and spring rate. As highlighted, the measurement of these parameters is extremely delicate, as the ranges and values are in high resolutions.

The second part of the test setup has a single axis linear stage and the test probe. This part of the test setup places the device test probe to come in contact with the ICT fixture probe, which is held by the first part of the test setup. The test probe is standardised, regardless of the pin head types, probe size or probe length. The single axis linear stage is equipped with a motor driver, which allows the device test probe to apply force to the ICT fixture probe, for the purpose of measuring spring force. When it comes to force applied (and displacement applied) to the ICT fixture probe, the measurement device should not apply excessive force to the fixture probe to cause damage to the spring or exceed the spring travel limit of the ICT fixture probe. The limits of the spring force and spring travel limit is highlighted in the specifications of each fixture probe used at the factory and is taken into consideration during the design of the measurement device.

Source Measure Unit (SMU)

To accomplish the measurement of the electrical quantities needed for CRES and spring rate measurement, a source measure unit (SMU) is used to measure the values. The SMU is wired to the test setup, whereby one connection is to the base plate of the ICT fixture probe assembly block (first part of test setup), while the other connection is to the test probe (second part of test setup).

Aside from connecting to the test setup, the measurement device is to be connected a computer, which contains the automation software. The connection made is via General Purpose Interface Bus (GPIB) to the computer. Due to the need for GPIB connection, the SMU selected for the measurement device should have the connection enabled (the presence of a GPIB port). When being wired to the test setup, the SMU needs to use the 4-wire Kelvin Measurement, as mentioned in the first designs of the device. This allows for a more accurate measurement of CRES and spring rate. As for the resolution of the SMU, the SMU should be

able to measure at values with 6 decimal places and above, to allow measurement of high resolutions.

Automation Software

The automation software is used to control the measurement device (namely the motor drivers), while enabling pre-setting and easier data collection. Some of the features that the automation software should possess include:

- Different level of user login
- Allow pre-condition setting for each ICT fixture probe type
- Able to save/recall different ICT fixture probe profile and setting
- Software control for displacement of ICT fixture probe spring travel
- Able to perform spring or contact force measurement in manual or auto mode
- Able to perform CRES measurement in manual or auto mode
- Real-time datalogging in CSV format

5.2 Project Activities

Development of Problem Statement

During the start of Student Industrial Training (SIT) semester, the problem statement and background was described by the host company supervisor (HCSV) to both the intern and the work buddy. During this phase, the expected project title was also provided, whereby a tool is needed to help in generating ICT fixture PM reports, while helping in the data acquisition related to the probe cycle count of the fixtures in the factory. At the same time, further exploration needs to be done to enable a more direct measurement of ICT fixture probe health, perhaps through measurement of CRES and spring rate of ICT fixture probes.

While the problem statement and background has been described, there is still some research and studying needed to understand the current conditions for the writing of ICT fixture PM reports, identifying the probe cycle counts from the tester PC, and handling replacements of ICT fixture probes. This was frequently communicated and informed by the work buddy, as the work buddy is the module engineer for ICT. From the discussions, it has been found that most of the reports are not stored in softcopy formats, while not having a centralised storage or database to keep the ICT fixture PM reports. As a result, it introduces difficulty in retrieving historical reports for reference and engineering study (probe cycle count and probe health studying). It is also found that the probe cycle counts are stored in individual folders of each product, which does not provide a centralised dashboard to analyse the cycle counts. As for the probe replacements, the replacements are not tracked in a centralised database or dashboard, which means that there is a lack of data collection on the failure modes for probes replaced.

As for the CRES and spring rate measurement, as this is a new measurement capability that is trying to be enabled, most of the information gathering is done through research and exploration of solutions by other companies or institutions. Much communication with team members is also done to understand the possible impacts of CRES and spring rate measurements, as well as the possible relationship between CRES and spring rate with the ICT fixture probe NDF cases.

Scope of Work and Tool Selection

After understanding the problem statement, background, and current conditions, the scope of work for the project is developed. There are two major options to explore with regards to developing a solution to building the data collection tools. One of the options is to develop or implement a web-based tool, while the other option is to develop a macro-based (or scripting) tool.

If the web-based system is selected, the system will allow users to log onto a web domain that runs the system. This domain (or website link) will allow users to either input new data (fixture PM report or probe replacement), or to search for historical data entries. When considering using a web-based system, it is necessary to think of the required tools and equipment needed to execute a website. These include a web domain registered under the Intel domain, user privilege and access levels (according to Intel's employee grouping in Office365), as well as IT development support (if not deciding to self-develop the full system). When it comes to developing websites, the full functions of the website need to be described in terms of its front-end, back-end and the database that handles the requests. When discussing front-end, there are a number of possible programming languages and tools to be used, such as the standard HTML/CSS/JavaScript languages. When it comes to back-end, there are also a few frameworks and tools at our disposal, such as PHP programming language, or frameworks such as Ruby On Rails and Laravel. As for databases, while developing on the developer's computer (localhost), would opt for MySQL, the database format used at the factory implements many different types, such as Microsoft SQL Server, MySQL, Oracle, and so on. It is necessary to understand what service and business unit that the project operates within to understand which servers and databases are available for the project.

If adopting a macro-based system is selected, it is necessary to decide which approach is to be adopted for scripting. In this context, it should be decided which programming language, frameworks or tools should be adopted to best execute the task. There are a number of approaches when it comes to macros and scripting. One of the more common languages used for scripting is Python. Python offers a simple syntax during programming, while providing an extremely large array of libraries to support any scripting or programming needs that the developer requires. Another popular scripting language used is Perl, which is quite commonly used in most organisations for developing scripts, albeit it being an older programming

language. When working on macros, especially when writing reports, Microsoft Office software (Word, Excel, Powerpoint, and Outlook) comes to mind. As such, there is also a viability to select programming tools related to the Microsoft Office tools. This would be accomplished by Visual Basic for Applications (VBA). VBA allows programming and macro development within Office tools, such as Excel shortcuts, custom toolbars, and simple automation. There is a challenge when choosing to use macro-based system, whereby the scripting should be able to scale or synchronise between all Intel employees associated to the tool. In other words, there needs to be a method to allow the scripting to not only work within localhost, but also to have the tool distributed to other employees, while allowing the users to search for reports completed by other users.

After weighing in the pros and cons of both the approaches (web-based system vs macro-based system), as well as considering the timeframe and capabilities of the developer, the macro-based system has been selected.

One of the reasons for selecting a macro-based system to execute the project is the shorter development time for the tool. A macro-based system would take a much shorter time to develop compared to a web-based system. Website development, as discussed earlier, is divided into front-end, back-end, and the database. Each of the sections requires different tools and frameworks, as well as a tedious setup and configuration stage before development. Opting for a macro-based system would cut down the languages and frameworks to a maximum of two, if not one. Since web-based systems work behind the Intel network and user privilege levels, there is a lot of communication, requests and approvals needed to have the web domain running for Intel employees. To have the project developed, trained, and tested before releasing to production requires time-savings at every possible part of the project lifecycle.

Another reason for adopting macro-based system is the ease of development. This is affected by both the necessary programming languages and frameworks to execute the project, as well as the capability of the developer. As mentioned before, website development requires many different languages and frameworks. While it is found that internal websites used for non-critical tasks are developed using Django framework, which is a web development framework using Python, there is still a higher difficulty in developing the web system, instead of adopting for a macro-based system. While Django greatly reduces the necessary

programming languages and frameworks needed, there is still the database handling, which introduces difficulty and a learning curve, within the project timeline.

While using the macro-based system has the challenge of the lack of a database, it is possible to implement a ShareDrive or Network Drive to store the tool. By giving access to the Network Drive to all associated Intel employees, it is possible to have all related personnel to product ICT fixture PM reports within a centralised drive location. As such, the produced files and reports that are generated by one user is easily visible and retrievable by another user, on the same drive location.

The analysis of pros and cons of implementing web-based system against macro-based system is summarised in the table below.

SOLUTION PROPOSALS	PROS	CONS
Utilise Intel IT Developers	• High quality system (equivalent to Circuit)	• Project may start developing next year (or not at all, due to low ROI)
Self-build (Web-based)	• Opportunity to learn more • Future use for ICT systems (sub-domains for other functions)	• Requires to-and-from communication with Intel IT on server usage (Microsoft SQL Server) • Requires hosting new web domain • Development will miss deadline
Utilise ShareDrive and Scripting	• Development can meet deadline • Easily maintained and controlled	• Less intuitive UI/UX (requires detailed instruction and documentation) • Elaborate scripting required to meet objectives

Table 1: Pros and Cons analysis between possible solutions to developing software project

With the tools and requirements confirmed (solidifying the scope of work), it is possible to start designing the tool, in terms of the user interface (UI) and user experience (UX) aspects.

UI/UX Designing and Database Designing

When designing the tool, it is necessary to visualise how the tool will appear to the user. Aspects of how beautiful or aesthetic the tool is, while having the associated menus, buttons and inputs to be clearly visible and appealing are all aspects of UI design. While using the macro-based system as opposed to web-based system does limit the UI capabilities of the tool, much of the UI efforts can be compensated using some Python libraries (*art* and *PySimpleGUI* are some examples). The most important aspect of UI design for the report generation tool is to ensure the users are able to clearly see the steps and questions asked by the tool when it comes to the procedures in ICT fixture preventive maintenance and probe replacement. This is to prevent the user from missing out on any steps during input, and thus generating a report with errors.

As for UX design, the tool needs to be carefully designed to allow ease and comfort in using the tool. Some of the aspects of giving ease and comfort to the user is by reducing unnecessary actions from the user, such as unnecessary text inputs, key presses, mouse clicks or navigations. The tool should also be easy to understand for the user, in the case of the next steps and how to use the tool. If the UX design of the tool is done correctly, there is minimal instructions, training, and documentation needed by the user before running the tool. UX design is especially important for this project, as the developer will not be at the factory or company for a long time, as well as the company may have a shift of team members under ICT who will be using the tool.

The UI/UX design is created and presented to the work buddy and HCSV before development of the tool begins. This is to present on how the tool will look like to users, as well as understanding what inputs are needed and gathered to produce the end result (the PM report). When displaying the UI/UX design, a useful tool is used, known as the wireframe.

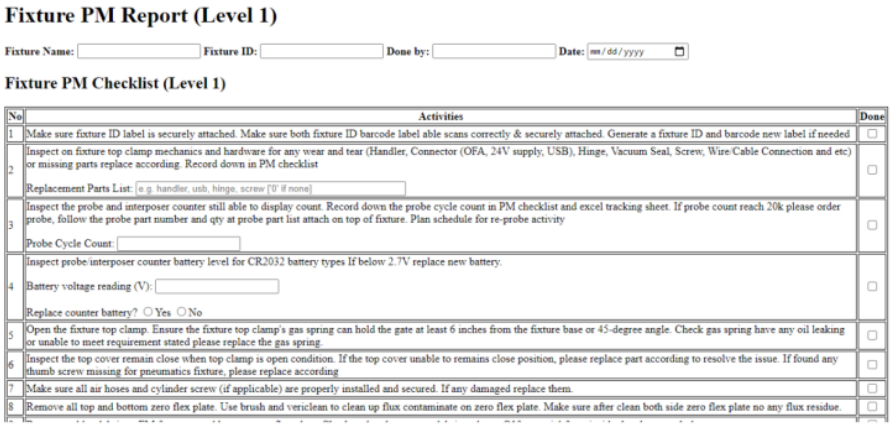


Figure 32: ICT Fixture Level 1 PM Report Generation Tool Wireframe (Menu)

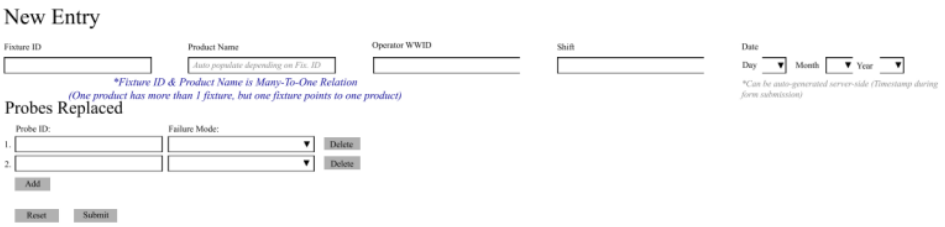


Figure 33: Probe Replacement Tracking Tool Wireframe (GUI window design)

Aside from the front-end design of the tool (UI/UX), there is a lot of data that is being handled and stored by the tool. Part of designing databases, especially designing the different data that is handled by the tool, involves knowing what the necessary features or columns of data is to allow the most efficient execution of the tool. This means that only necessary data is requested from the user to be input, while having secondary data (or less important data) being easily retrievable from another database or entity. The illustration of how the data is handled by different entities (or tables in databases), along with the relationship between these entities is shown by an entity relationship (ER) diagram. The ER diagram below describes the database design for the project.

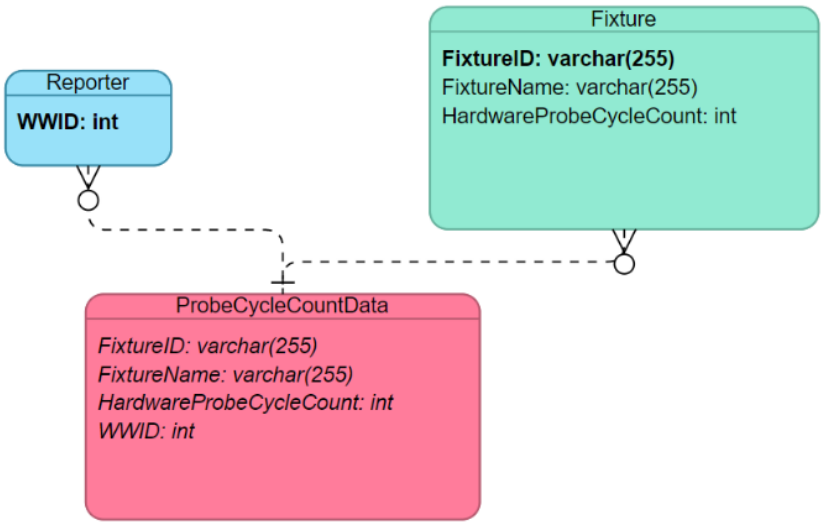


Figure 34: ER Diagram for ICT Fixture PM Report Generation Tool

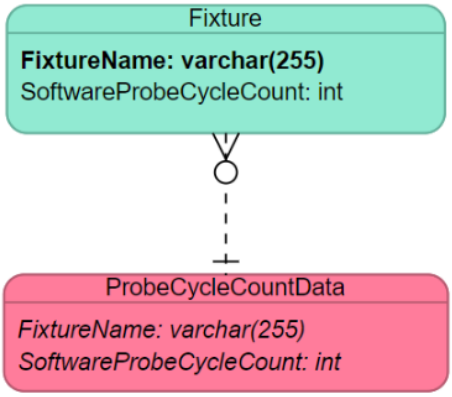


Figure 35: ER Diagram of Probe Cycle Count Harvesting Script

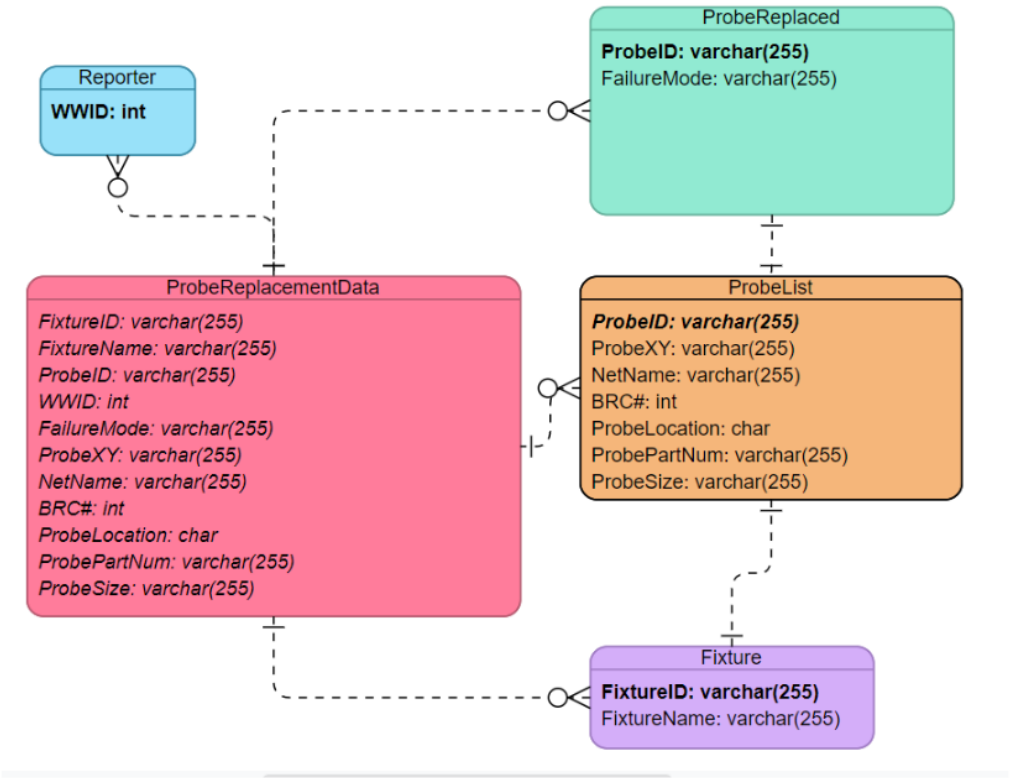


Figure 36: ER Diagram for Probe Replacement Tracking Tool

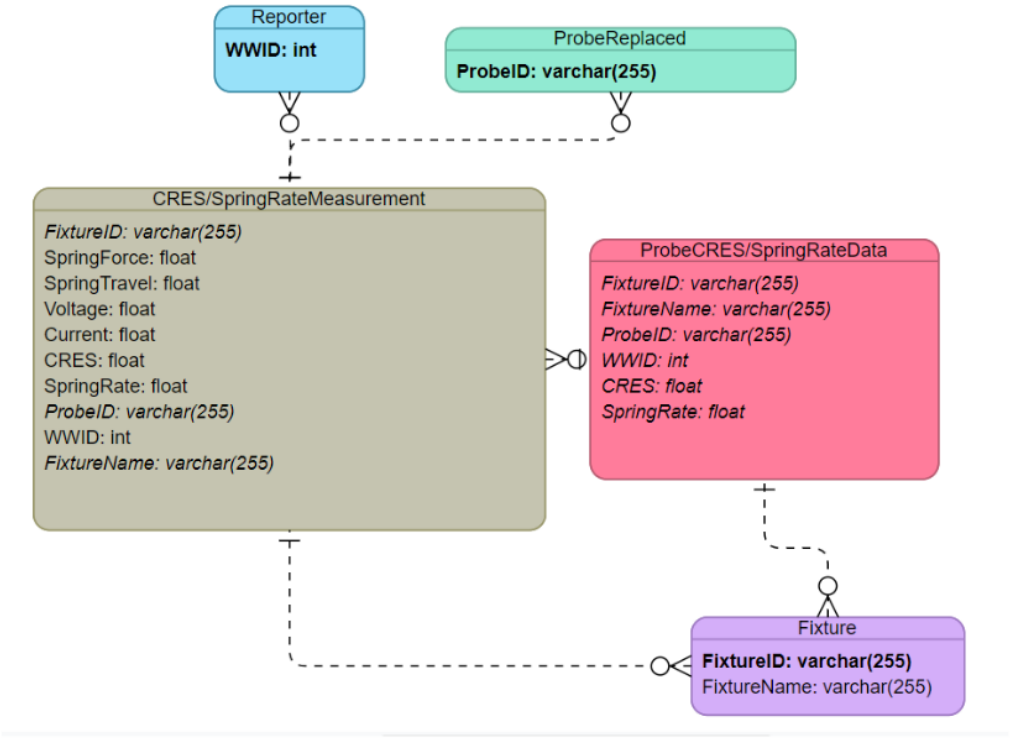


Figure 37: ER Diagram for CRES and Spring Rate Measurement Device

Alpha and Beta Version Development

Once the direction and designs of the project has been confirmed, it is easier to develop the tools according to the guidelines highlighted by the wireframe and ER diagrams.

Concepts of Developing Pre-release Versions

When developing an alpha and beta version, it is important to hold onto the rule of 'build cheap, build fast, test, reiterate'. What this means is that the pre-release versions should be something that uses the least resources (both in monetary, manpower, and time), while being easily deployed for testing. After testing, the results should help in the reiteration and improvement from the current release. This will build towards a more robust tool as the development progresses through the project timeline.

Another important aspect of building a good alpha and beta version of a tool is to allow for maximum data acquisition from the tool. In other words, the alpha and beta version should be able to yield results on how well the tool is performing. This can come in the form of feedback tools, built-in error logging capabilities, runtime and performance calculations, and so on. When developing the projects, this important aspect is implemented in the tools, such as error logging capabilities and runtime/performance tracking. Furthermore, while the tools were not built with inherit feedback tools, the testing and deployment were done in structured control environments (starting from localhost deployment and testing, to testing within module team, to testing with a targeted segment from the intended user group of the tool). This has allowed regular and frequent feedback to be given to the developer to allow for quick troubleshooting and improvement to be executed onto the project.

Version Control and Documentation

When developing the project, especially for the software, it is important to keep track of the releases of the different versions of the project. As such, most of the project development involved version controlling and documentation of the releases using version control software. The most frequently used version control software in the industry, as well as the software used by for this project is Git and GitHub.

Individual repositories were created using GitHub for each of the data collection tools. Within the repositories, the tools (in the form of scripts and generated executable files) were stored in the main branch. Whenever a new version release is developed, the changes are pushed to the repository, with the appropriate push message highlighting the changes made to the code. The inclusion of the messages helps in tracing the new features or bug fixes made to the software. While it is common practice in the industry to develop on a new branch and do a pull request to merge with the main branch, since the project is smaller in scale and there is only one developer for the tools, the software is pushed directly to the main branch.

As for the documentation, the project is described in detailed within the documentation of each repository. This is accomplished by writing in the README.md file for each repository. The format of documentation follows the following:

- Introduction

- Installation (for both pure usage and development purposes)
- Usage
- Contributing and Bug Reporting (email address of the developer)

Installation

For usage only

The repository contains the .exe generated from the Python scripts, produced using Pyinstaller. If you wish to run and use the tool, please download the following file and folder:

- menu.exe
- wkhtmltopdf

menu.exe is the .exe file from the Python scripts. ***wkhtmltopdf*** folder contains the files necessary to run the HTML-to-PDF conversion for the PM reports. Without the ***wkhtmltopdf*** folder, the script will produce an error and will not generate the PDF file, thus stopping the program at the HTML generation (Probe Cycle Count and Pareto Chart generation will also not run).

Ensure that **both the file and folder are in the same directory**. Placing any of the files either in a separate folder or in a parent/subdirectory will produce an error as above.

For geeks who want to play with the code

You would need to clone the following files:

- images.py

Figure 38: Documentation of ICT Fixture PM Report Generation Tool in GitHub

Communication within the project team

Throughout the development cycle, there is frequent communication and presentation done with the work buddy (also as the module engineer for ICT), as well as with the HCSV (as the 'Decider' role in the project). The communication is done at regular intervals, depending on the difficulty and size of the project. For the data collection tools, most of the communication and presentation is done fortnightly, with more critical issues, requests and enquiries being communicated immediately or at the nearest convenience. As for the ICT Fixture Probe CRES and Spring Rate Measurement Device, communication and presentations are done almost every week, with certain instances of the communication done more than once a week. As is shown, the difficulty, size, and novelty of a project requires different amounts of communication and presentation time for the project.

Communication with teams outside the project team

For certain areas of the project, there is a need to communicate with individuals and teams outside the Parametric Test team. Some communication is also done with individuals outside the organisation, such as vendors and suppliers.

During the start of the ICT Fixture PM Report Generation Tool and Probe Replacement Tracking Tool development, there has been communication with managers from Functional Test to work on a collaborative solution. The purpose of the communication has been mostly centred on seeking assistance in development of the tools, combining the proposed tools with the Functional Test team's available tools, as well as guidance and associated communication with the IT development team working with Functional Test.

Communication with the IT team has been critical to the development of the project. While the IT team was not enlisted to develop the project, a lot of guidance and suggestions were provided by the team in helping to decide the scope of work of the project (selection of tools and approach). Aside from that, the IT team is the main focal person when it comes to setting up the database or network drive which houses the data collection tools. As such, much of communication, in terms of the creation of the network drive, size of the network drive, and user privilege levels granted, were done with the IT team.

During the development of the ICT Fixture Probe CRES and Spring Rate Measurement Device, there has been communication with teams in Functional Test and Debug. Communication has been frequently done with

Communication with personnel outside the organisation

For the ICT Fixture Probe CRES and Spring Rate Measurement Device, as a lot of the project execution relied on a vendor, frequent communication was done with this party. The communication centred around presentation of first designs, proposal and realignment of scope of work (device requirements, measurement ranges, and probe specifications), as well as device design proposal and quotation from vendor.

There were occasional meetings held at the factory between the project team members and representatives from the vendor team. While these meetings were held for bigger occasions and discussing more critical agendas, the large majority of communication is done through email. At the same time, the limitations of travelling due to the pandemic has caused most communication to be done virtually.

The development of the CRES and Spring Rate Measurement Device is currently at the quotation phase. While the contributions during the internship period has allowed Parametric Test team to explore the possibilities of measuring ICT fixture probe CRES and Spring Rate, while obtaining first designs and vendor designs, as well as quotation from the vendor for the device, the project is to be continued for execution (device building and deployment) after the internship period.

Validation and Deployment for Production

Once a stable release of the tools has been developed, with the necessary requirements and approval from the work buddy and HCSV, the tools are ready to be released for validation. Validation involves releasing the tools to the targeted user group in a controlled and structured process. Validation aims to prove the stability and proper functioning of the tool within a real use case and production environment. Once the tools have been deemed to have passed validation, it is safe to have the tools to be released for production use.

The data collection tools were developed to validation standards. They have been tested by the work buddy (module engineer) and its operation has been demonstrated to both the work buddy and the HCSV. As part of the validation phase, the tools are released to be tested by the technicians who will be the main user segment for the data collection tools. Any errors and issues that occur, as well as difficulties and feedback that arise from the tools, are shared with the module engineer and the developer (intern) to troubleshoot and modify. While there could possibly be minor bugs and exceptions that occur during the validation runs, it should be at the minimum. The developer is also tasked to monitor and conduct the necessary bug fixes immediately when there are reports of issues. Since a tool operates best when thorough testing has been conducted, it is best for validation to begin as soon as possible (at least 3 weeks before releasing to production), to allow for best results from the data collection tools. These principles have been implemented for the project.

5.3 Gantt Chart

5.3.1 Data Collection Tools

TASK	MONTH/WEEK															
	SEP				OCT				NOV				DEC			
Development of SIP topic																
Study project requirements and background																
Develop problem statement & objectives																
Develop scope of work (SOW) and confirm tools and framework																
UI/UX Designing, Database Designing (wireframe & ER Diagram)																
Begin Development																
Request for network drive for storage of tool and generated outputs (Fixture PM Report Generation Tool)																
Complete Development of tools																
Testing & Admin Training (maintainers)																
Validation and deploy for production use																
SIP report writing																
SIP report submission																
SIP presentation																

5.3.2 CRES and Spring Rate Measurement Device

TASK	MONTH/WEEK															
	SEP				OCT				NOV				DEC			
Development of SIP topic																
Study project requirements and background																
Develop problem statement & objectives																
Explore & Research Parameters to be measured																
Develop First Designs																
First Approach and Presentation of First Designs with Vendor																
Confirm Scope of Work (SOW), Measurement Ranges and Device Specifications with Vendor																
Vendor proposes designs, quotation																
SIP report writing																
SIP report submission																
SIP presentation																

6.0 RESULTS & DISCUSSION

6.1 ICT Fixture Preventive Maintenance (PM) Report Generation Tool

From the development of the ICT Fixture Preventive Maintenance (PM) Report Generation Tool, both the Level 1 and Level 2 PM activities done on ICT fixtures are now able to be recorded in a centralised database, on a collaborative network drive.

ICT Fixture LEVEL 1 PM Report for Fixture_01 (FIX_01)

Reported by: 12345677
Fixture Name: Fixture_01
Fixture ID: FIX_01
Date Reported: 2020-12-06

Items	Contents
1. Make sure fixture ID label is securely attached. Make sure both fixture ID barcode label able scans correctly & securely attached. Generate a fixture ID and barcode new label if needed	Done
2. Inspect on fixture top clamp mechanics and hardware for any wear and tear (Handler, Connector (OFA, 24V supply, USB), Hinge, Vacuum Seal, Screw, Wire/Cable Connection and etc) or missing parts replace according. Record down in PM checklist.	Done
Replacement parts list:	None
3. Inspect the probe and interposer counter still able to display the count. Record down the probe cycle count in PM checklist and excel tracking sheet. If probe count reach 20k please order probe, follow the probe part number and qty at probe part list attach on top of fixture. Plan schedule for re-probe activity	Done
Probe Cycle Count:	20000
4. Inspect probe/interposer counter battery level for CR2032 battery types. If below 2.7V replace new battery.	Done
Battery voltage reading:	None
Replace counter battery YES / NO	None

Figure 39: Sample Output of Generated ICT Fixture Report

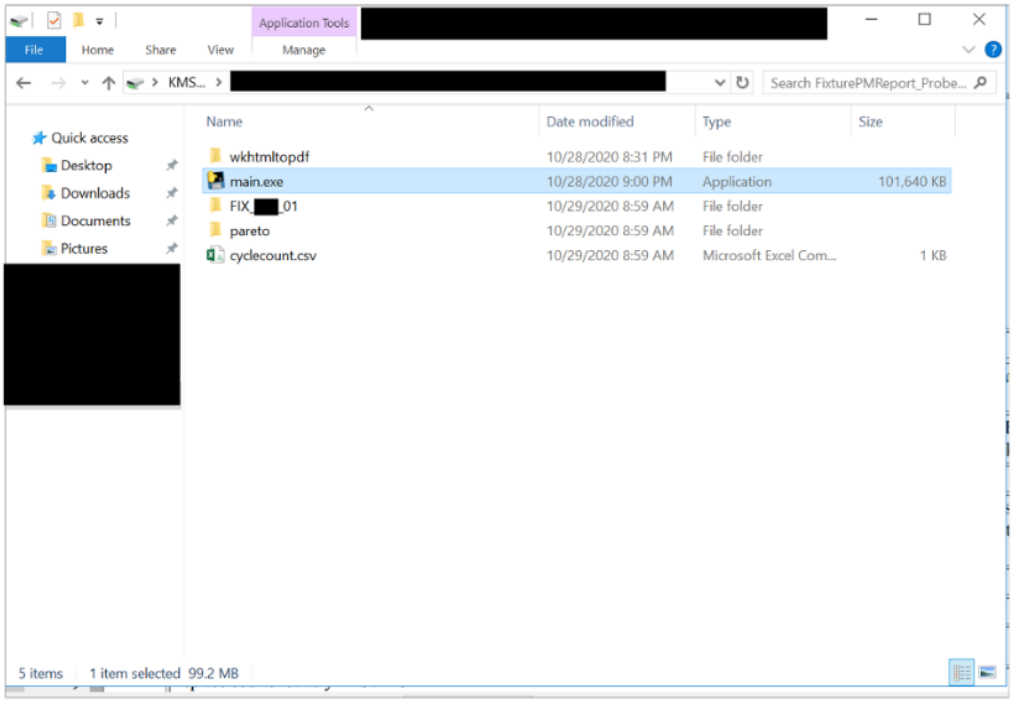


Figure 40: Report Data CSV file, HTML and PDF reports generated in Fixture ID folder

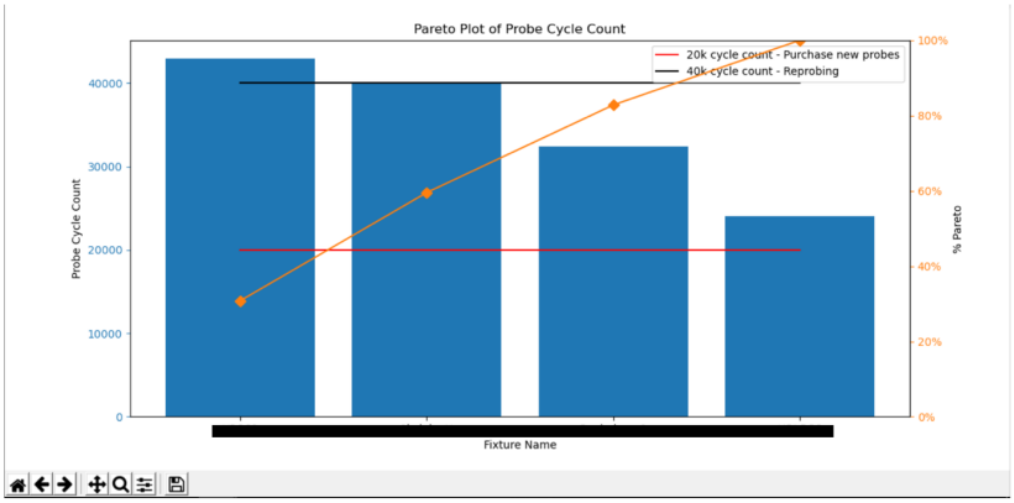


Figure 41: Pareto Plot of Top Hardware Cycle Counts retrieved from Level 1 PM reports

Time Savings for Generating, Storing and Retrieving Reports

Before the implementation of the project, the reports are written by editing an existing Word template. This has caused additional time that is needed to produce the report, as the details of the Word template would need to be manually edited for an individual PM report. Using the developed report generation tool, it allows the user (technicians conducting the PM activity) to quickly and easily generate reports.

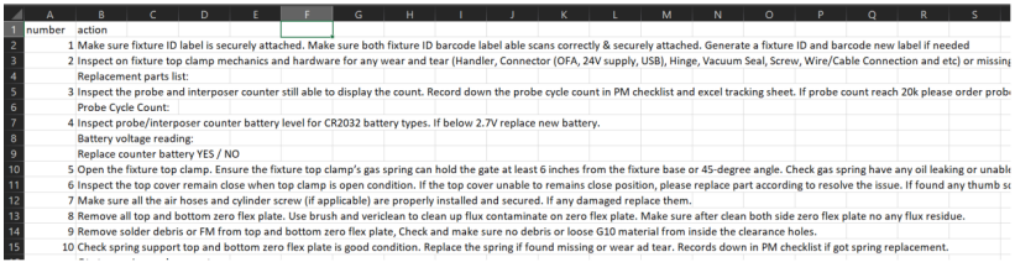
There is also less wasted time in needing to store and retrieve the written reports. On some occasions, engineering team would require checking or study on historical PM reports for reference on a fixture. Before the project implementation, the reports would either be printed and stored in cabinets, stored within the computer of the technician, or emailed to the module engineer on some occasions. With the introduction of not only a centralised database (network drive) to store all historical reports generated, but also an organised file/folder system according to fixture ID, it allows for quick retrieval of the reports whenever required for studying.

Organised File/Folder System

The tool allows users to generate the reports and store them in an organised system, according to the fixture ID. This makes the historical reports to be easily found and retrieved, as a user would need to search the reports by the folders named with the respective fixture ID. With the introduction of an organised file/folder system, it is also less likely for files and folders to be tampered with or go missing, as not only are the reports stored in a PDF format (which prevents editing) but are generated with timestamps and date as file names.

Dynamic Template and Adaptability to Specification Changes

For the purpose of adaptability to factory specification changes and future expansion, the tool has been developed to be dynamic. The report generation tool works by reading the steps and procedures from an Excel file. The Excel file can be easily edited to change the PM activity steps or add additional steps to the activity by the module engineer. There is no need for the engineer or a developer to edit the code each time there is changes made to the factory specifications for PM.



number	action
1	Make sure fixture ID label is securely attached. Make sure both fixture ID barcode label able scans correctly & securely attached. Generate a fixture ID and barcode new label if needed
2	Inspect on fixture top clamp mechanics and hardware for any wear and tear (Handler, Connector (OFA, 24V supply, USB), Hinge, Vacuum Seal, Screw, Wire/Cable Connection and etc) or missing
	Replacement parts list:
3	Inspect the probe and interposer counter still able to display the count. Record down the probe cycle count in PM checklist and excel tracking sheet. If probe count reach 20k please order prob
	Probe Cycle Count:
4	Inspect probe/interposer counter battery level for CR2032 battery types. If below 2.7V replace new battery.
	Battery voltage reading:
	Replace counter battery YES / NO
5	Open the fixture top clamp. Ensure the fixture top clamp's gas spring can hold the gate at least 6 inches from the fixture base or 45-degree angle. Check gas spring have any oil leaking or unabl
6	Inspect the top cover remain close when top clamp is open condition. If the top cover unable to remains close position, please replace part according to resolve the issue. If found any thumb s
7	Make sure all the air hoses and cylinder screw (if applicable) are properly installed and secured. If any damaged replace them.
8	Remove all top and bottom zero flex plate. Use brush and vericlean to clean up flux contaminate on zero flex plate. Make sure after clean both side zero flex plate no any flux residue.
9	Remove solder debris or FM from top and bottom zero flex plate, Check and make sure no debris or loose G10 material from inside the clearance holes.
10	Check spring support top and bottom zero flex plate is good condition. Replace the spring if found missing or wear ad tear. Records down in PM checklist if got spring replacement.

Figure 42: Excel File containing steps and questions for conducting PM activity

Collection of Hardware Cycle Count and Plotting of Graph

One of the most important improvements that the tool has provided is the collection of hardware probe cycle count from the fixtures that are reported in the tool. When the PM activity is conducted, the recorded probe cycle count is retrieved separately onto another CSV file. A Pareto Chart is also plotted to display the probe cycle counts of each fixture, with the cycle count limit lines drawn onto the chart. This popup of the chart each time the report generation tool is executed acts as a dashboard to remind and notify the user that Level 2 PM (or fixture reprobing) needs to be conducted, in order to reduce NDFs to occur due to damaged or malfunctioning probes. The plots are also saved as PDF and stored in a separate folder, whereby the module engineer can easily refer to for the latest plots to know the fixtures that are pending Level 2 PM. With the collection of hardware cycle count, this enables the studying of the ICT fixture probe health, which is the overall goal of the project.

6.2 Probe Cycle Count Harvesting Script

With the implementation of the Probe Cycle Count Harvesting Script, it is now possible to have a dashboard of all software probe cycle counts of the ICT fixtures running in the factory. The dashboard, displayed in the form of a table, can be run monthly to show the trend of probe cycle counts of the fixtures.

Software Cycle Count Report (2020-11-25)

BoardName	CycleCount_2020-11-09	CycleCount_2020-11-11	CycleCount_2020-11-18
[redacted]	7151	7696	9413
	8466	8765	9620
	2352	2457	2543
	2853	2874	2970
	1000	1000	1000
	51048	51465	52602
	432	432	470
	5082	5082	5082
	29246	29449	29928
	29411	29427	29432
	17358	17825	18284
	23641	23641	23641
	30050	30050	30050
	80498	80502	80513
	14135	14172	14238
	29755	29755	29755
	34583	34583	34583

Figure 43: Generated HTML file with Table of Software Probe Cycle Counts from running tool

Time Savings to Collect Probe Cycle Counts

Using the Probe Cycle Count Harvesting Script, it serves as a much quicker way to obtain all the probe cycle counts of fixtures being run at the factory. There is no longer the need to open the fixture counter files, which are either stored in a Counter folder or in respective product folders, in order to collect the probe cycle counts. There is plenty of time savings from running the tool as compared to locating and opening the individual fixture counter files.

Ability to Study Trend of Probe Usage

Since the tool can be run on any time interval required by the user, it is possible to run the tool in a specific interval that shows the trend of probe usage by all fixtures. By analysing the probe cycle counts for a fixture within a specified time interval (e.g., monthly basis), the tool provides the ability to make an estimate of how much the probes are used during the period, provided they have information regarding the volume of products that are being tested on the fixture during that period. As is the main goal of the project, this trend and the counting of the probe cycle allows an estimate of the health of the ICT fixture probe.

6.3 Probe Replacement Tracking Tool

Before the introduction of the project, ICT fixture probe replacements (done during production for the purpose of troubleshooting NDFs) are not formally conducted. While there are records made on certain occasions (perhaps for studying the performance of a problematic product), the lack of a formal replacement tracking has prevented efficient engineering study on the probe ID being replaced, probe physical specifications, and failure modes that cause the replacement. With the introduction of the tool, there is now the ability to track probe replacements, as well as gather the necessary engineering data that is needed to investigate probe-related test issues in the factory.

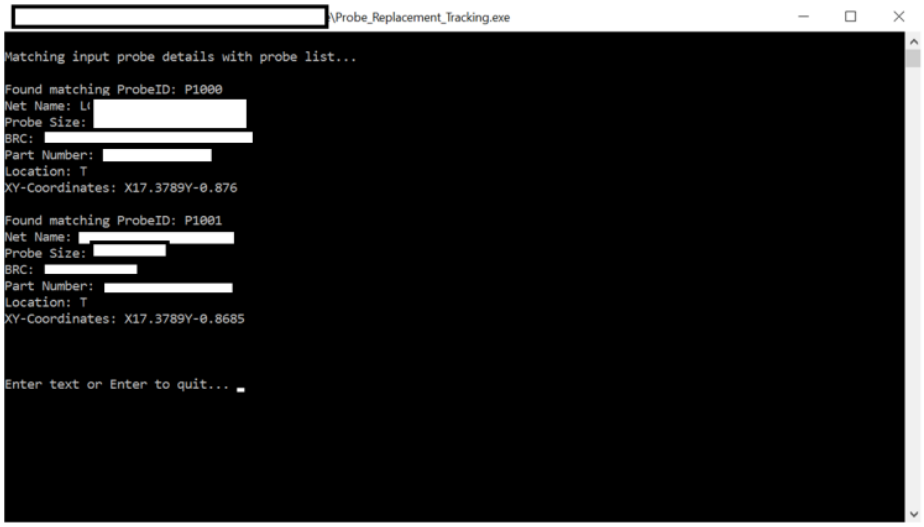


Figure 44: Terminal Display of matching probe ID with engineering data from Probe List

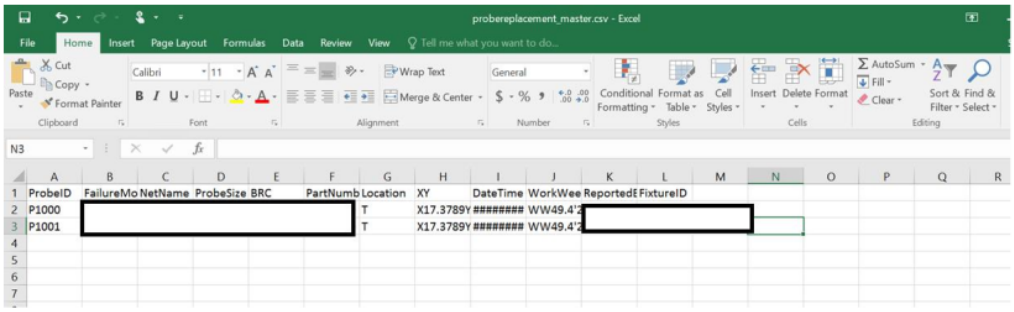


Figure 45: Master CSV file containing recorded Probe ID and respective engineering data

Database to Store Probe and Failure Mode Data

The Probe Replacement Tracking Tool stores the input from probe replacements using an organised file/folder system. This allows easy retrieval of the data, whether in the form of a master CSV file containing data from all fixtures or individual CSV files for each fixture in the factory. As opposed to the ICT Fixture PM Report Generation Tool, a separate network drive was not setup for this tool. Alternatively, the tool is kept in the tester PC drive, as the probe replacements are done directly on the fixtures during production, in the presence of the tester PC. The existence of a database to store this data is in itself a new introduction for the probe replacement activity.

Bar Chart Plots of Data

The tool also provides data visualisation related to the records provided during probe replacement. The bar charts produced analyse the data by the fixture ID. The charts show a summary of the top probe IDs that are being replaced, as well as the top failure modes that are causing probe replacements. This gives useful insights for engineering team to help in investigation of root causes for high NDFs or probe failures.

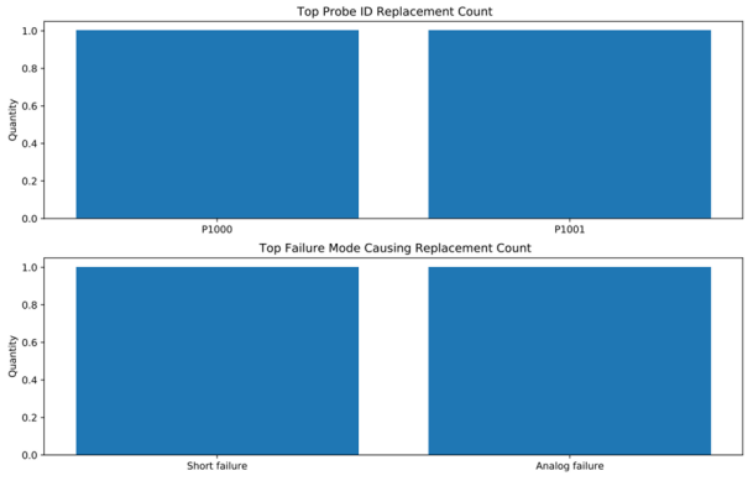


Figure 46: Bar Chart plot of Top Probe ID and Top Failure Mode

6.4 Contact Resistance (CRES) and Spring Rate Measurement Device

The CRES and Spring Rate Measurement Device provides an exploration on a direct approach to studying the health of ICT fixture probes. Different from probe cycle counts, the measurement of electrical quantities and mechanical properties of the fixture probe provides a more viable approach to understanding whether a probe is healthy enough to continue to be used for production.

Throughout the project timeline, the exploration has provided with not only understanding on the parameters that are being measured to understand ICT fixture probe health used in the industry by other organisations and institutions, but also resulted in designs and circuit configurations that will help enable the execution of this project.

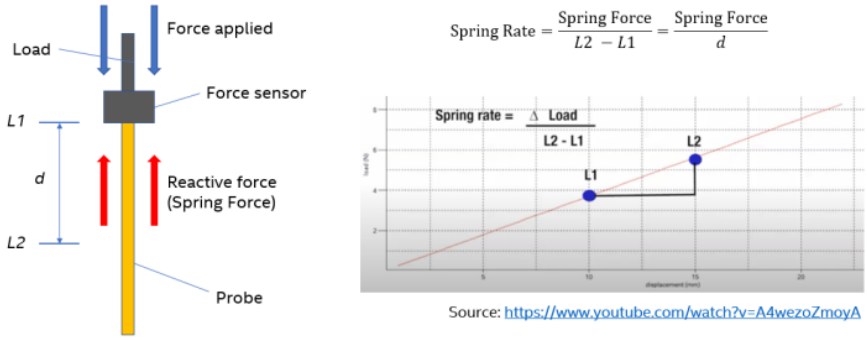


Figure 47: First Designs of CRES Measurement Mechanism

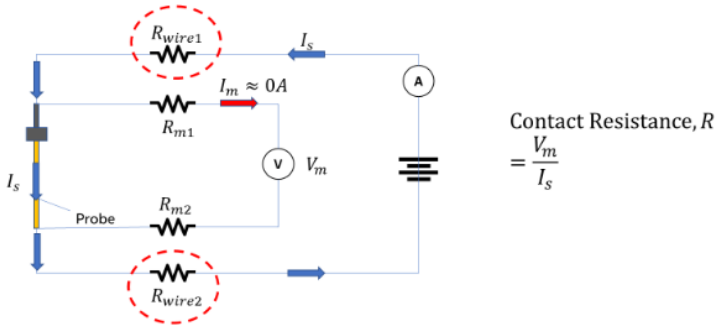


Figure 48: First Designs of CRES Measurement Mechanism

Furthermore, the communication, proposed designs and quotation provided by the vendor has provided the Parametric Test team with a solid direction on enabling CRES and Spring Rate measurement of ICT fixture probes. While the quotation has been provided by the vendor, the remaining parts of the project is set to be completed after the internship period. The areas that are to be worked on for the remaining of the project include the building of Return on Investment (ROI), provision and allocation of budget for the device development, as well as execution or measurement on probes using the device.

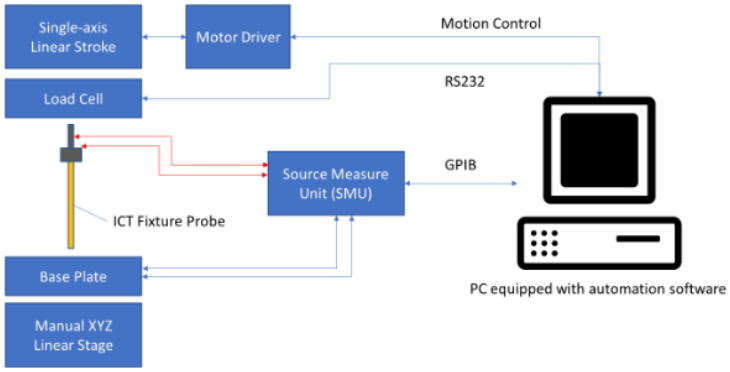


Figure 49: Proposed Designs (Functional Block Diagram) from Vendor

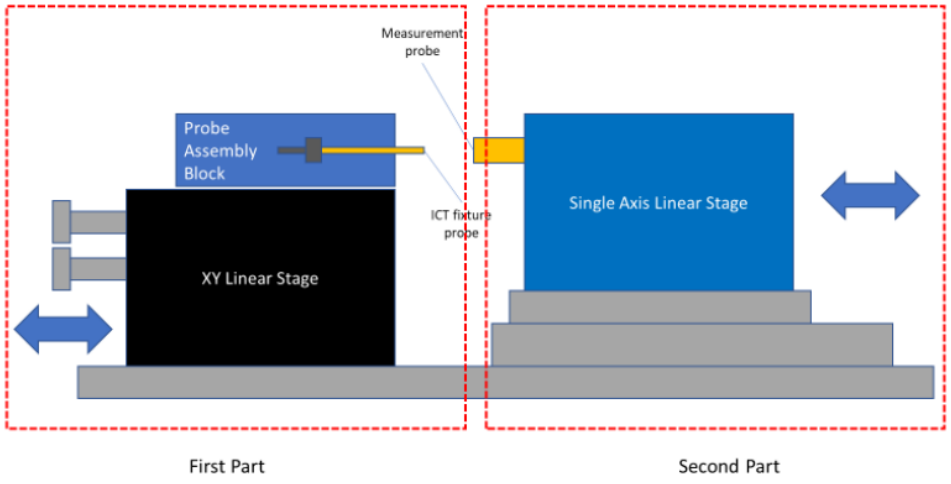


Figure 50: Test Setup Diagram

While there is a deep interest to provide measurement of CRES and Spring Rate of ICT fixture probes, the potential cost-savings for the device is still not at the optimal for the time being. This is because additional investigation and engineering study, along with validation with numerous products and product volume is needed in order to enable the extension of ICT fixture probe health for production purposes within the factory. Having said that, the

background study, designs, and quotation from the vendor has paved the way for enabling this capability within Parametric Test team, which will be further explored and executed after the internship period.

6.5 Overall Project Results

In total, the development and deployment of data collection tools (ICT Fixture PM Report Generation Tool, Probe Cycle Count Harvesting Script, and Probe Replacement Tracking Tool), along with the conceptualisation of the CRES and Spring Rate Measurement Device has allowed for available engineering data to be collected and studied within the factory.

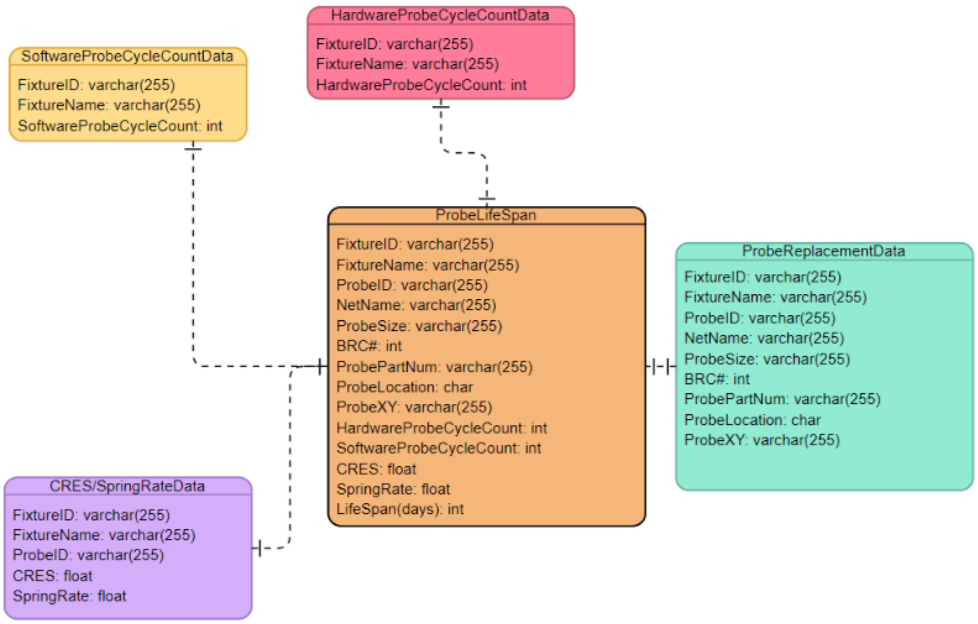


Figure 51: ER Diagram for Overall Project

Through the collection and studying of the data collected from running the tools, as well as measuring CRES and Spring Rate using the device after the internship period, it is possible to understand deeper on the possible causes of probe failures, as well as the impact of probe failures to the electrical and mechanical properties of the probe. This, in the big picture,

greatly helps the Parametric Team in investigating the root causes of NDFs caused by probe failures, as well as helps in identifying the action plan for reducing the NDF rate for ICT.

7.0 CONCLUSION & RECOMMENDATIONS

7.1 Conclusion

The test performance of In-Circuit Testers (ICTs) is an important aspect of ensuring manufacturing or process defects can be detected and repaired before releasing the product to the customers. The performance of ICT also ensures that there are no defects that escape to the next processes within the manufacturing line, potentially causing additional issues and defects to the product and machines handling the boards.

In order to measure the performance of ICT, the no-defect found (NDF) rate serves as a useful metric. It is the performance goal of ICT to reduce NDF rate below certain thresholds. While NDFs are caused by many factors, probe failures have been identified as a frequent contributor to high NDF rates. To allow further understanding and proper investigation of probe failures causing NDF, as well as identifying action plan to reduce NDF rate caused by probe failures, data collection tools and measurement devices are needed.

With the introduction of the ICT Fixture PM Report Generation Tool, not only will the hardware probe cycle count of all fixtures in the factory be easily extracted, viewed, and analysed by the engineering team, the tool also serves to ease the report writing and storage process for technicians and PM crew. After developing the Probe Cycle Count Harvesting Script, it is much easier to gather and display the software probe cycle count of all fixtures in the factory. This data also helps engineering team to estimate the probe usage within a defined time interval, thus helping to understand the lifetime of probes in a fixture. Through the Probe Replacement Tracking Tool, the probe replacements done on fixtures during production, as a troubleshooting approach, can be properly recorded, in terms of the probe IDs being replaced, failure mode, as well as the associated probe specification data. As for the CRES and Spring Rate Measurement Device, while it is not within the timeframe of the internship period to develop and use the device to measure the electrical and mechanical properties of the probes, the exploration, designs and proposal, as well as quotations from the vendor has paved the way for enabling CRES and Spring Rate, and thus the potential extension of lifespan of fixture probes within ICT.

The capability of gathering and studying all these data collectively gives better understanding to the engineering team on how different factors affect the ICT fixture probe health, enabling estimation and measurement of probe health, as well as exploring future capabilities and innovation to measure and extend the lifespan of ICT fixture probes.

7.2 Recommendations

Merging of all Data Collection Tools into A Single Software

During the development of the project, each tool is developed for a specific area of ICT (e.g., fixture PM and probe replacement). While the tools are serving different purposes across ICT, the contributions that are provided by the project helps the overall goal of ICT. Since the project is a cumulation of multiple data collection tools and measurement device, it is therefore a good approach to combine all the data collection tools into a single software. Not only would this help in centralising the tools, thus reducing the problem of backing up and keeping track of multiple software and scripts, combining all the data collection tools into a single software could also help in the correlation of entities between the different data collected. By combining the software, it may provide better insights on the probe health estimates and failure mode from studying the data (probe cycle count, probe specifications, failure modes) as a whole.

Implementing Aesthetic Look into the Software using GUI-Based Software

At the moment, the tools that have been developed for ICT are still heavily reliant on a Terminal window. While the tools are able to run independently through an executable file (.exe file), there is still poor aesthetic when it comes to the design of the tools. While it may not be a top priority when it comes to internal software (used within the organisation) for important tasks (such as PM activity and probe replacement), improving the design and aesthetic of the tools would give the tools more appeal, especially in use. A suggestion to improve the beauty of the software is to incorporate the software completely in Graphical User Interface (GUI).

During the scope of work phase of the project, it has been explored to make use of different GUI and TUI (Text-based User Interface) Python libraries to make the software more beautiful. However, it is found that by using some of these libraries (*npyscreen* for TUI, *tkinter*

and *Kivy* for GUI), we lose the dynamic template capability that currently exists for the project. In other words, it would not be easy to import or read the steps and questions from an Excel file to the GUI/TUI, as the GUI/TUI window is limited by the window sizing, which is not very flexible with the text it is being given.

Having said that, further investigation and exploration on this possibility, as well as expansion from pure Python tools to potentially using C++ based GUI programmes may be able to circumvent the issues mentioned above. Therefore, there is potential to explore this possibility as a recommendation for the project

Explore Possible Alternatives of Measuring CRES and/or Spring Rate before Full Device

At the end of the internship period, the CRES and Spring Rate Measurement Device are at the quotation stage, whereby the designs and specs for the device have been confirmed and are awaiting order. While the project in itself took much time for exploration, designing, communication between the project team and vendor for the purpose of confirming the scope of work, there is potential in exploring alternative or 'barebone' approaches to measuring CRES and Spring Rate or either one of these.

During the project timeline, there have been efforts put into investigating alternative methods to measuring the CRES of an ICT fixture probe. One of the approaches used is by directly measuring the electrical resistance of the ICT fixture probe using an SMU that is used at the factory. The measurement uses a 4-wire Kelvin Measurement; however, the results prove to be inconsistent, whereby the measurement fluctuate heavily, and a consistent and acceptable result is not found (probes confirmed to be damaged occasionally show to have lower electrical resistance compared to fresh probes that have not been used for production yet).

There is a possibility that an alternative solution could be developed for a cheaper cost and within shorter timeframe. However, when considering these alternatives, there are sacrifices that must be made in terms of the quality of finishing of the device, which translates to the effectiveness of measurement and overall ROI of the project. This possibility is still considered as a recommendation for the project.

8.0 REFERENCES & CITATIONS

Feinmetall Contact Technologies. (2017, June). Catalogue - Contact Probes for PCB Testing .

Kuphaldt, T. R. (2006). *Lessons in Electric Circuits, Volume I - DC (5th Edition)*. Koros Press Limited.

Prechelt, L. (2003). Are scripting languages any good? A validation of Perl, Python, Rexx, and Tcl against C, C++, and Java. *Adv. Comput., 57*, 205-270.

McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, *14*(9).

Podrzaj, P. (2019, August). A brief demonstration of some Python GUI libraries. In *The 8th International Conference on Informatics and Applications (ICIA2019)* (p. 1).

SIP Report_25204_Shahidan bin Idris_EE

ORIGINALITY REPORT

2%	2%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Teknologi Petronas Student Paper	<1%
2	Submitted to University of Maryland, University College Student Paper	<1%
3	Submitted to University of Wales Swansea Student Paper	<1%
4	hdl.handle.net Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	"Trichoderma reesei", Springer Science and Business Media LLC, 2021 Publication	<1%
7	thesis.library.caltech.edu Internet Source	<1%
8	www.ie.boun.edu.tr Internet Source	<1%

9	depositonce.tu-berlin.de Internet Source	<1%
10	www.docme.ru Internet Source	<1%
11	itsolutionpokhara.blogspot.com Internet Source	<1%
12	acadgild.com Internet Source	<1%
13	Min-Jung Son, Minwoo Kim, Taik-Min Lee, Jihoon Kim, Hoo-Jeong Lee, Inyoung Kim. "Mechanical and electrical properties of reverse-offset printed Sn-Ag-Cu solder bumps", Journal of Materials Processing Technology, 2018 Publication	<1%
14	Submitted to University of Leeds Student Paper	<1%
15	www.binaryfolks.com Internet Source	<1%
16	cgspace.cgiar.org Internet Source	<1%
17	agupubs.onlinelibrary.wiley.com Internet Source	<1%
18	Submitted to City University of Hong Kong Student Paper	<1%

19	charactercentral.net Internet Source	<1%
20	www.mdpi.com Internet Source	<1%
21	etheses.bham.ac.uk Internet Source	<1%
22	ulir.ul.ie Internet Source	<1%
23	www.ess.washington.edu Internet Source	<1%
24	www.samtrans.com Internet Source	<1%
25	Min-Jung Son, Jae Won Jeong, Hyunchang Kim, Taik-Min Lee, Hoo-Jeong Lee, Inyoung Kim. "Effect of particle size distribution on the mechanical and electrical properties of reverse-offset printed Sn–Ag–Cu solder bumps", Journal of Materials Science: Materials in Electronics, 2018 Publication	<1%
26	com-devi.fun Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		