DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Development of an Automatic Commissioning Station for Acoustic Emission Sensors

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação da disciplina:

DAS 5511: Projeto de Fim de Curso

Igor Althoff Vidal

Florianópolis, Julho de 2019

Development of an Automatic Commissioning Station for Acoustic Emission Sensors

Igor Althoff Vidal

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Hector Bessa Silveira

____

Banca Examinadora:

Oliver Radestock
Orientador na Empresa

Prof. Hector Bessa Silveira
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Prof. Gustavo Artur de Andrade
Avaliador

Miguel Budag Becker
Debatedor

Gabriel Thaler
Debatedor

Acknowledgements

[EN]

Indtact GmbH gives me one of the greatest opportunities of my life. It is not all the companies that allow a foreign student from a far country to make an internship with, I will be always grateful for this chance and I did my best to make a good job. A special thank you for Oliver that was my supervisor and support me on the many questions about Acoustic Emission.

The WG amazing persons that I lived with during these 6 months. Max, Fiona, Jan, Manu and Nele. You helped me in so many kinds of stuff that It is hard to measure. Thank you to show me the Bavarian way of life. Your friendship makes everything easy, even if it is weird that you do not put ketchup on the lasagna.

The Lebendiges Wort church and all the persons that I met there, in special Feli, Melli, Ricardo and Carmen. You reminded me that there is life beyond the thesis and the world is a fantastic place. We did a lot together and I will remember all these moments.

[PT]

Agradeço à Universidade Federal de Santa Catarina por todo o conhecimento compartilhado. Cresci profissionalmente, mas isso é de pouca valia perto do quanto cresci como pessoa durante esse tempo. Obrigado aos professores do DAS. Nesse documento gostaria de agradecer especialmente ao Prof. Hector Bessa, por ter me orientado nesse período, você é um exemplo de dedicação dentro do departamento.

Meus pais João e Silvia, que sempre incentivaram o extraordinário em mim, não deve ser simples ser pai e mãe, mas vocês são perfeitos. A minha irmã Larissa, por todos os meus projetos de Lego destruídos na infância, me fez aprender a recomeçar, sou muito feliz por ter nascido como teu irmão.

Obrigado aos meus amigos. As nossas histórias continuarão a ser contadas por gerações, ou talvez não. Talvez as histórias nem sejam verdadeiras. Porém contaria tudo novamente, porque vale todo o tempo perto de vocês. Nossa juventude em inicio de ir embora, como os cabelos caindo. Eu estive lá para ver isso, e quero estar para ver cada ruga.

O fantástico da criação, da natureza, a cor do sol, o sorriso que não sei donde veio, minha vida e meu viver, obrigado Deus.

[DE]

Vielen Dank!

Abstract

The company Indtact GmbH, based in Würzburg - Germany, needs a platform to test a new Acoustic Emission (AE) device called smartPredict 2.0 in order to achieve the highest quality in its products. This new device is a easy multi-sensor solution to perform AE analysis and it provides a strong connection with the 4.0 industry. The tests are the last step before sending to the clients, which are usually called manufacture quality control or commissioning. An Acoustic Emission sensor is useful to measure acoustic waves or pulses inside a material. AE analysis is a nondestructive measurement technique and can be used in many applications: monitoring tools, fault detection, control processes, etc. This platform involves software and hardware, with manual and automatic tests. The automatics tests are performed generating a signal and measuring with the sensor in the analysis. Signal processing and statistics techniques are used to analyze the data and check if the sensor meets the commissioning requirements. Using the data measured in the time domain an FFT is generated in which it is interesting to extract a range of informations: global and local peak, global and local peak frequency, mean frequency, and other variables. Using the FFT informations three different tests will be performed: the analysis of the global peak, the FFT local peak and the integral of the FFT test, only if the three tests are correctly verified then the sensor is approved. The three tests will be repeated in many different frequencies. The software is written using different languages/tools, as QT/C++, Python, Html, Json, Labview and others. The main program is written with QT/C++ and the other modules with the other languages/tools. This supervisory is useful to save all the data from the tests and generate reports, also guide the user on the tests. The automatic tests connect with the computer using an Analog Discovery 2 board and the board is also used to generate the signal on a certain controlled frequency. A sound wave generator attached to the same surface of the sensor is the physical platform of the system. This project combines a solution that allows to perform the tests and also save all the data. The platform ensure the company to provide a product with quality for the clients.

Keywords: QT/C++, Acoustic Emission sensor, Test platform, FFT, commissioning, 4.0 industry.

Resumo

A empresa Indtact, sediada em Würzburg - Alemanha, necessita de uma plataforma para testar o novo sensor de emissão acústica (AE), denominado smartPredict 2.0, com o objetivo de alcançar um excelente padrão de qualidade nos seus produtos. Esse novo dispositivo é uma solução simples para realizar análise de AE, é também um dispositivo conectado com a ideias da indústria 4.0. Esses testes serão um último passo antes de enviar o produto ao cliente, esses testes são usualmente denominados controle de qualidade de manufatura ou comissionamento. O sensor de Emissão Acústica é importante para medir ondas acústicas ou pulsos em um material. Análise de AE é um teste não destrutivo e pode ser utilizado em muitas diferentes aplicações: monitoramento de ferramentas, detecção de falhas, controle de processos, etc. Essa plataforma é parte software e parte hardware. Os testes são manuais e automáticos. Os testes automáticos são realizados gerando um sinal com uma espécie de alto falante e medindo com o sensor que se encontra em análise. Técnicas de estatística e processamento de sinais são utilizadas para analisar os dados e verificar se o sensor possui uma boa qualidade e pode ser enviado ao cliente. Utilizando os dados medidos no tempo uma FFT é gerada e é importante para extrair uma variedade de informações, como: pico, frequência de pico, o módulo da FFT, a média da FFT, etc. Utilizando esses dados obtidos da FFT serão realizados três testes: a análise da frequência de pico global, dos picos locais (se houver) da FFT e um teste da integral da FFT, somente se todos esses testes estiverem corretos o sensor está aprovado. Esses testes serão repetidos diversas vezes em diferentes frequências. O software é escrito utilizando diferentes linguagens/ferramentas tais como QT/C++, Python, Html, Labview e outros. Esse supervisório é utilizado para salvar todos os dados dos testes e gerar relatórios, e também auxiliar o usuário durante os testes. Os testes automáticos conectam com o computador utilizando o sistema Analog Discovery 2 e esse sistema também será responsável por gerar um sinal em determinada frequência controlada. Um emissor de ondas de som anexado à mesma superfície do sensor é a plataforma física do sistema. Este projeto combina uma solução que permite realizar os testes e também salvar todos os dados.

Palavras-chave: QT/C++, Sensor de emissao acústica, Plataforma de Testes, FFT, comissionamento, Indústria 4.0.

List of Figures

Contents

1 Introduction

Acoustic Emission (AE) studies the spontaneous release of transient elastic waves [1] in solids as a consequence of sudden localized changes in stress, from 1 Khz to 100 Mhz. Stress in engineering is the consequence of a load under a material. For example, a bridge with an intense traffic, the cars crossing the bridge will stress the structure of the bridge generating intern cracks (Acoustic Emission events), the asphalt needs maintenance to fix this stress consequences. The figure 1 illustrates how an Acoustic Emission event happens, it is the results of the forces or loads under a structure. Many processes in engineering release elastic waves that can be measured and important informations may be obtained. The use of this informations depends on the related problem of interest, for example: as a materials testing tool, as a method for improving our understanding of deformation and fracture processes, as a technique for material process control, etc. Acoustic Emission is a new subject matter of study and enhancements are being continuously developed. To measure these Acoustic Emission events a range of different sensors are used, as Piezoelectric sensors, Gyroscopes, etc. The analysis of the data provided by the AE measurement is not so simple, since Acoustic Emission sensors are highly sensitive to the surrounding environment and also because to understand the important data to be obtained demands experience. One of the great advantages of the Acoustic Emission analysis is the possibility to make non-evasive tests, with minimal influence on the system where the tests are being performed.

iNDTact GmbH is a German company which develops different technological solutions to measure AE events. iNDTact GmbH is a relatively new company (founded in 2013), and lately has been able to increase its size as well as the number of clients.The Indtact company already used its sensors in the areas of: Manufacturing, Construction, Automotive, Aviation, Energy and Medical. The clients of iNDTact are important European companies as: Siemens, DB, Airbus, Porsche, Eesa, JTC, Rexroth, etc. New products are being developed every day. A new sensor called smartPredict 2.0 is under development and the expectation is that when released the sales will be a success, since the smartPredict 2.0 is more simple to use than other AE sensors available on the market. The company currently is concerned about how to maintain the quality of the products with this potential high demand in the near future. smartPredict 2.0 also provide LoT connection, this shows a 4.0 industry relation with iNDTact GmbH.

The quality of a product is directly connected with level of reliability a client has on the featured tasks of the device. A reliable product is mandatory for robust applications.

[1] "Elastic wave, motion in a medium in which, when particles are displaced, a force proportional to the displacement acts on the particles to restore them to their original position." [1]

A sensor is a kind of product that can be used in several different systems. A temperature sensor can be used, for example, in a simple kitchen oven or in a nuclear boiler. The latter needs a robust and reliable sensor or a major disaster may happen. The question is: how can a company develop a reliable product? The answer is simple, testing the product before sending it to the client, this will increase the probability that the product will work adequately on all the range of different applications. Therefore, quality control procedures should detect all the possible failures that may occur on the manufacturing line. On an embedded system, this can be, for example: bad wire connections, wrong version Firmware installed, a capacitor with the wrong capacitance, etc.

Commissioning is concerned with ensuring that the performance, functionality and features of the final products meet the design specifications. [2]. The company iNDTact GmbH already has a commissioning procedure, but in which an employee performs manual tests on the sensor and records manually all the data in a simple Excel file and manually produces a report in a PDF file. This manual commissioning procedures demands a considerable time and it is more susceptible for mistakes from the employee. The Acoustic Emission device smartPredict 2.0 will be a complex system with the possibility to read many different variables, this will increase the challenge to manually test the range of variables from the device. As previously mentioned, the company's expectation is that the new smartPredict 2.0 AE product under development will have high demands from the market. Therefore, a new commissioning procedure which is highly automatized is currently of major interest for the company.

One method to enhance manufacture quality control tests of produced sensors is to test if they meet the required technical specifications by emulating its operating environment and then verify the reliability of the measured data provided by the devices. This should be done in the production line before delivering the products to the clients. An adequate emulation of the operating environment may be achieved by generating a wave on the surface of the sensor in a controlled manner. Such wave will then propagate through the surface. A sensor attached to this structure can measure the wave. The wave can be transformed to an interesting domain, as the frequency domain. The analysis on the frequency domain of a sensor working properly must find the same frequencies that was generated with the addition of some harmonics. The harmonics happens when the wave propagates through the structure and it has contact with the structure material. The goal of the analysis on the frequency domain is to find similarity between the signal generated and the signal measured and verify if the similarity is on a certain range.

A supervisory system is mandatory, since the procedures must be as simple to the employee performing the tests as possible. The supervisory will be responsible for many tasks: help the employee to perform the tests, record the data, generate automatic PDF reports, call the automatic tests, take the ID of the sensor, the data about the environment of the tests and other informations.

The project is surrounded with different challenges. Since Acoustic Emission analysis is a relative new subject matter of study there are few background about commissioning procedures for AE sensors. The standards ways to generate a wave on the structure are with no replicable techniques (e.g.: a ball falling on the structure), a replicable wave is mandatory since the products must have the same test environment to ensure the same reliability. The automatic tests are not described in the AE literature, an analysis can be made in different ways and different domains. The supervisory development depends on the tools that iNDTact Gmbh has some background, the joint of the tools that are interesting for this project and the tools that the company has background demands a complex supervisory system with different software tools. For example, Labview was used on the supervisory, but the connection with the other modules of the supervisory using Labview was a challenge, because Labview does not provide a simple way to communicate with other software tools. Also, the time constraints is considerable, 6 months is a short period for develop this complex project, the new smartPredict 2.0 is under development and can not be used for tests, this induce the system to be as generic as possible, in this case a generic system increase the complexity. The main goal of this project is to provide a functional way to test the sensor smartPredict 2.0 and save all the tests data. This document will explain all the solution in technical aspects.

The document is divided in the follow way: contains a chapter explaining the problem (chapter 2), a chapter providing a theoretical background about the subjects of this system, the software, hardware and signal processing subjects (chapter 3), a chapter discussing what was the solution, how the automatic tests was developed and the high level of the supervisory solution (chapter 4), a chapter containing the results and what was implemented, as the low level of the supervisory (chapter 5) and a conclusion (chapter 6).

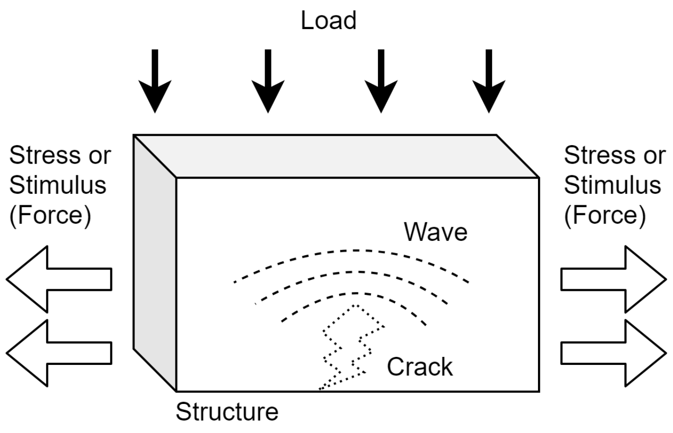


Figure 1 – Acoustic Emission events are generated from stress under the material. The stress is consequence of the forces and loads on the material.

2 Acoustic Emission and Indtact GmbH

This chapter will provide an introduction about the environment of this project. How is the Acoustic Emission analysis, what is Acoustic Emission events, where Acoustic Emission analysis can be made (section 2.1). The iNDTact GmbH company, what are the products developed by the company, what clients the company works, how is the company team, what is the new smartPredict 2.0 sensor (section 2.2). The last section (section 2.3) provide a detailed explanation about the project goals and requirements.

2.1 Acoustic Emission: overview

Acoustic emission is the class of phenomena whereby transient elastic waves are generated by the rapid release of energy from a localized source or sources within a material, or the Acoustic Emission aspects transient elastic wave(s) so generated [3], the *Introduction* chapter provide an example about Acoustic Emission, it is the explanation about why a bridge with intense traffic generates acoustic emission events inside the structure (see figure 2).

Acoustic emission testing is a nondestructive test capable to detect and locate faults in mechanically loaded structures [4]. It is a kind of new subject of study started by J. Kaiser in the 1950s. It is originated by deformation, caused by stress, emitting an acoustic emission, usually on the range 150 - 300 Khz, which is above the frequency of audible sounds. Using a sensor, filters and amplifiers it is possible to find an area of interest to analyze. There are many possible techniques to analyze an AE signal and the technique will depend on the kind of application is under interest. Figure 2 (figure adapted from [5]) shows an example of acoustic emission event and what comes next. The first step is a crack generated inside a structure, it is the consequence of a stress inside the structure. The following step is the propagation of this crack through the structure, in this step the original crack in contact with the structure will change, this is the addition of the harmonics to the original crack. The next step is the measurement of the crack propagated trough the structure, an AE sensor attached to the same structure read this signal, a sensor will never provide a perfect measurement, the signal is then modified because of the sensor characteristics. The last step is amplify and filter an interesting area to analyse. When comparing the first step with the last step (figure 2) the differences are visible, the original crack is inside the last step but to extract this original crack from the last signal a range of different tools must be used (e.g.: Fourier transform). This signal propagation example illustrates the challenge that is analyse this Acoustic Emission events to extract relevant informations.

The Acoustic emission analysis can be used in:

- Proof testing;
- Online condition & structural health monitoring;
- Leak detection and localization;
- In process weld quality monitoring;
- Mechanical property testing & validation;
- Diagnosis of the integrity of large structures;
- Tool monitoring;
- Quality inspection of bonding in composites and laminated structures;

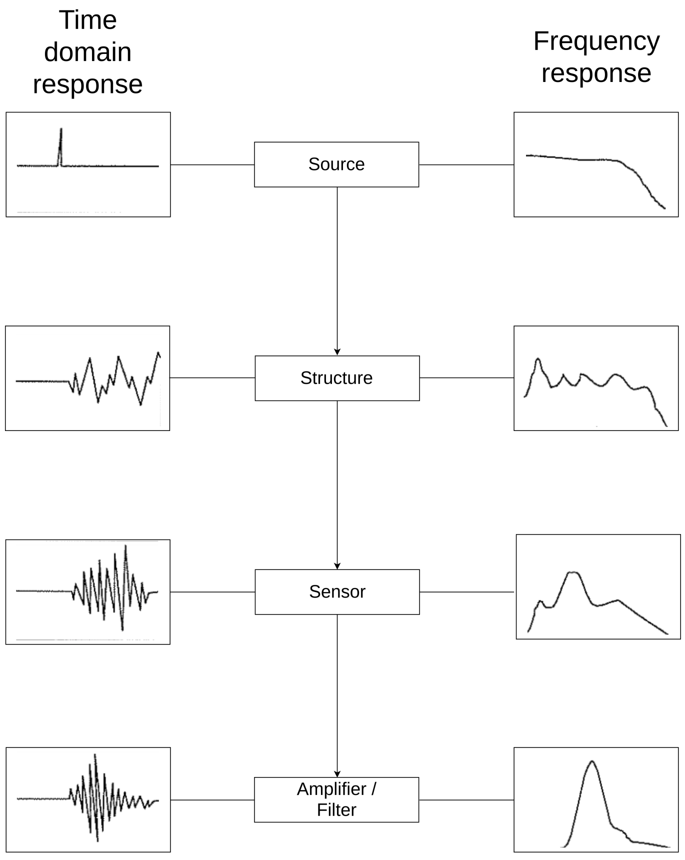


Figure 2 – Signal shaping chain. The original signal is changed while crossing the material. The left column is the signal in time, the right column is the signal in the frequency domain. To analyze the first signal is necessary to filter and choose an area of interest.

The arrangement of the devices in an Acoustic Emission analysis can be found in figure 3, this is the standard physical setting to perform Acoustic Emission tests. A computer, an amplifier and a sensor are used to measure the crack generated inside the structure. As previously explained in the figure 2 the original crack propagates through the structure and the signal is changed, each step from the figure 2 represents one device from the figure 3. The difference between the two figures is the presence of a computer, which will check the signal. The computer reads all the raw input, obtained from the filter/amplifier, and transform this data using different tools: FFT, Bode plot, time plot, etc. The best tool will depends the kind of analysis that is being performed. The computer output helps the researcher or an Artificial Intelligence to find interesting aspects of the data.

One interesting example of AE analysis is to detect failures on the manufacturing floor. On the manufacturing floor usually there are several motors, performing a variety of tasks. An employee is in charge to check if the motors are working properly, he knows when is time to re-calibrate or change some components from the motors. However, automatize this task may prevent mistakes that can be expensive. A solution is using AE analysis. An Acoustic Emission sensor can be attached to the motor and measure the Acoustic Emission events, analyzing these events provide the possibility to know when will be necessary maintenance. These results can be sent to the factory network and anyone can read. This example also shows a strong connection between Acoustic Emission analysis and the 4.0 industry.

The common techniques to measure AE events involve the use of piezoelectric effect (details about piezoelectric at section 3.8) and optical interferometry. The standard way is using the Piezoelectric sensor, it is used for flex, touch, vibration and shock measurement [6]. [1]

Difficulties in extracting the information from the acoustic signal have been one possible reason why process AE monitoring is not more widespread [7]. The Acoustic Emission analysis is highly affected by the environment, any vibrations on the surroundings appear on the measured signal. However, the improvement of filter techniques, new Artificial Intelligence technologies, etc are upgrading the Acoustic Emission analysis.

[1] Indtact GmbH system uses piezoelectric sensors, more details at section 2.2.

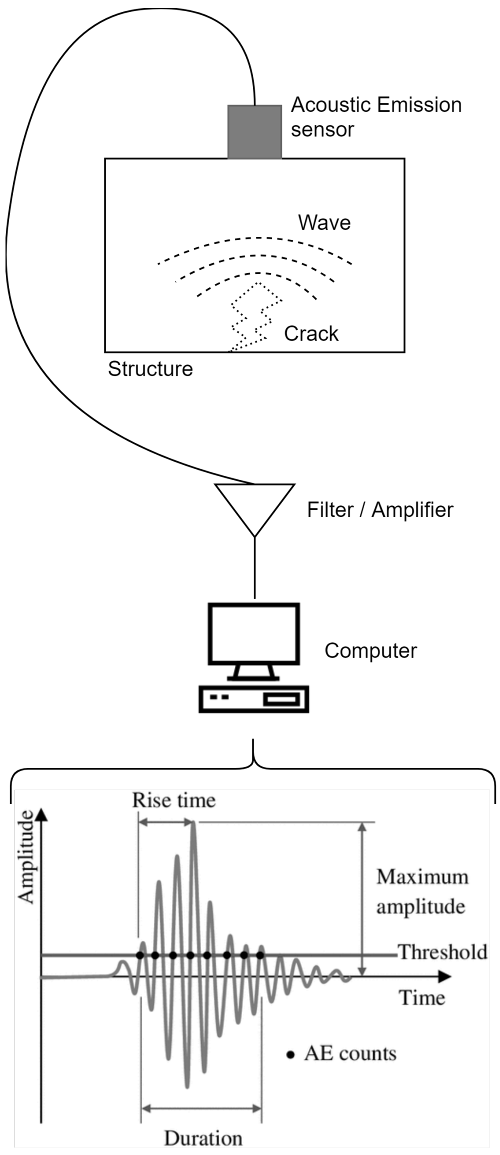


Figure 3 – Example of measuring and analyzing an AE event. The sensor is attached to the surface where the AE happens. The data filtered by the amplifier is read by the computer where can be transformed into any interesting domain.

2.2 Indtact GmbH

The Indtact GmbH company (Figure 9) is based in Würzburg / Bavaria / Germany (with a sales partner in Singapore), it was founded in 2013 [8] as a small startup with products focused in Acoustic Emission analysis. The company develop amplifiers (Figure 7), piezoelectric sensors (Figure 10) and many other products. The company also can be hired to make Acoustic Emission technical analysis and it generates technical reports about the analysis. The product that differ iNDTact GmbH company from other Acoustic Emission companies on the market is a device called smartPredict (figure 10) that is a a multi-sensor system comprising complex parameter recording, digitization, data conditioning and analysis. It is possible to read with the smartPredict device the following information:

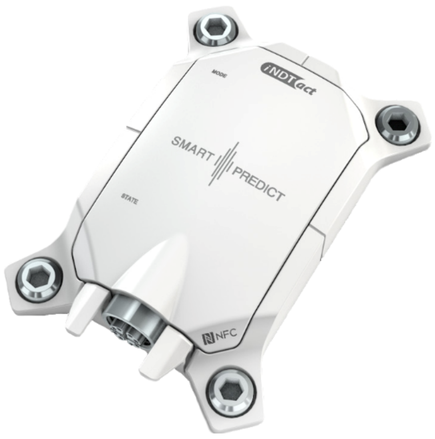
- Acoustic Emission;
- Acceleration;
- Angular velocity;
- Temperature;
- Magnetic field;

Acoustic Emission analysis is sensitive to many different variables, even the temperature, for example, must be considerate on the AE analysis. A good Acoustic Emission analysis must treat all the environment variables. The main difference between iNDTact GmbH and other companies is that the iNDTact solution already provides the possibility to read all the important variables in only one device. A simple piezoelectric sensor can measure from 1Khz to 1Mhz, this range is not enough for a complete analysis, so other sensors must be used to increase the AE analysis. The smartPredict join all this sensors in one single device, this gives a simple way to perform a complete AE analysis. The figure 6 shows the combination of different sensors range and the results is the complete range of the device smartPredict.

The Indtact company already used its sensors in the areas of: Manufacturing, Construction, Automotive, Aviation, Energy and Medical. The clients of iNDTact are important European companies as: Siemens, DB, Airbus, Porsche, Eesa, JTC, Rexroth, etc. Actually, iNDTact has around 20 employees, each one with different backgrounds as: physics, Electrical Engineer, Software Engineer, Business, Geophysics, and others. The company works with a technology that daily became more popular and it is directly connected with the 4.0 industry.

The new smartPredict 2.0 (figure 4 and figure 5) is an Acoustic Emission device in which upgrades many of the features (e.g.: is more simple to setup, gives LoT connection, etc) from the last products. Also, it provides many different AE variables to measure as the smartPredict. The sensor was developed based on the feedback of the clients during these 6 years iNDTact GmbH in the market. The range of smartPredict 2.0 device is between 0.06Hz - 48KHz [2]. The smartPredict 2.0 allows performing the measurements easier than the other sensors on the market. Between the advantages of smartPredict 2.0 is that all the digital parts are already integrated with the analog inside the device and the device is smaller in size than the smartPredict.

[2] The measuring range of the smartPredict 2.0 is far from all the Acoustic Emission Events range (0.01Hz - 1MHz), but it is enough for the kind of market that iNDTact GmbH works.



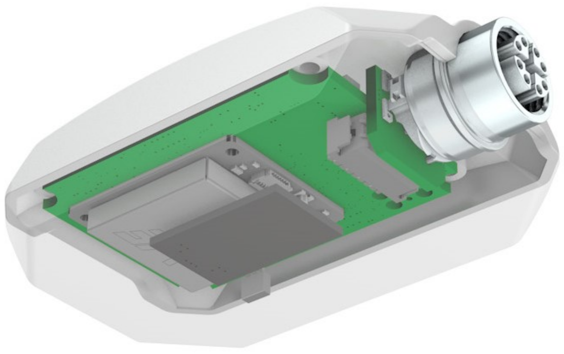


Figure 4 – new smartPredict 2.0: front view Figure 5 – new smartPredict 2.0: inside view

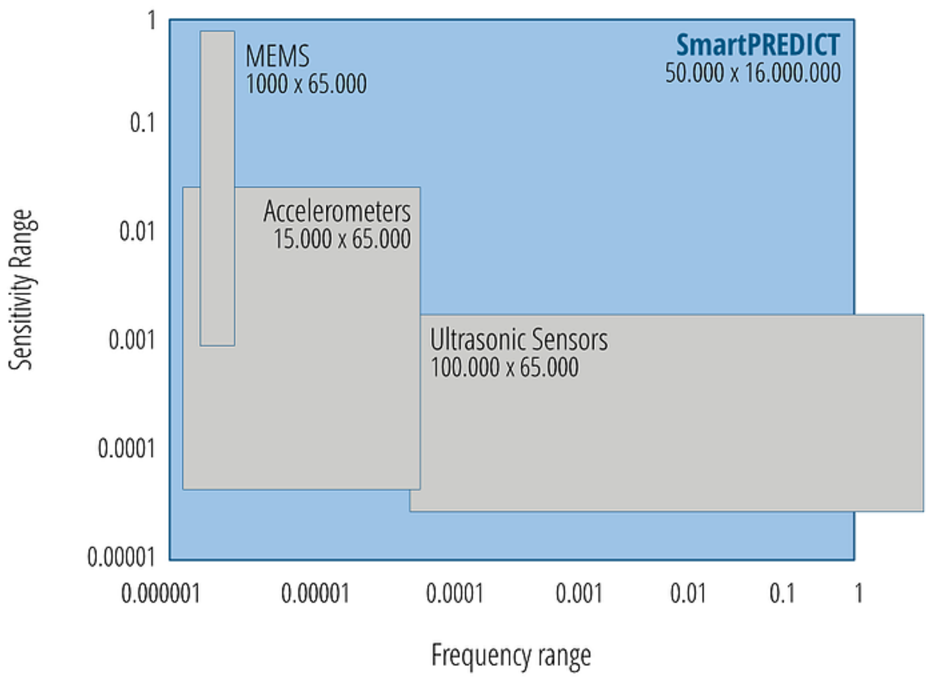


Figure 6 – iNDTact SmartPredict range comparison. The smartPredict device is the combination of different sensors that are interesting to measure AE events. The smartPredict provides a full solution to measure AE events.

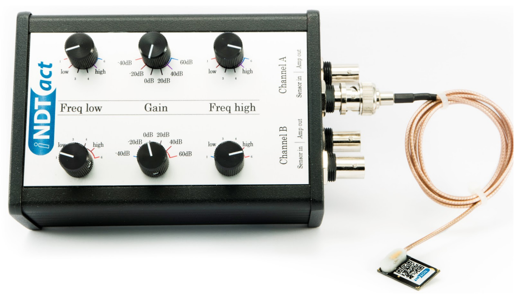


Figure 7 – champs 1.1 - iNDTact Amplifier. It was used during the tests to filter the signal measured from the AE sensor.

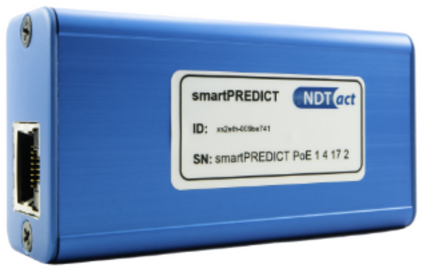


Figure 8 – SmartPredict sensor. This is the first device that join all the AE interesting sensors. The figure 6 shows the measurement range of this sensor.

Figure 9 – iNDTact GmbH logo

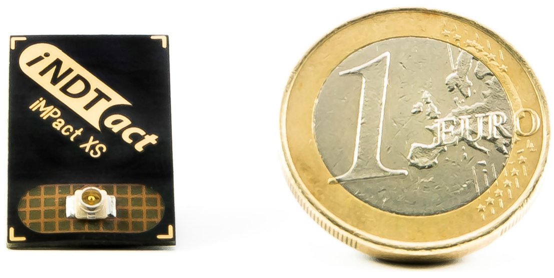


Figure 10 – imPact XS sensor. A sensor developed from iNDTact GmbH in which allow measuring Acoustic Emission signals. This was used to generate the sound and to measure signals during the project.

2.3 Problem Description

The Indtact GmbH company is developing a new smartPredict 2.0 acoustic emission device, because the device is more simple to use and robust than other similar devices on the market the company has high expectations about the sales of this new device. This new device will be manufactured by another company and it needs to be tested before send to the final client. The tests will ensure the quality of the product. This is commissioning, as explained on the introduction. Between the development of a product until send to the final client there are many steps, the commissioning will be the last step before send to the client, this flow is explained at figure 11.

The Indtact Gmbh demands an automatic platform to the commissioning of the smartPredict 2.0. The functionalities of the smartPredict 2.0 must be tested. The complexity of the device, with many variables, make this commissioning procedures a challenge. The platform must provide methods to analyse each of the sensor before send to the client. This methods must be enough robust to identify errors at the assembly of the smartPredict 2.0. However, this methods are not described on the literature. The question that this project needs to answer is: how identify errors on this complex device?

The platform must also guide the user during the tests. The company wants to test the devices quickly as possible and a supervisory will provide a way to visually guide the user. The screen of the supervisory must provide the informations on an ergonomic setting to easily guide the user.

Another mandatory function of the platform is to save all the informations from the tests on a database. This is important as a record to future analysis of the tests. Also, generate automatic PDF reports.

The problem description can be organized on a list of requirements:

- Build a platform to perform the tests:
  - The platform must make the tests as fast as possible;
  - The platform needs to be the most automated as possible;
  - The platform must save all the data;
- Develop a Step-by-step test procedure;
- Analyse the PCB sensor and think about possible necessary changes;
- Develop a supervisory program to assist the employee that is performing the tests:
  - The supervisory must have a connection with a database;
  - The supervisory must have automated tests but also will guide the employee in some manual procedures;
- Make a Failure Tree Analysis on the sensor system;
- Implement a platform that will perform automated tests:
  - The platform needs to generate a signal in controlled frequency;
  - The platform must understand if there is an error;
- The tools used on the project must be usual to the company;

As previously explained in the *introduction*, the platform is surrounded with challenges. There are not so much literature about Acoustic Emission and commissioning procedures. The new smartPredict 2.0 is highly complex and with many different variables to test. There is no conclusion about the best tool to perform the automatic tests (e.g.: tima analysis, bode analysis, Frequency analysis, etc). The way that the platform must generate a signal to test is uncertain. The supervisory is also a complex system, since the tools and libraries must match the company background, but at the same time the company background does not fit with the best solution. The solution is showed on the chapter 4, after a theoretical background provided from the chapter 3.

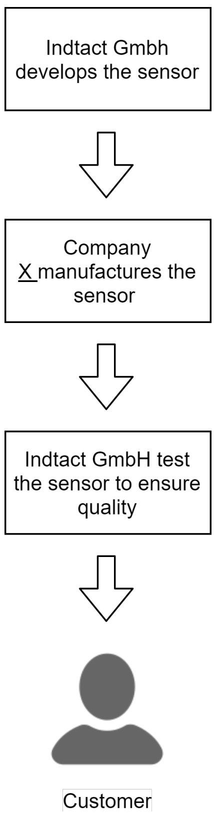

Indtact Gmbh develops the sensor
Company X manufactures the sensor
Indtact GmbH test the sensor to ensure quality
Customer

Figure 11 – Project flow explanation. The project aims to test the sensors before send to the costumers leading to a better quality of the Indtact devices. The sensors that are not good must be discarded.

3 Theoretical background

It is important to provide a theoretical introduction to the topics of this project. Many tools were used from different areas, as: software engineering, computer languages, Fourier transform, Cross-correlation, etc. The solution can be hard to understand if a theoretical background of the subjects is not provided. The main objective of this chapter is not provide an extremely detailed description of each tool, because some of the tools demands a book to explain (e.g: Fourier Transform), but a short introduction to help the reader. All the sections and subsections provide an example, to simplify the understanding.

The section 3.1 explain about the software tools used during the development, is divided in the subsections: QT (subsection 3.1.1), Absolute and relative path (subsection 3.1.2), Json (subsection 3.1.3) and Labview (subsection 3.1.4). The section Industry 4.0 provide a background about this new industry concept and the company Indtact (section 3.2). Section 3.3 tells about the software methodology used, the Rapid Application Development. The section 3.4 is about Fault Tree Analysis. The Analog Discovery 2 and other boards are detailed on the section 3.5. The sections 3.7 and 3.6 talks about Cross Correlation and Fourier Transform. The last section 3.8 clarify about the Piezoelectric effect and sensor.

3.1 Computational aspects

3.1.1 QT

Developed by two Norwegian software engineers Haavard Nord and Eirik Chambe-Eng, the Qt framework first became publicly available in May 1995 [9]. It is a framework multi-platform graphic interface to C++. Allow to develop libraries and applications only once and use in different systems. Enable you to develop to the following platforms: Windows, Windows CE, Mobile, Symbian, OS X, X11, Embedded Linux, Maemo, MeeGo, QNX / BlackBerry 10, Wayland.

The QT creator is a GUI developed to help to code in QT. It is easy to develop an app with QT creator and QT. The QT is open-source and free with GNU license, so it is free for internal projects and open source projects. Some of the advantages are on the subsections below.

3.1.1.1 QT creator

QT creator allows the programmer to easily create Widget applications. For example, to create a program to read a Dial and send the value to a progress bar is simple. Open the *forms > mainwindow.ui*. Insert a Dial, a progress bar and a button, just selecting and dropping from the widgets screen. Automatically the QT will set the properties at the header *(.h)* file. Right-click at the *button widget > go to Slot... > clicked ()*. With this QT will set the header and the source to always that this button is clicked a function is called (QT call SLOT), inside this function, it is possible to set an action. In this example, the program will get the *int* value from the dial and set the progress bar with this value. So, always that the button is clicked will update the progress bar (figure 12).

```
void MainWindow::on_pushButton_clicked()
{
    int value = ui->dialExample->value();
    ui->progressBarExample->setValue(value);
}
```

Algorithm 3.1 – QT creator example

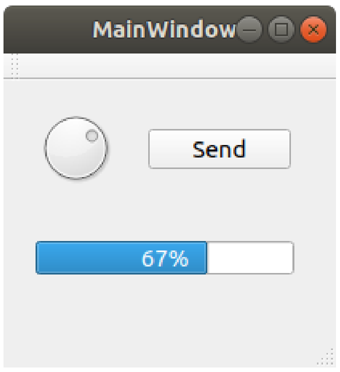


Figure 12 – QT widget example. The dial can be manipulated and each time that the button is clicked the progress bar is updated with the value of the dial.

3.1.1.2 Message between objects

One of the most important advantages of QT is the way that objects can communicate with each other. It is call *connect* function [10]. To introduce this function the same example as the last subsection is used, but now always that the dial change the value of the progress bar will also change, without the button action. For this task there is only one change on the main code, it is just set *mainWindow.ui right click on the Dial > go to slot... > valueChanged(int)*. QT will set the header and source to always that dial change something happens. In this case, it will send a SIGNAL.

```
void MainWindow::on_dialExample_valueChanged(int value)
{
    emit sendValueFromDial(value);
}
```

Algorithm 3.2 – SIGNAL definition

Now it is necessary to connect the SIGNAL with the function (SLOT) to change the value at the progress bar, the way to do this is as follows.

```
[...]
connect(ui->dialExample,SIGNAL(sendValueFromDial(int)),
        ui->progressBarExample, SLOT(setValue(int)));
```

Algorithm 3.3 – Connect definition

It is also important to set the header (.h):

```
[...]
signals:

    void sendValueFromDial(int);
[...]
```

Algorithm 3.4 – SIGNAL on header

3.1.2 Relative and absolute path

A program often needs to open a file or even another program, there are two ways to set the path of a file, by relative or absolute path. Each one has different purposes.

3.1.2.1 Relative path

A relative path is a way to specify the location of a directory relative to another directory. The path must be specified with just . and \. To pass the path of the very working directory is .\ and to come back one directory on the tree is ..\. For example, to indicate the file *text.txt* and the working directory is Debug. The relative path will be .\..*DirectoryWithText**text.txt*.

```
/
└─ Home
   ├─ DirectoryWithText
   │  └─ text.txt
   ├─ Debug
   │  └─ example.exe
   └─ Other directory
```

3.1.2.2 Absolute path

An absolute, or full, path begins with the drive ID followed by a column, such as *C:*. For example, using the example above *C:**Home**DirectoryWithText**text.txt*. This approach to set a path is not good for a professional application, once the Absolute path is different for all the platforms that the application is being used.

3.1.3 Json

It is a data-interchange format developed to be easily readable by humans and machines. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999 [11]. It can be used in many different languages. It is written as a dictionary, the ID of the object indicates some array, number, string or object.

The example bellow sends two different messages, with different data inside each message. The *message 1* contains strings, *message 2* contains strings and one array with strings.

```
{
    "Message1": {
        "description": "Some description 1",
        "message": "Some message 1"
    },
    "Message2": {
        "description": "Some description 2",
        "message": "Some message 2",
        "options":["1","2","3"]
    }
}
```

Algorithm 3.5 – Json example

Json will be an important tool on this project, since many modules will share data. For small databases a Json can also be useful, this advantage will be used on this project. The advantage that Json brings to the project is the possibility to easily insert new manual or automatic tests to the system.

3.1.4 Labview

It is a graphical development environment. Developed by National Instruments [12]. It is usually used for: data acquisition, Instrument Control and Industrial Automation.

The Labview has a user-friendly interface in which allow researchers that do not have programming experiences to make programs. The iNDTact company uses mostly Labview in its applications. Because it is not open source and Non-textual Labview has critics from the engineers, also it is not managed or specified by a third party standards committee such as American National Standards Institute (ANSI), Institute of Electrical and Electronics Engineers (IEEE) or International Organization for Standardization (ISO). The example at the figure 13 shows a list that sum with each component of other list generating an array output. It is mandatory to use Labview on this project, since Indtact has many employees with a background in Labview. The use of Labview will ensure a long life of this project on the company.

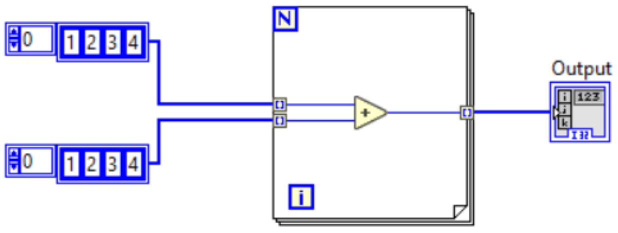


Figure 13 – Labview example. The two squares are arrays, each element of the array will sum, the output is then sent to the variables visualization area of Labview.

3.2 Industry 4.0 and Indtact GMBH

During the centuries happened several changes in the industry. The 1.0 industry comes with the mechanization of the factory, 2.0 industry with electricity, 3.0 industry the digitalization and use of the computers [13]. Actually, the world is going through the Industry 4.0, based on an advanced digitalization within factories, the combination of Internet technologies and future-oriented technologies in the field of "smart" objects (machines and products) seems to result in a new fundamental paradigm shift in industrial production. The high amount of data that the factories produce every day, using sensors, now can be easily shared with computers and the computers can take decisions. All this data was hard to process by humans. Other interesting things that Industry 4.0 revolution will bring are: Optimize logistics and supply chains, Autonomous equipment and vehicles, Robots, Additive manufacturing (3D printing), Internet of Things (IoT) and the cloud. Some numbers already confirm that the Industry 4.0 revolution is good. $28 billion is the amount of expected cost reduction in the automotive sector between 2016 and 2020 due to Industry 4.0. 30% is the increase in productivity that could be achieved by first wave IoT adopters [14].

The iNDTact GmbH matches completely with the Industry 4.0 paradigms. The Acoustic Emission solution brings connectivity with the Internet of Things. The data that the sensors produce can be used to computers take decisions. The Industry 4.0 paradigms can be observed in what the company already implemented using its products: Monitoring of cogeneration units (CHP – combined heat and power plants), Bearing monitoring in fans (integrated iMPactXS Unit), Bearing monitoring in large-scale industrial fans and turbines Process supervision of fully automatic assembly line of electric motors CM of building technology in high-rises.

3.3 Rapid Application Development

On this project, the good programming practices have been used. "The ability to comprehend a program written by other individuals is becoming increasingly important in software development and maintenance. Studies have shown that 30-90% of software expenditure is spent on maintaining existing software. Studies have also shown that maintenance programmers spend about half of their time studying the code and related documentation" [15]. An important thing is to code a readable and rewritable code.

A methodology for software development is mandatory. Without a software methodology, all the development will be in risk. There are steps in which a software must follow and they are important to reach the goal to provide a functional supervisory system. Choose the correct methodology will lead to the success of this project. There are thousands of methodologies, the correct one will depend on the characteristics of the project. This project has some important characteristics: the development time is short (less than 6 months), the team is basically one member and the final user will weekly suggest upgrades and give feedback.

The Rapid Application Development was written by James Martin in 1992, is defined as high-quality systems, fast development and delivery and low costs. These objectives can be summed up in one sentence: the commercial need to deliver working business applications in shorter timescales and for less investment. It was created as an alternative to the cascade methodology, that is still really famous. It is basically the continuous improvement of the system and the continuous feedback from the final user until reach a final system that is satisfactory for the requirements. The advantages are that the project can be separate in small tasks and each of these tasks can be approved by the final user. As shown the Figure 14 with the steps of the Rapid Application Development, first the requirements are defined with the costumer and then, on the circle, the continuous development until reach a final system. After the circle, there is still the testing and debug/share of the system.

Rapid Application Development was chosen because of the characteristics of the project, one developer and weekly meetings with the final user, on this project the final user is the own company. Also, with a project separate in small tasks, the progress was more easy to visualize and also easy to change in case of error.

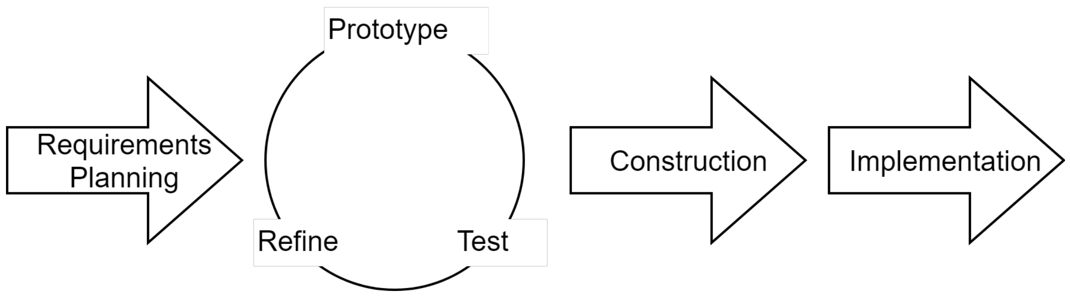


Figure 14 – Rapid Application Development diagram. Before the circle is the requirements steps. On the circle is the part in which the implementation and development are made, it is also this circle steps that differ the methodology from others, providing a quick way to develop. After the circle the tests and validation.

3.4 Fault tree Analysis

FTA (Fault Tree Analysis) is a top-down method to analyze the undesirable states on a system. Basically indicates on the top of the diagram an error and inside the tree is located the events in which move to the fail. It is used in the area of safety engineering and reliability engineering. The main purpose is: understand how systems can fail, identify the best ways to reduce risk, determine event rates of a safety accident or a particular system-level failure.

The diagram is made of three different shapes, using Boolean logic. The figure 15 shows a FTA example. The system is an LED blinking, looking at the FTA diagram it is possible to go through the events and observe that the failure can have 3 possible reasons: LED is wrong polarized, power supply fail or Firmware error.

On this project FTA analysis was used to understand what are the fails that can happen on the smartPredict 2.0. What are the fails that lead to the malfunction of the system. With this information's it was possible to think about the kind of tests that will be necessary to perform to identify these errors. The FTA of the system was built talking with the engineering team and also seeing the schematics of the new sensor. The FTA will not be discussed in further chapters, but it was extremely important to the understanding of the problem.

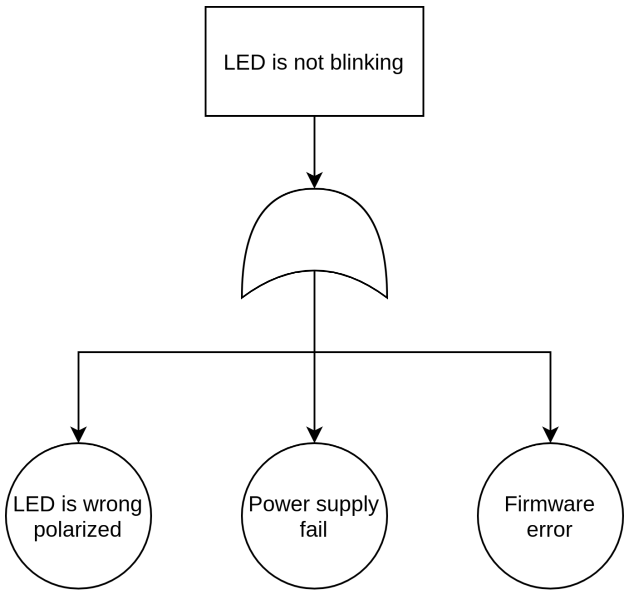


Figure 15 – FTA example. The error showed in the top of the tree can happen because of three reasons. These reasons appear on the bottom of the tree.

3.5 Analog Discovery 2 and BNC board

The Analog Discovery 2 (Figure 16), from the Digilent company, is a multi-function instrument that allows you to make several engineering tests using a small equipment connected to a computer. It can be used, for example, to take data from a sensor and/or to generate waves on a controlled frequency. The features include (source: [16]):

- Two-channel oscilloscope (1MΩ, +/- 25V, differential, 14-bit, 100MS/s, 30MHz+ bandwidth - with the Analog Discovery BNC Adapter Board)
- Two-channel arbitrary function generator (+/- 5V, 14-bit, 100MS/s, 12MHz+ bandwidth - with the Analog Discovery BNC Adapter Board)
- Stereo audio amplifier to drive external headphones or speakers with replicated AWG signals
- 16-channel digital logic analyzer (3.3V CMOS, 100MS/s)1) 2)
- 16-channel pattern generator (3.3V CMOS, 100MS/s)3) 4)
- 16-channel virtual digital I/O including buttons, switches, and LEDs – perfect for logic training applications 5) 6)
- Two input/output digital trigger signals for linking multiple instruments (3.3V CMOS)7)

- Two programmable power supplies (0...+5V , 0...-5V). The maximum available output current and power depend on the Analog Discovery 2 powering choice:
  - 250mW max for each supply or 500mW total when powered through USB
  - 2.1W max for each supply when powered by an auxiliary supply
  - 700mA maximum current for each supply
- Single channel voltmeter (AC, DC, +/- 25V) Network analyzer – Bode, Nyquist, Nichols transfer diagrams of a circuit. Range: 1Hz to 10MHz
- Spectrum Analyzer – power spectrum and spectral measurements (noise floor, SFDR, SNR, THD, etc.)
- Digital Bus Analyzers (SPI, I2C, UART, Parallel, CAN)

Analog Discovery 2 also provides an external add-on board. The figure 17 shows one of these boards, which provides a way to connect BNC cables to the Analog Discovery to use with the waveform generator or the oscilloscope. The Analog Discovery 2 was also a requirement from the company, since they have many of these boards and other internal projects also use the same board. The Analog Discovery 2 will be responsible to be the connection between the computer, the sensor and the wave generator. So, it will generate a controlled signal and also read the raw data.

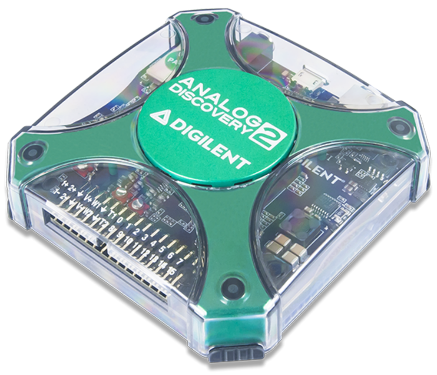


Figure 16 – Analog Discovery 2. This board is responsible to generate the signal and also measure the signal making an interface between the analog devices and the computer.

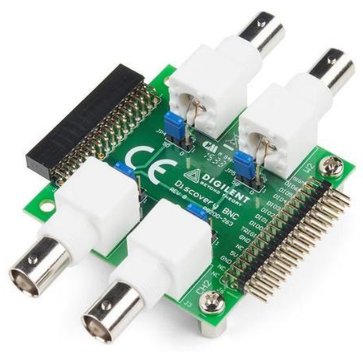


Figure 17 – BNC board. It is an external device that goes attached to the Analog Discovery 2. Allow connecting BNC cables (oscilloscope standard cables) to the Analog Discovery 2.

3.6 Fourier Transform and FFT

Sometimes the most intuitive way to see raw data do not show something interesting. For example, the time plot in figure 18 does not show anything special, it looks like a noise signal. But, when the data is transformed to the frequency domain it is possible to observe two big frequencies acting on the signal. A Fourier analysis is an interesting approach to observe specifics signals characteristics. For example: allow measuring the predominant frequency on a certain sample. The idea behind the Fourier transform is that any signal can be represented using only trigonometric or exponential functions. The math behind is not complicated but it is not the main goal of this project describes in small details the low level of the Fourier transform.

The Fast Fourier Transform it is a way to transform a time sample in frequency domain [17]. It is a computational optimized Discrete Fourier Transform (DFT) and it is really important at signal processing, allow to make the Fourier transform more easily. The DFT is defined as [18]:

$$F(u) = \sum_{x=0}^{N-1} x_n \cdot e^{\frac{-i2\pi k \cdot n}{N}} \tag{3.1}$$

where x_n is a sequence of N complex numbers $\{x_n\} := x_1, x_2, x_{N-1}$. k is used to describe the frequency domain ordinal. Some variables are important when creating an FFT. The sample rate (*fs*), is how many times per second the acquisition board will read the data, on this project the Analog Discovery 2 sample rate is set 100 Khz. The bandwidth or Nyquist frequency (*fn*), in which is the maximum frequency that the FFT can measure is $fn = \frac{fs}{2}$ [19].

Using python, for example, it is possible to perform a FFT with one single line of code. The code line *yf = scipy.fftpack.fft(y)* make the FFT. The *scipy* library must be called before. The figure 18 shows the output.

The Fourier transform is important to the Acoustic Emission analyses, since the frequency is the signature of many processes on the engineering. With the possibility to read this signature it is possible to observe if the process is working fine. For example, if a motor turn on a velocity of 100 Hz is expected that the Fourier analyses of the Acoustic Emissions of this motor appear a peak at 100 Hz (and other harmonics).

The signal to perform an FFT is assumed to be a periodic finite signal. In case a leak happened on the limit of this periodic signal high frequencies appear on the FFT. As said [20]: "When the number of periods in the acquisition is not an integer, the endpoints are discontinuous. These artificial discontinuities show up in the FFT as high-frequency components not present in the original signal. These frequencies can be much higher than the Nyquist frequency and are aliased between 0 and half of your sampling rate". These high frequencies must be avoided. To minimize these effects a technique call windowing can be used. Windowing is basically multiplying the time signal with a function that smoothly goes between 0 to 1. On this project, a Hanning window was used (Figure 19).

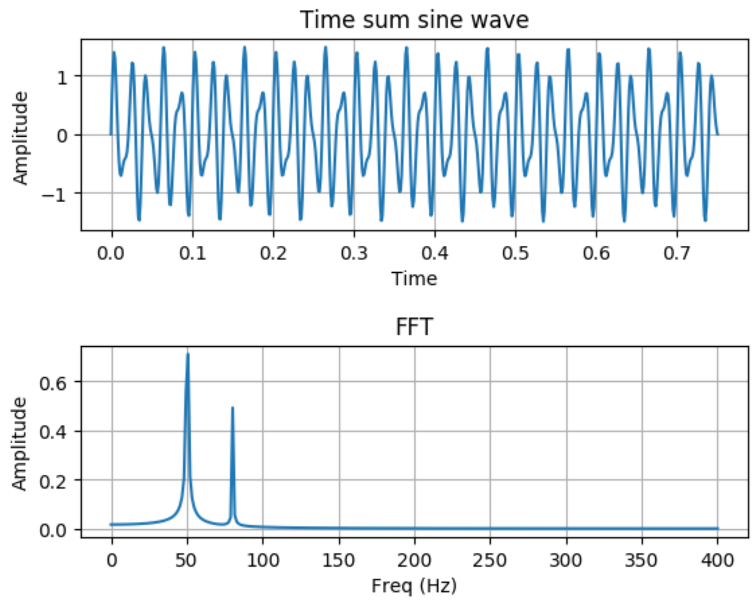


Figure 18 – FFT example. Allow observing the signal from another point of view, on the frequency domain. a) The signal in the time domain. b) The signal in the frequency domain.

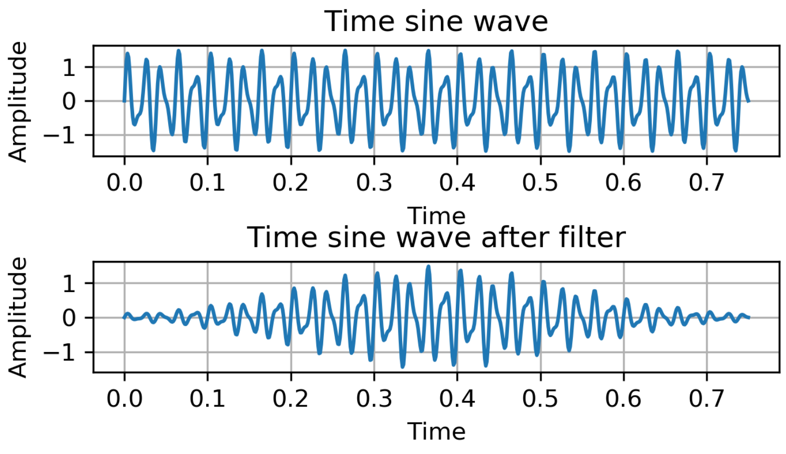


Figure 19 – Hanning window example. a) The time signal sum of two sines. b) The time signal after filter with the Hanning window.

3.7 Cross correlation

It is a signal processing method to observe the similarity in two signals delayed on time, in other words, it is used to find where two signals match. It is represented by the equation

$$(f \star g) \triangleq \int_{-\infty}^{\infty} \overline{f(t-\tau)} g(t) dt, \tag{3.2}$$

where $\overline{f(t)}$ is the complex conjugate of $f(t)$, τ is the delay. In another words one of the signals is fixed while the other signal is shifted in time. In each interaction the difference is calculated between the signals points, the interaction with less difference has the best match. In the image 20 (source: [21]) shows two signals shifted in time (first plot), the cross-correlation is plotted second and indicate the best match in time. The last plot shows the two signals shifted.

The Cross-correlation was the first method used as an automatic test on this project. Cross-correlation with RMSE was used to match two signals, one from a calibrated sensor and another from a sensor being tested. The results were not as expected and this method was suspended. The following chapter will explain it. But it is important to demonstrate why this method did not work.

3.8 Piezoelectric effect and Piezoelectric sensor

The Piezoelectric effect is recognized as the capability of a material to generate an electric charge when submitted to physical stress. "The piezoelectric effect is very useful within many applications that involve the production and detection of sound, generation of high voltages, electronic frequency generation, micro-balances, and ultra-fine focusing of optical assemblies" [22]. This means it is possible to use a Piezoelectric as a generator or as a sensor. The piezoelectric sensor is explained at the book [5]: "When struck by an impulse, a piezoelectric element will vibrate like a bell, "ringing" at its resonant frequency. In fact, there may be many resonant frequencies all excited together. If shaken by a vibratory motion, a piezoelectric element will produce a corresponding oscillating voltage at the same frequency as the motion. The element has a linear response; if the input motion is doubled, the output voltage will also double".

A piezoelectric sensor is the most common sensor to measure Acoustic Emission Events. The Indtact GmbH uses a piezoelectric inside the smartPredict 2.0 device. Also, a piezoelectric effect generator was used during the development to simulate the AE crack.

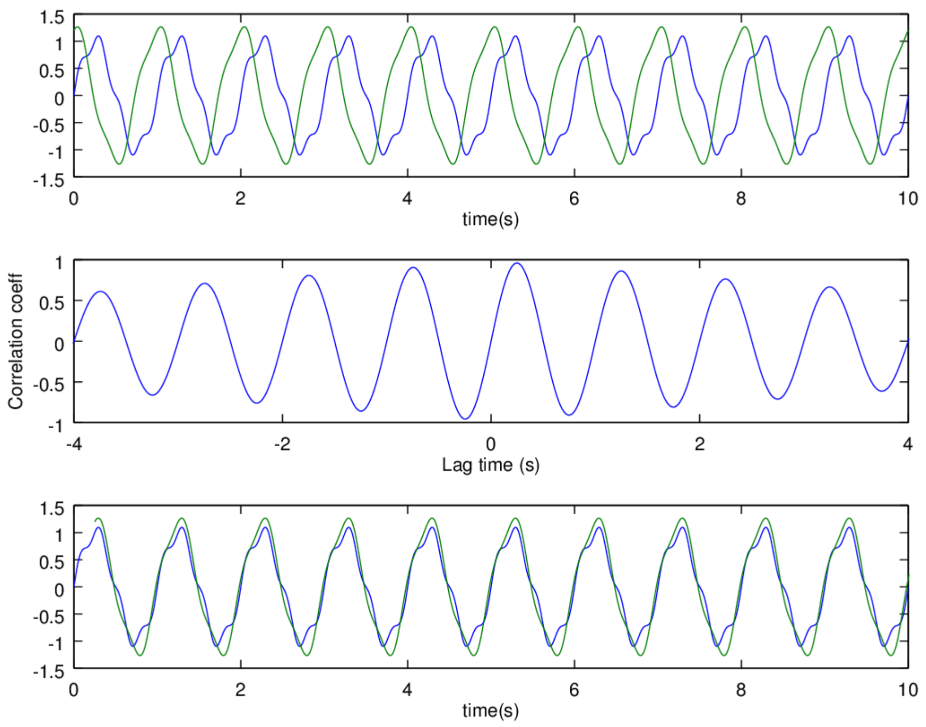


Figure 20 – Cross-correlation example. The image explains the process of the Cross-correlation technique. a) The signals shifted in time. b) The plot of the cross-correlation factor between the iterations. c) The signal after use the cross-correlation.

4 Proposed solution and Methodology

This chapter is focused in show the solution or what methodology was used to reach the requirements of the system. The chapter is the continuation of the section 2.3 (Problem description). The section 4.1 introduce the solution. Section 4.2 explain about all the automatic tests proposed, the section also discuss about the constraints of each automatic test. The section about automatic tests is divided in: cross correlation test, analysis of the global peak of the FFT test, analysis of the local peaks of the FFT test and analysis of the FFT integral. The section 4.3 clarify about the computational aspects, explaining in details why each tool is used on the supervisory.

4.1 Solution overview

Based on the requirements provided by the company and several discussions the detailed project was designed. The supervisory system will be implemented in the cross-platform tool QT/C++. The connections with other systems will be made using Python, the reason of this choice is because the Indtact company mostly use Labview with their tools, Labview has a connection with Python [1], so the same modules can be used on the future in other projects. The database is under development by another person and will be written on Microsoft Access. The Analog Discovery 2 is used in many projects inside the company. Html, Json and text file will also be used.

The test platform (Figure 21) is developed with different components, each component has a purpose on the platform. The computer will guide the user on the tests and also will validate and process all the raw data recorded. The Analog Discovery 2 board will be connected with the Discovery BNC in which will be responsible to record the data and generate the wave in a specific frequency. The amplifier, developed by Indtact (figure 7), will filter the non-desirable frequencies and amplify the area of interest. The sensor will get the signal generated from the wave generator [2].

As described in *Introduction*, to check if a sensor is working or no, automatic tests will be performed. The automatic tests will be basically verifying if the same frequency that it was simulated is being measured. To perform this task a signal must be simulated on a material, this signal generated will represent a crack on material (AE event), this is the Piezoelectric generator in Figure 21. The sensor that is being observed is on the same surface, the idea is that the sensor should measure the same frequency that the

[1] as previously explained this connection between Labview and Python is not simple, it was a challenge on the project.

[2] Piezoelectric generator, see section 3.8.

Pizeoeletric generator is providing to the material. To check if it is the same frequency the raw data will be transformed into an FFT, where several tests will indicate if that signal match with the Pizoeletric generator. FFT parameters as: global peak, local peaks and integral will be used.

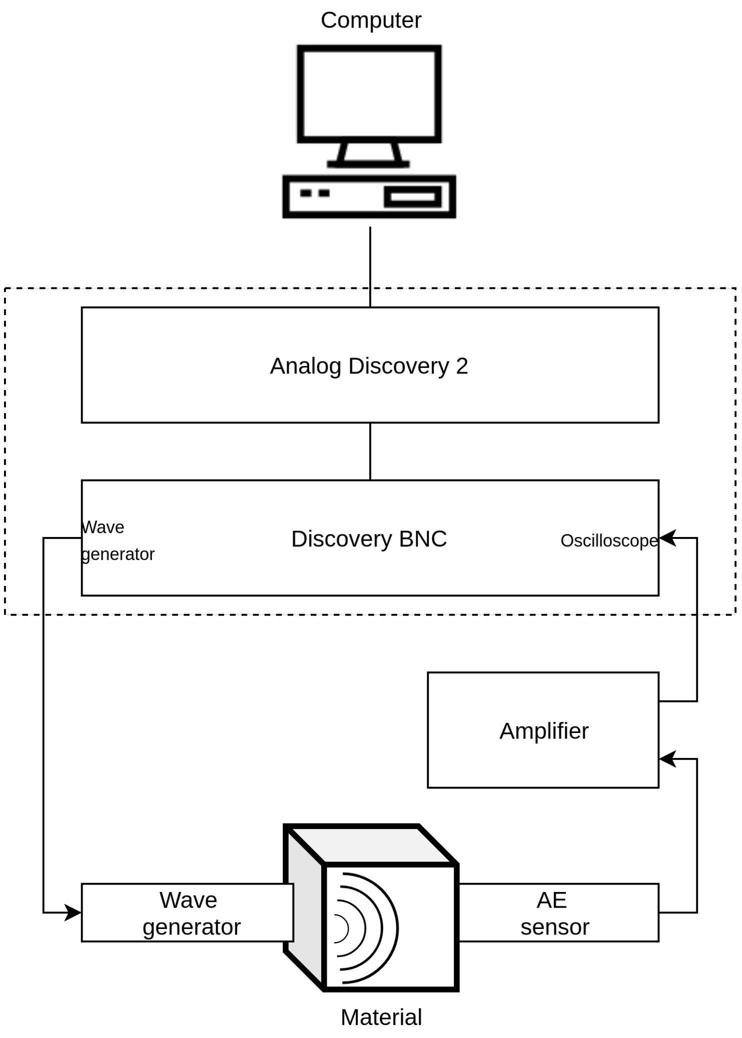


Figure 21 – Hardware platform structure. A sensor is attached to the same material of a wave generator. Analog Discovery 2 (with the BNC board) generate and measure the signal. The raw data is processed by the amplifier and send to the computer, which transforms the data to the frequency domain using the FFT transformation. The computer analyses the FFT. The computer will also guide the user on the tests and save all the data.

4.2 Automatic tests

The automatic tests are important to the system. The area of Acoustic Emission sensor does not have many literature describing techniques to test if a sensor is good or not good. The best way is verifying if the sensor is working to the tasks that should be working. It is necessary to simulate the environment in which the sensor will be, on this case, measuring acoustic emission events. As show the figure 3 a crack is propagated trough a material and this is an acoustic emission event.

The standard way to simulate a crack, on the Acoustic Emission area, are the techniques: fracture of pencil leads and the drop of a ball. These techniques are interesting since they generate a signal with a big range of frequency to analyze. The problem is that is not robust, because it is not possible to generate the same signal two times, so it is useless to this project. It is interesting to find a way to generate a crack in which the signal can be the same always. The proposed way is using a wave generator. The wave generator is attached to a material and the it provides a sound on a recognized frequency.

The simulated crack must be measured by the sensor with a similar shape. For example, if a simulated crack is generated with the frequency 800 Hz then the sensor must read a peak at 800 Hz and small frequencies around, but the global peak at 800 Hz. The figure 22 illustrates this. Different ways were think to make this automatic tests understand if the FFT match with the shape of the generated signal and will be shown on the next subsections.

The characteristic that allows measuring the signal with the same frequency is because the frequency does not change on a material, only the amplitude. So the signal that the sensor reads must be with the same frequency, but different amplitude. If the simulated crack is close enough to the Acoustic Emission sensor, this amplitude difference is attenuate.

To generate the wave and to measure the signal the Analog Discovery 2 board was used, as shown the Figure 21. Since the range of the sensor is between 0.06 Hz to 48 Khz all the frequencies outside this range are not important. To remove the high frequencies a Low-pass Butterworth filter of order 10 is used with the cut frequency 48 Khz (figure 23).

The tests are written in separated modules, external to the QT supervisory system. This allows easily to insert new automatic tests without change the QT codes. The supervisory will always check if the test indicates an error or no by the output of the module. The output must contain:

$$\begin{array}{lcl} iNDtact\_test\_return\_true\_test\_1 & & \\ \qquad\qquad AND & & \\ iNDtact\_test\_return\_true\_test\_2 & OR & iNDtact\_test\_return\_true\_for\_all\_tests \\ \qquad\qquad AND & & \\ iNDtact\_test\_return\_true\_test\_3 & & \end{array} \tag{4.1}$$

then the supervisory will treat that test as approved.

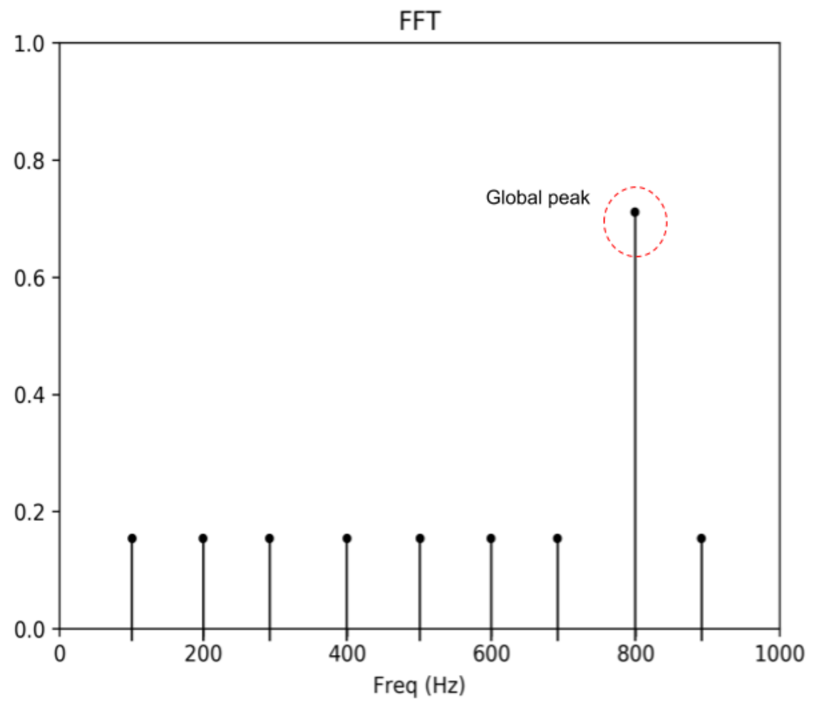


Figure 22 – Example of a good FFT with the global peak at 800Hz. There is only one visible peak and other frequencies appear with small amplitude. These small-amplitude frequencies are associated with harmonics and the material properties where the signal flow.

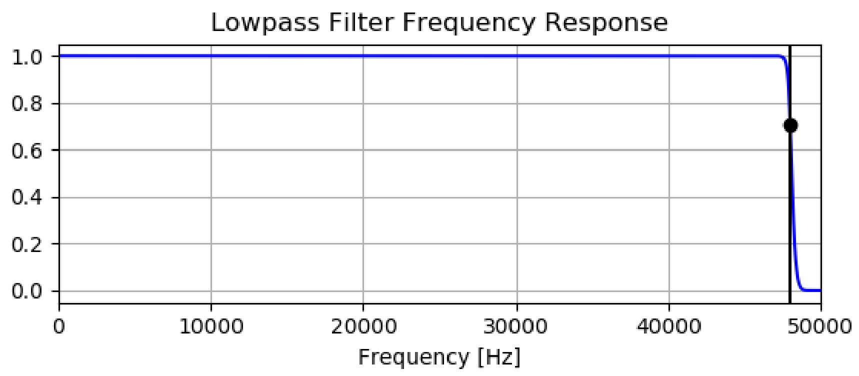


Figure 23 – Low-pass Butterworth filter, order = 10, cutoff = 48 Khz. This is used to filter the raw data from the Analog Discovery 2. The 48 Khz was used because this is the maximum range of the smartPredict 2.0.

4.2.1 Analysis with Cross Correlation and RMSE

The first method used on this project to analyze if the sensor is working, it was comparing two signals in time, point by point using cross-correlation [23].

- Signal 1 - from a calibrated sensor;
- Signal 2 - from the sensor that the tests are being performed;

These different sensors will be at a small distance (< 1 cm) from each other and when the test starts an impulse is produced between the sensors. The signals are measured using the two sensors, the raw signal measured is sent to the computer where the raw data will be transformed, after the cross-correlation is used the following signals are generated:

$$Y_1 = \begin{pmatrix} \hat{y_1} \\ \hat{y_2} \\ \vdots \\ \hat{y}_{n-1} \\ \hat{y}_n \end{pmatrix} Y_2 = \begin{pmatrix} \tilde{y_1} \\ \tilde{y_2} \\ \vdots \\ \tilde{y}_{n-1} \\ \tilde{y}_n \end{pmatrix}, \tag{4.2}$$

Y_1 and Y_2 have the same x axes as

$$x = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ n-1 \\ n \end{pmatrix} \omega, \tag{4.3}$$

ω is just a scale. Then the a sum

$$R = \sum_{i=0}^{n} \frac{\sqrt{(\tilde{y}_n - \hat{y}_n)^2}}{n} \tag{4.4}$$

is made. If the R is bigger than a threshold then the sensor is not good, otherwise, the sensor is good. This is illustrated in the following steps:

1. The signals are changed using the cross correlation and any delay between the signals is eliminate;
2. The signals are compared point by point using the RMSE value;
3. If the output of the RMSE value is higher than a threshold that sensor probably is not good;

However, the first tests with this approach were performed using two well-calibrated sensors and indicate that this approach is useless. An Acoustic Emission sensor produces a signal that is inherent to many different variables and each sensor has a time signal with different shape. The test on the figures 25, 26, 27 and 28 are receiving the same signals and it is visible that the signals are not even close to each other. So, this automatic test was removed from the project, because it is useless.

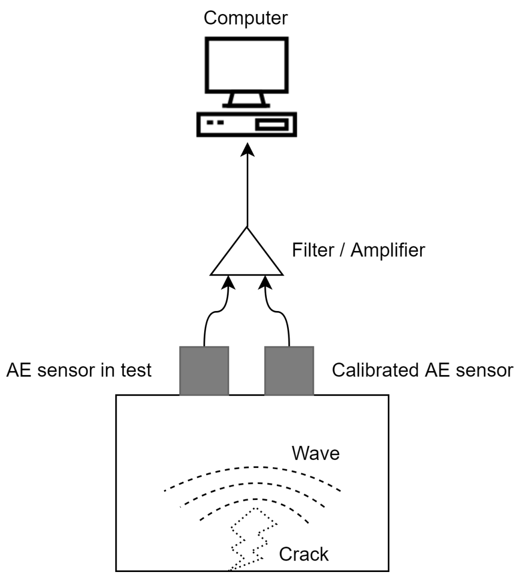


Figure 24 – Cross-correlation test physical platform. The two sensors are attached close to each other. The signals are measured and send to comparison.

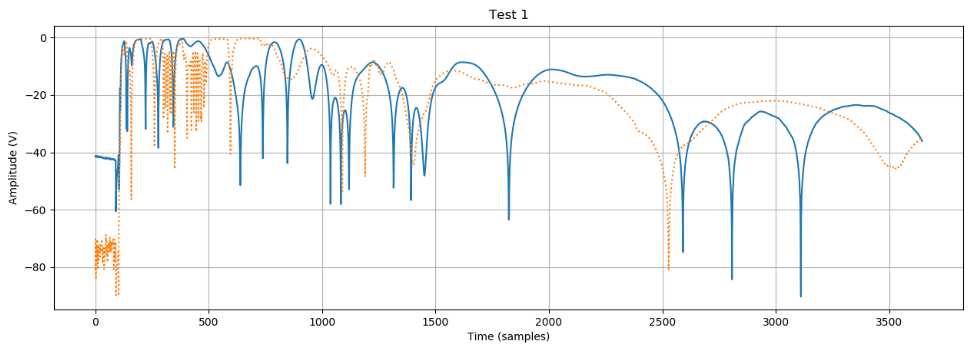


Figure 25 – Cross correlation: test 1

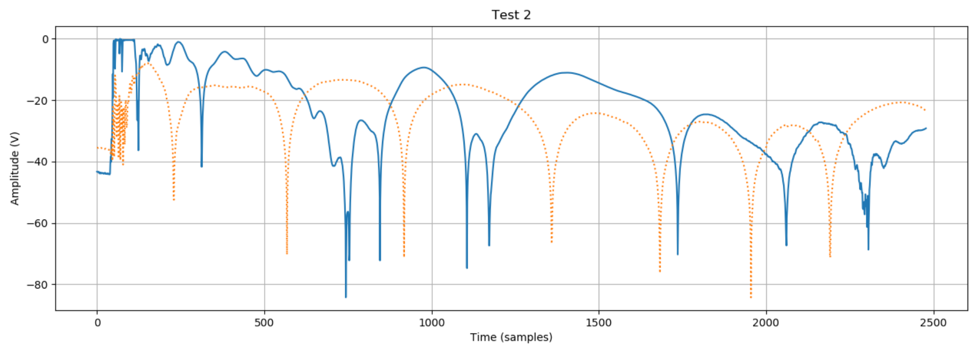


Figure 26 – Cross correlation: test 2

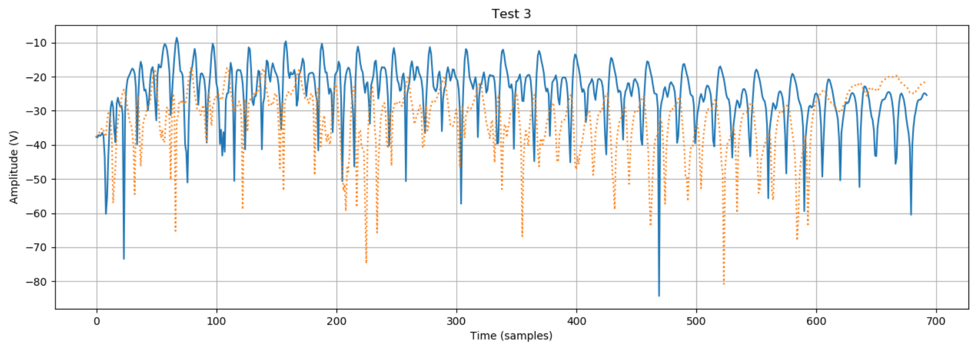


Figure 27 – Cross correlation: test 3

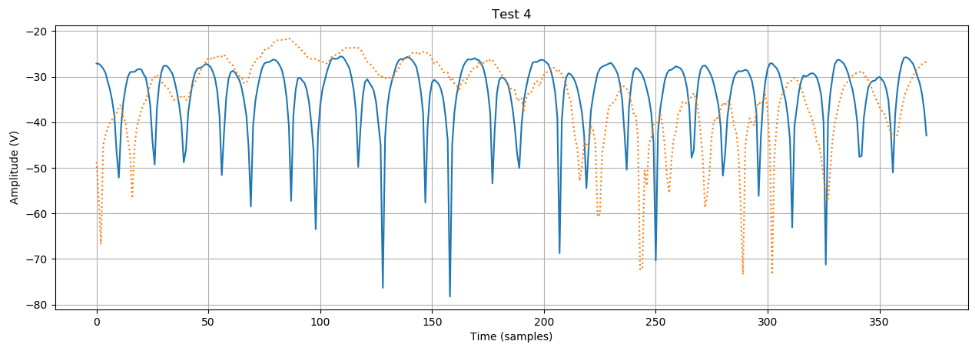


Figure 28 – Cross correlation: test 4

4.2.2 FFT global peak analysis

The computer will manipulate the data in order to find the informations inside this data. First, the signal will be processed with a High-pass Butterworth filter or a Signal detrend filter intending to eliminate weird low frequencies (DC effect). Then with the data, an FFT is generated as

$$FFT_y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix}, \tag{4.5}$$

where the variable n is equal to the number $dim(FFT_y)$. The FFT_y is normalized to be more simple to analyse, as

$$FFT_{y_{normal}} = \frac{FFT_y - \mu}{\sigma}; \mu = \frac{\sum_{i=0}^{N-1}}{N}; \sigma = \sqrt{\frac{\sum_{i=0}^{N-1}(FFT_{y_i} - \mu)}{N}}, \tag{4.6}$$

then the FFT_x will be defined as

$$FFT_x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ N-1 \\ N \end{pmatrix} \omega, \tag{4.7}$$

and ω is equal to

$$t = \frac{f_s}{N}, \tag{4.8}$$

where f_s is selected by the user and is equal to the number of samples that the AD2 board will get per time unit (described as sample rate in chapter 3). On AD2 f_s is defined in Hz, then get the maximum value of $FFT_{y_{normal}}$:

$$MAX_{1 \leq i \leq N}(FFT_{y_{normal}}) = f_i. \tag{4.9}$$

The maximum value relative to some frequency is selected as

$$FFT_{x_i} = f_x, \tag{4.10}$$

with the FFT_{x_i} founded it is possible to make the comparison

$$|f_x - f_{test}| \leq \Delta_{tolerance}. \tag{4.11}$$

If the comparison return yes then the test is approved, if no then the test is not approved. A simple way to understand this is seeing the figure 29. The FFT generated by the data measured must have a global peak at 800 Hz, or at least, close to 800 Hz, on the Δ area. The 800 Hz is just an example because several frequencies will be tested.

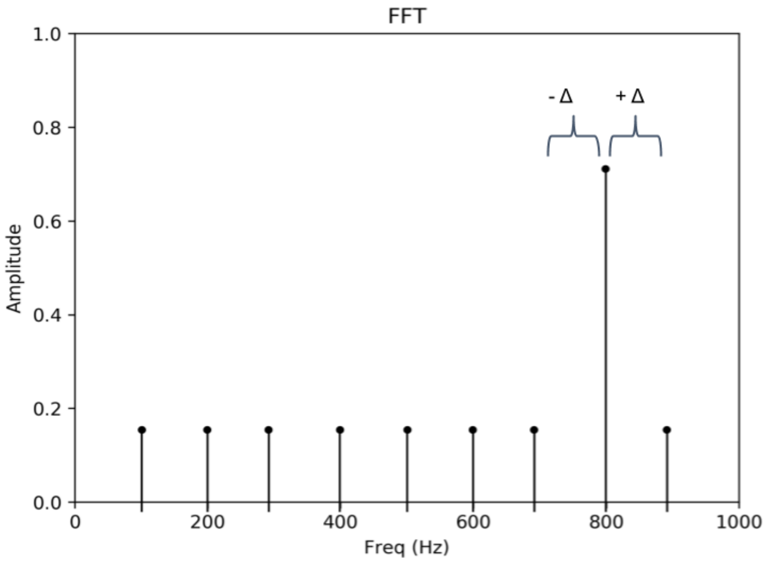


Figure 29 – FFT global peak test example. The global peak must be inside the tolerance area. This example was performing a test at the frequency 800Hz. Several other frequencies will be used to validate all the range of the sensor.

4.2.3 FFT local peak analysis

The test FFT peak analysis is intuitive, but not enough robust. The FFT peak analyses only observe the global peak and not all the rest of the FFT, important informations may be lost. There is the possibility to happen local peaks and if happen it is an error indicator [3], because this frequency was not simulated on the crack. So, a test must treat this error indicator.

If the peak is ignored (in other words, remove the peak from the first FFT used on the *global peak analysis*) generating a histogram with the distribution around this $FFT_{without\_peak}$ on a sensor working and well-calibrated and a closed platform the results will be something close to the figure 31. In this example, the histogram had a maximum distribution at 0.27 (FFT_{max}). With this value it is possible to observe if any other value (with the exception of the global peak) is above this value so if $FFT_{without\_peak}(i)_{i=any\_value} > FFT_{max}$ there is an error.

[3] Local peak does not mean harmonics generated by the major frequency in contact with the material. Harmonics appear naturally and usually do not has a big amplitude in comparison with the global peak.

The histogram was built using a python script, with the *matplotlib* Library, called *calibrate_ auto_ test_ histogram.py* [4]. This script can not be called by the QT supervisory, but the FFT_{max} results can be changed at the Supervisory settings or directly on the *Json database file*.

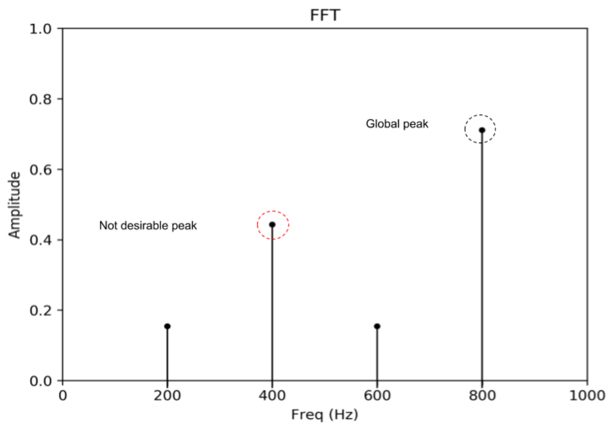
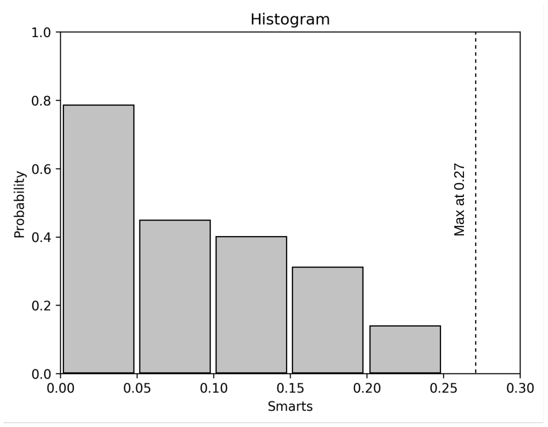


Figure 30 – FFT error example test local peak. There is a global peak at 800Hz, but a local peak appears, this indicates an error because this frequency was not generated by the wave generator output.



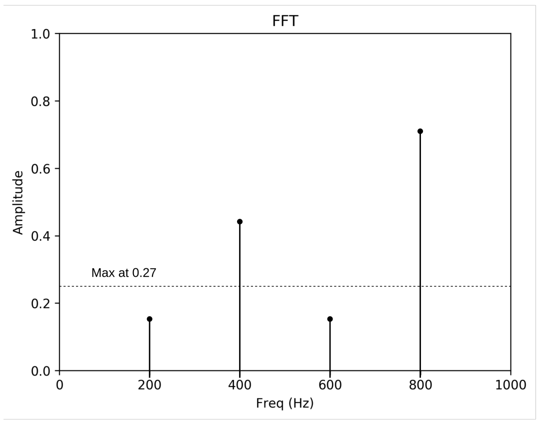


Figure 31 – Histogram around x-axes and $FFT_{without\_peak}$. From the histogram, it is possible to get the maximum tolerated local peak. After this maximum point indicates an error.

Figure 32 – Results after the FFT local peaks test. The line is the maximum tolerated local peak from the histogram, all local peaks above this line indicates an error.

4.2.4 FFT integral analysis

Thousands of tests can be created in order to find a malfunctioning sensor. The tests must be enough robust to identify weird behaviors on the sensor. The *FFT global*

[4] It can be found inside the project, at the *root / iNDTact_ testSensorApp / Resources / Python* folder

peak test and *FFT local peak test* identify some interesting aspects, but other variables can be observed to the automatic tests be more robust. The integral of the FFT. It is defined as:

$$\beta = \sum_{i=0}^{len(FFT)} (FFT_y)_i, \tag{4.12}$$

where $len(FFT)$ is the number of points of the FFT. The integral can be important to identify weird behaviors, as show the figure 34. These weird behaviours can be an indicator of an error. The natural assumption is that small harmonics should appear, but a non-desirable shape is an error. To measure the maximum integral tolerance several tests must be made and a histogram must be created. There is one important requirement, to perform the integral the FFT global peak must be removed, as the *FFT local peak test.* An example of a histogram can be observed in Figure 33. So this integral analysis increases a little more how robust the tests are.

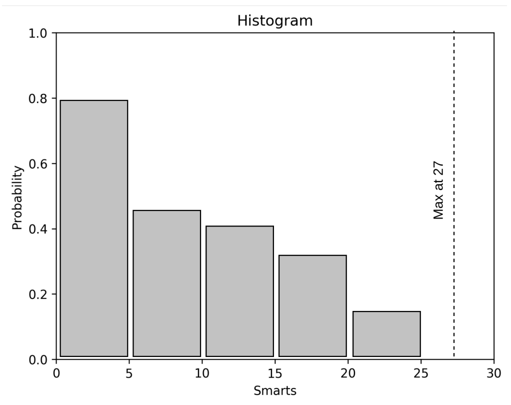


Figure 33 – Histogram around β. The histogram provides the maximum tolerated integral of the FFT.

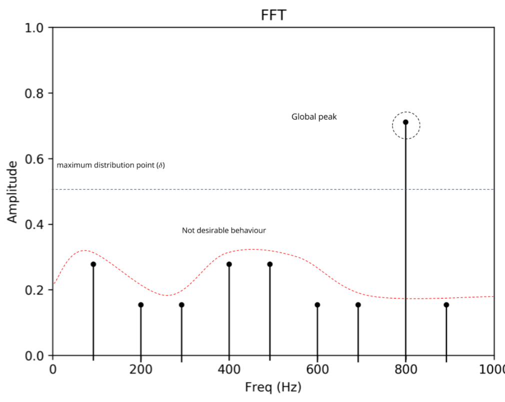


Figure 34 – Explanation about the FFT integral test. The weird shape around the small amplitude frequencies indicates an error.

4.3 Computational aspects

The figure 35 shows the project structure but with the focus at the QT/C++ supervisory system, the supervisory is the center of the project, where all the informations will go trough. Different languages/tools were used to develop this supervisory and each one has a purpose. It is a supervisory with a kind of complexity, since used: Python, QT/C++, Labview, Json, Html, Microsoft Access database and many other libraries. The computational solution was developed trying to match the company background and the best tools for the solution.

QT/C++ is a tool that is increasing each year and fits perfectly with this system. QT/C++ provides several tools that allow fast development and match with a short time to develop. QT/C++ is becoming more popular every day and iNDTact GmbH is growing fast, so it is natural that in the future they will cross the same way, this first QT/C++ can be a first step to iNDTact use QT. Another advantage is that QT is free for open source applications or internal projects, allowing the company to save money. Another big advantage of QT that was not used on this project is the cross-platform solutions.

Python is an important computer language in the software world, "Nearly 12%" of the programmers use it on the operating systems like UNIX, Linux, Windows and Mac OS [24]. It is extremely simple to use and really powerful. Python provides many advantages and libraries that allow fast development. Basically, the automatic tests were written in Python and also the connection with the database. iNDTact Gmbh has some Python experts that can edit the script in the future. Since iNDTact does not have any other employee that works with QT or C++ it is important that parts of the supervisory (the parts that can have some changes in the future) are written in python. This external QT/C++ modules written in python can also be used in several other projects inside the company. Also, because the QT has his own Debug system using *qmake* and python can be changed without debugging the QT again, this saves several minutes on the project. Each Python module is called as an external script and can communicate with the supervisory, sharing data.

Labview was used as an alternative to the python automatic tests, so there are two options, perform the automatic tests in Python or Labview [5]. The main reason is that iNDTact has a major background in Labview and a project written in Labview has more chances to have a long-life on the company. One of the biggest challenges of the project was to connect QT and Labview. Labview is not exactly a computer language and does not provide an official way to call by the command line. To perform these tasks were necessary to create an external server that manipulates Labview, it is possible to call and receive data from Labview (Appendices: D.2).

Json is a way to share data between modules or computer languages, for small purposes can even be used as a database. Json is used many times on the system for different purposes. Since there are many modules and different computer languages sharing data on this supervisory it is important to provide a way that can share data on a similar language, for example, the messages to write on the database they are written on a Json file, then the same Json file is read by the Python database module in which send this data to the Access database. The settings database is used to save the options data that is used on the supervisory. The automatic tests and Manual tests Json file are an important

[5] Some of the functionalities provided by the Analog Discovery 2 libraries can only be used in Python. For example, the number of samples, there is no limit in Python, but has in Labview. Also, it was noticed that the board does not turn off properly using the Labview library

advantage of this supervisory system. Since the Automatic tests or Manual tests can change at any time it is important to provide an easy way to rewrite these tests. If a new test needs to be implemented then it is just insert some lines on this Json files and will appear on the supervisory. The Automatic and Manual Json save all the information about the tests and are read also by the Python and Labview modules (Appendices: D.5).

The Access was used because is already implemented on the company, a Python module makes the connection using some SQL tools (Appendices: D.4). The connection with the Access database is intermediate with an Excel file, the Excel file sends all the data to the Access database automatically. The HTML template was written apart from the QT/C++ because can be changed at any time by the company. Using HTML make it simple to write a template. This template is used to generate the PDF report file (Appendices: D.6).

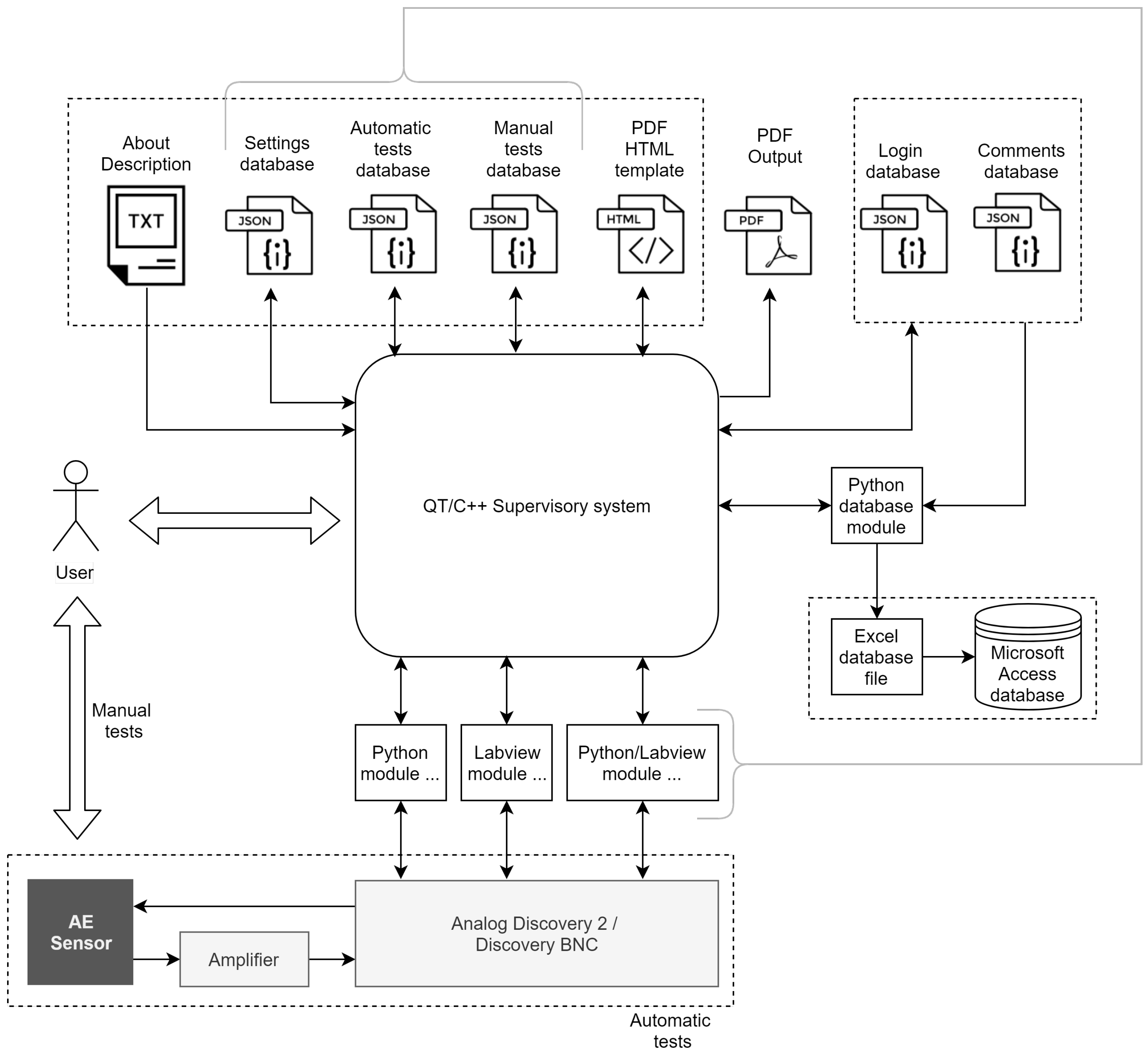


Figure 35 – QT/C++ supervisory system structure. The center of the supervisory is the QT/C++ part, which connects all the other tools. The Json modules share data between the modules or can be used as a small database. Python modules perform automatic tests and communicate with the database. Labview also makes the same automatic tests. Other tools, like HTML, are also used to easily change the template of the PDF report. The user interaction with the supervisory and the sensor being tested.

The design has many functionalities: improve the user experience, improve the ergonomic, help the user to find things easily, etc. "A screen's layout and appearance and a system's navigation affect a person in a variety of ways. If they are confusing and inefficient, people will have greater difficulty doing their jobs and will make more mistakes.

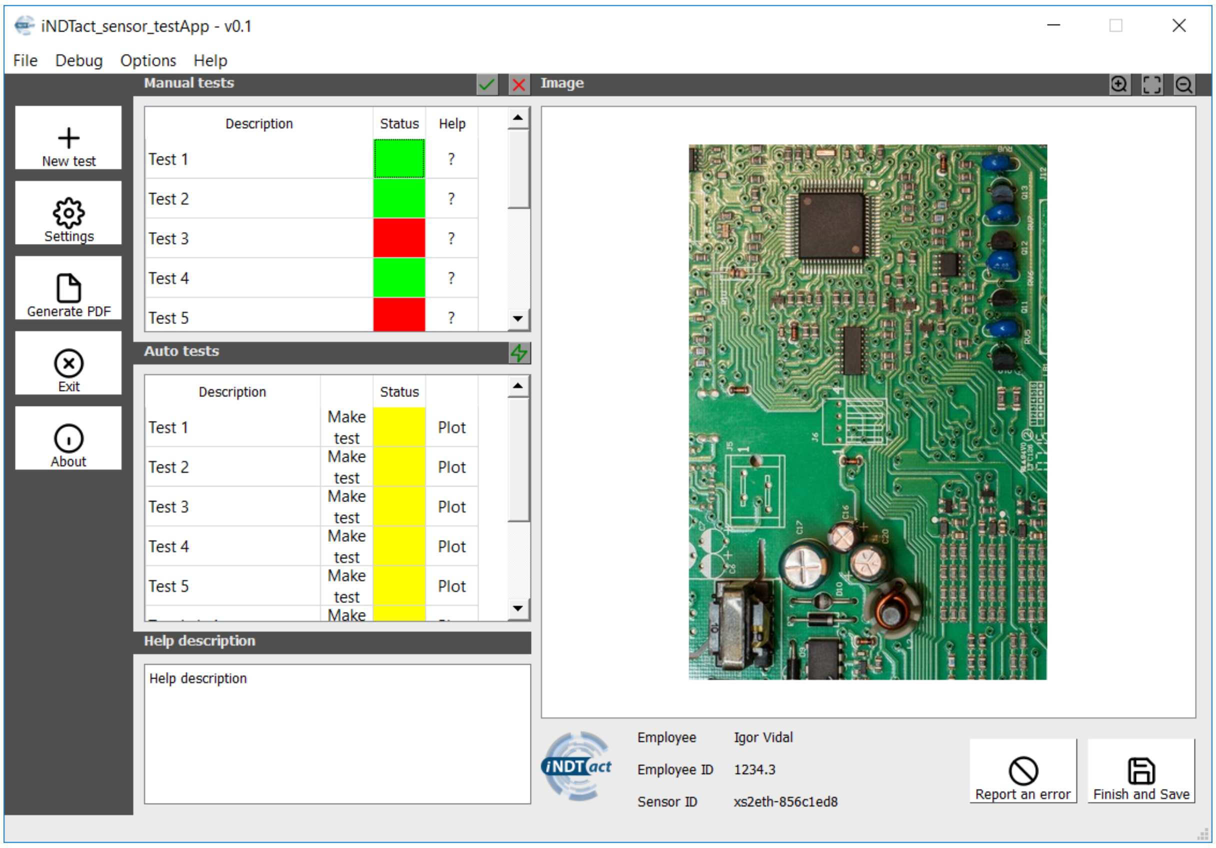
Poor design may even chase some people away from a system permanently. It can also lead to aggravation, frustration and increased stress. [...] Poor design can also have a huge financial cost to users and organizations. A critical system, such as one used in air traffic control or in a nuclear power plant, may compromise the safety of its users and/or the general public" [25]. The supervisory of this project was implemented with some criteria: less color as possible, ignore animations in order to use less computer memory and minimum information. QT creator allows to use many tools as labels and frames to change the design and also the *styleSheet...* tool. The figure 36 show the result. [6]

The structure of the Main Window of the supervisory was built thinking about the aspects above and also the requirements of the company. The two tables *Manual tests* and *Auto table* are generated with the informations inside the JSON *Automatic tests database* and JSON *Manual tests database.* The help description shows additional information about the test (the information will appear after the user double click the button *?* of that test), for example, the current that a wire should consume. The buttons guide the user for some functionality. The small squares above the *Manual tests* and *Auto table* help to make the tests more quickly, next to the *Auto table* the button start automatically all the automatic tests. To set a *Manual tests* as approved (green) it is just necessary a double click at the "status" column of each test, another double click will change to not approved (red). The same system of color is used at the *Auto table*, but the color will automatically be selected by the test (the user can not change on this case, just if repeat the test). The main figure area is also to help the user, in which will appear the PCB or the Plot of the automatic test. [7]

The figure 37 is the Login window, the Login window will get all the data about the tests from the user, this informations will be saved to the Json database file. Then, if all the informations are filled, the main window will be open. The table in the login window is updated as the tables in the main window. A new line can be inserted, it is necessary just modify with few lines the json database file.

[6] The icons are from an external open-source library call *Feather* [26]

[7] The electronic board that appears on the Figure 36 it is just an example, the final version must appear the real smartPredict 2.0 board, but since this document is public it is interesting to maintain secret this part.



Manual tests

Description	Status	Help
Test 1	🟩	?
Test 2	🟩	?
Test 3	🟥	?
Test 4	🟩	?
Test 5	🟥	?

Auto tests

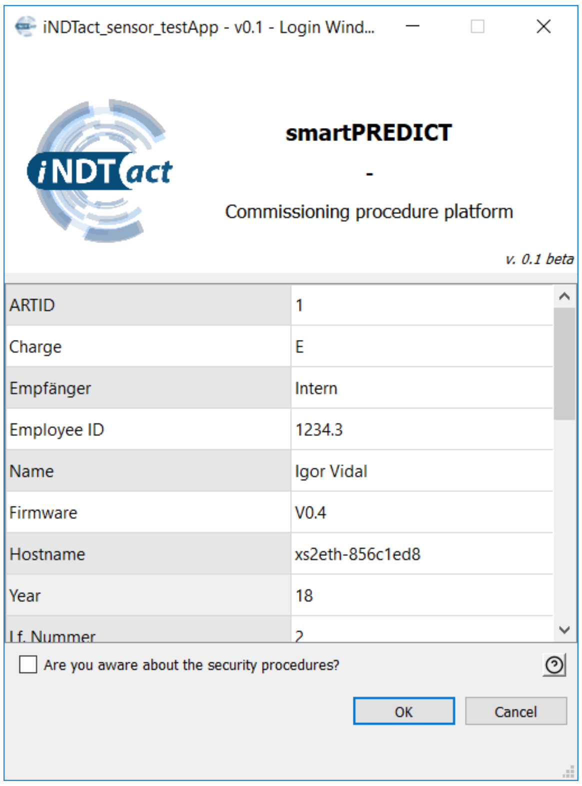
Description		Status	
Test 1	Make test	🟨	Plot
Test 2	Make test	🟨	Plot
Test 3	Make test	🟨	Plot
Test 4	Make test	🟨	Plot
Test 5	Make test	🟨	Plot
	Make	🟨	

Help description

Help description

Employee	Igor Vidal
Employee ID	1234.3
Sensor ID	xs2eth-856c1ed8

Figure 36 – Supervisory QT/C++ Main Window. This is the area in which the user will spend most of the time. The manual tests and automatic tests are on the lists in the center. The columns fill with color indicates if the test is approved or no, the "red" color indicates an error, the "green" color indicates that it is approved, the "yellow" color means the automatic test was not started. The images appear on the right, where the PCB is showed. The buttons provide an easy way to flow between the many functionalities. Other small buttons facilitate the work of the user, providing a zoom button, make all the automatic tests in one click, etc. A stop button appears always that an automatic test starts in case of necessity to abort the automatic test.



smartPREDICT

-

Commissioning procedure platform

v. 0.1 beta

ARTID	1
Charge	E
Empfänger	Intern
Employee ID	1234.3
Name	Igor Vidal
Firmware	V0.4
Hostname	xs2eth-856c1ed8
Year	18
Lf. Nummer	2

☐ Are you aware about the security procedures?

Figure 37 – Login window supervisory. This is the initial window when open the supervisory. Here the user will fill with data (all this data will also go to the database). If some of the information is missing the next window will not open and an error message will appear. The button with *?* opens a PDF file with the security procedures to perform the tests.

5 Implementation and obtained results

The chapter 5 is focused in talks about the implementation and obtained results. The final structure of the project is complex system with many tools inside, this chapter will not discuss all the details behind the low level of the structure, however it will highlight some topics.

The chapter is divided in many sections or subsections. A section explaining the low level of the supervisory (section 5.1), it is divided in many different subsections discussing interesting topics as: the way the supervisory change the windows, how the supervisory call the automatic tests, the flow of some functions, etc. The section 5.2 explains the PDF output. The log file output is discussed in the section 5.3. The options windows is the section 5.4. The details about the Json structures that it was used on the system is at section 5.5, each Json file has a subsection explaining. The last section (5.6) is the analysis of the results. The appendices A and B support at the understanding of this chapter.

5.1 Low level computational aspects

5.1.1 QT supervisory message between windows

The QT supervisory has a lot of different screens (e.g.: option windows, main windows, login window, etc), called UI. Each screen has different purposes. The UI also declare a class, with many functions inside, this functions will be the low level of each screen. The UI's send information to each other, because some data from a UI can be important on another UI (e.g.: the main window needs the ID o the sensor from the login window). On this project all the UI's communicate with the *MainWindow UI*, this happened because that is where all the tests will be performed. The screens can communicate with each other using the QT function *connect*. The figure 38 shows how are the interactions behind the screens.

5.1.2 State machine between windows

The change of window is controlled by some conditions, some of them are select by the users and others are under restrictions. The way of change was implemented with the idea that the user sometimes can click on a button without this intention, so the program must avoid this. A state machine diagram (Figure 39) explain all the changes. Basically, always that a button to open a new window is clicked a Message window will appear to ask if the user really has the intention to change to another window.

5.1.3 State Machine enable/disable windows

Some buttons when opening a new window will disable the *Main Window UI*, and it will be enabled again after the window opened close. So the user only can manipulate one window per time, this was implemented to avoid undesirable states on the program. A state machine explain this behaviour (figure 40).

Also, another enable/disable event inside the supervisory is when the automatic test start. The automatic table is disabled always that an automatic test is being performed and it is enabled again when the test stop. This is to avoid weird behavior, for example, start two tests at the same time.

5.1.4 Automatic Test structure

5.1.4.1 Python

There is a class flow to make an automatic test. The flow will depend on the desired kind of test. The structure was built with different classes, as a way to be more clear and organized. Each class has a different purpose. First, on the top, the function *Test_N.py* is called, inside this function is the definition of the test, with the information of the frequency in which the test will be performed. Then the *setupTest.py* will contain the step by step of each class to be called. Some of the classes will need to communicate with a JSON file or with an external library. The figure 41 shows how each class is called.

5.1.4.2 Python and Labview

On this approach (Figure 42) each language (Labview and Python) will manipulate some part of the flow. The Labview part will be responsible to setup all the environment and get the data, then export this data to a *.txt* file on the *.csv* standard. The python part of the process (file *getLabviewDataFromFile.py)* will make all the rest (e.g.: organize the datas and generate FFT, analysis, etc). Important to explain that to call a Labview *.vi* it is necessary to use a python code first.

5.1.4.3 Labview

It is also possible to perform the test using basically Labview (Figure 43). All the tests, analyses and outputs will be built there. The *.vi* is organized in different *subvi's*. To call a Labview *.vi* is necessary first call a python module (*callLabviewByPython.py*), it is not possible to call directly a Labview from QT (Appendices: D.2). This same python module will get the output from the Labview and send it to QT. It is important to remember that in the automatic Json file the *language* key is writed as Labview, but internally a python script will be called, it is like this to be more simple to the user.

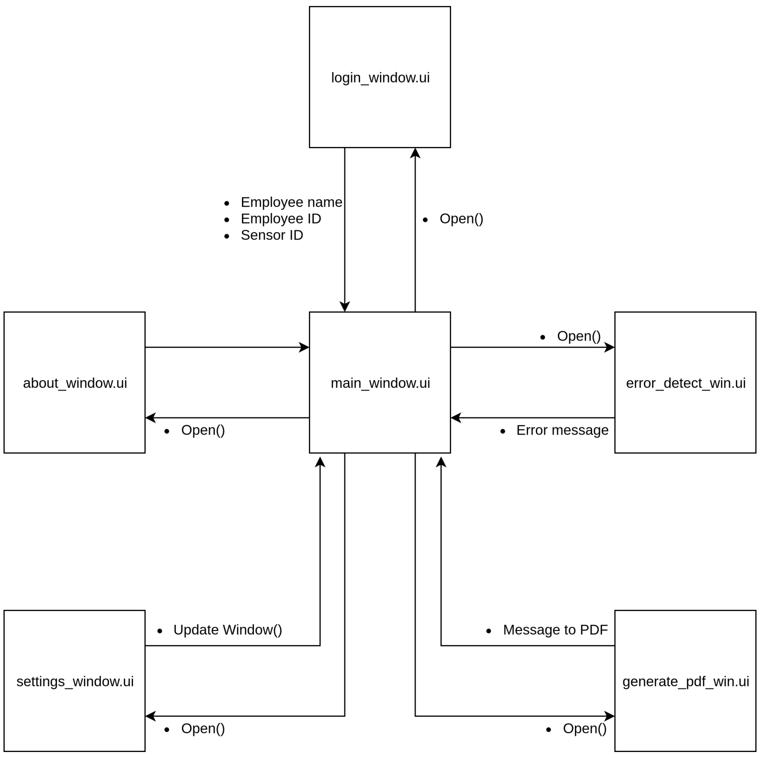


Figure 38 – UI order of the messages between the screens. All the windows communicate with the *main window UI*

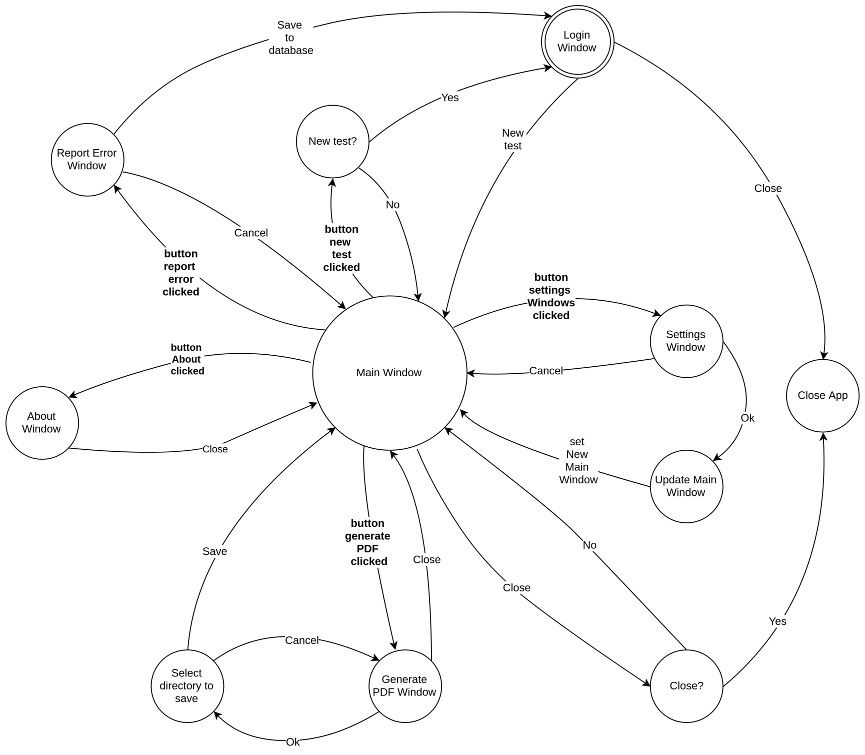


Figure 39 – State machine UI. Show the steps to change to another window, this avoids the user to reach an undesirable place on the supervisory.

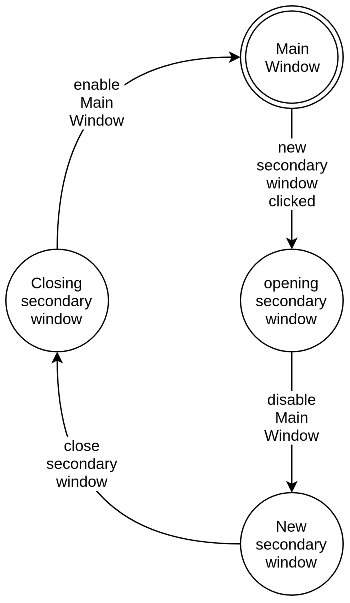


Figure 40 – State machine enable/disable UI. This means only one window can be used per time, the other window is blocked until the user closes the second window.

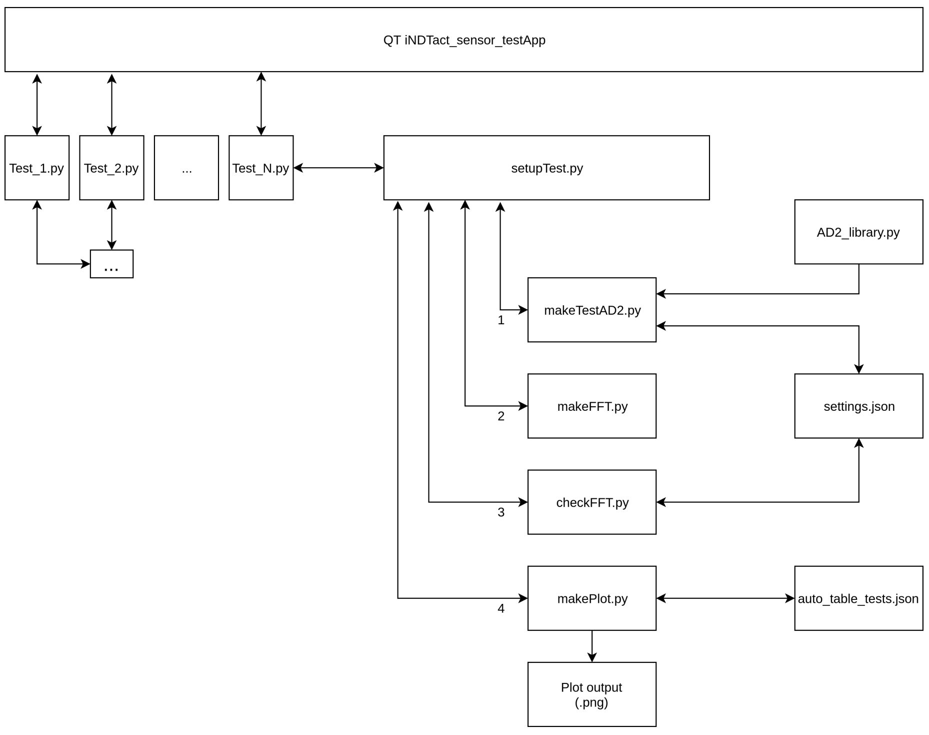


Figure 41 – Python test diagram.

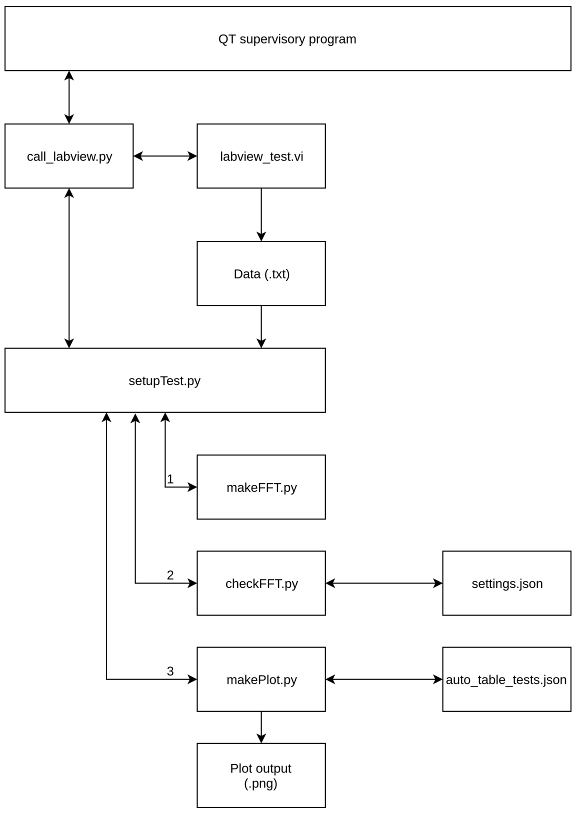


Figure 42 – Python and Labview test diagram.

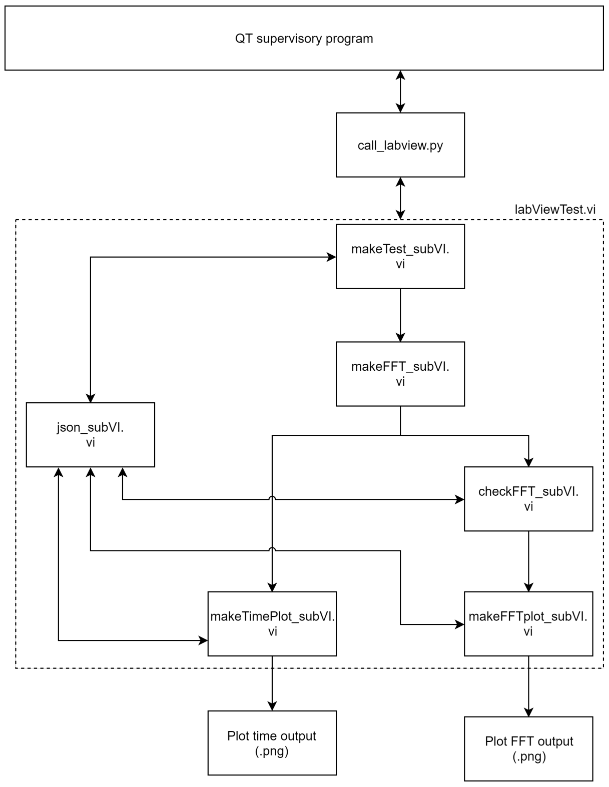


Figure 43 – Labview test diagram.

5.1.5 QT Functions Flow example

To make an organized code it is important to divide the script into small functions, so it is easier to remake when necessary. Many functions were written inside the QT/C++ supervisory and it is not the main subject of this document show all the low level details of the codes, but some examples can be interesting to support the understanding of the codes. The examples are in the subsections 5.1.5.1, 5.1.5.2 and 5.1.5.3.

5.1.5.1 Update screen

The figure 44 shows all the functions in which are called to update the main window. This function is called when the main window is opened and it is necessary to organize all the screen with informations provided by many places (e.g.: Json manual and automatic files, the Json login window, etc). It will call each function that organizes the *automatic table*, the *manual table* and others internal variables.

5.1.5.2 Automatic table clicked

Always that the user clicks on the auto table a range of functions is called (Figure 45). First, the script will identify if the column that was clicked is a plot or a *make test*. If it is a plot then a image respective to that line is output on the supervisory area. Now, if it is *make test* the supervisory setup all the environment to call the python process. A lambda function defined wait the process finish (Appendices: D.3). If more tests will be called (in case the user click the *make all the tests* button) then after the test finishes the setup function is called again.

5.1.5.3 Receive message to insert in database

Save the data in a database is one of the requirements of this project. There are two ways to call the python script that save the data on a database (Figure 46), one is clicking the button *Report an error* and the other way the button *Finish and save*. They will arrange the structure to send to the database and then record in a Json file, in which the Python database module will read.

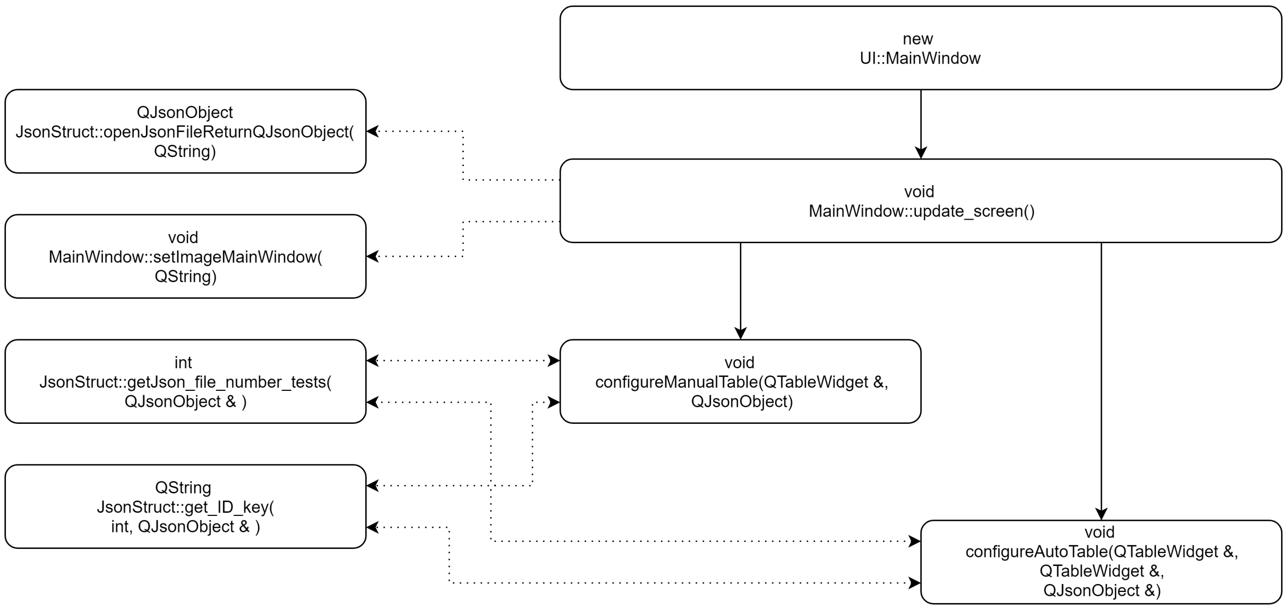

new UI::MainWindow
void MainWindow::update_screen()
QJsonObject JsonStruct::openJsonFileReturnQJsonObject(QString)
void MainWindow::setImageMainWindow(QString)
int JsonStruct::getJson_file_number_tests(QJsonObject &)
void configureManualTable(QTableWidget &, QJsonObject)
QString JsonStruct::get_ID_key(int, QJsonObject &)
void configureAutoTable(QTableWidget &, QTableWidget &, QJsonObject &)

Figure 44 – Update screen flow diagram.

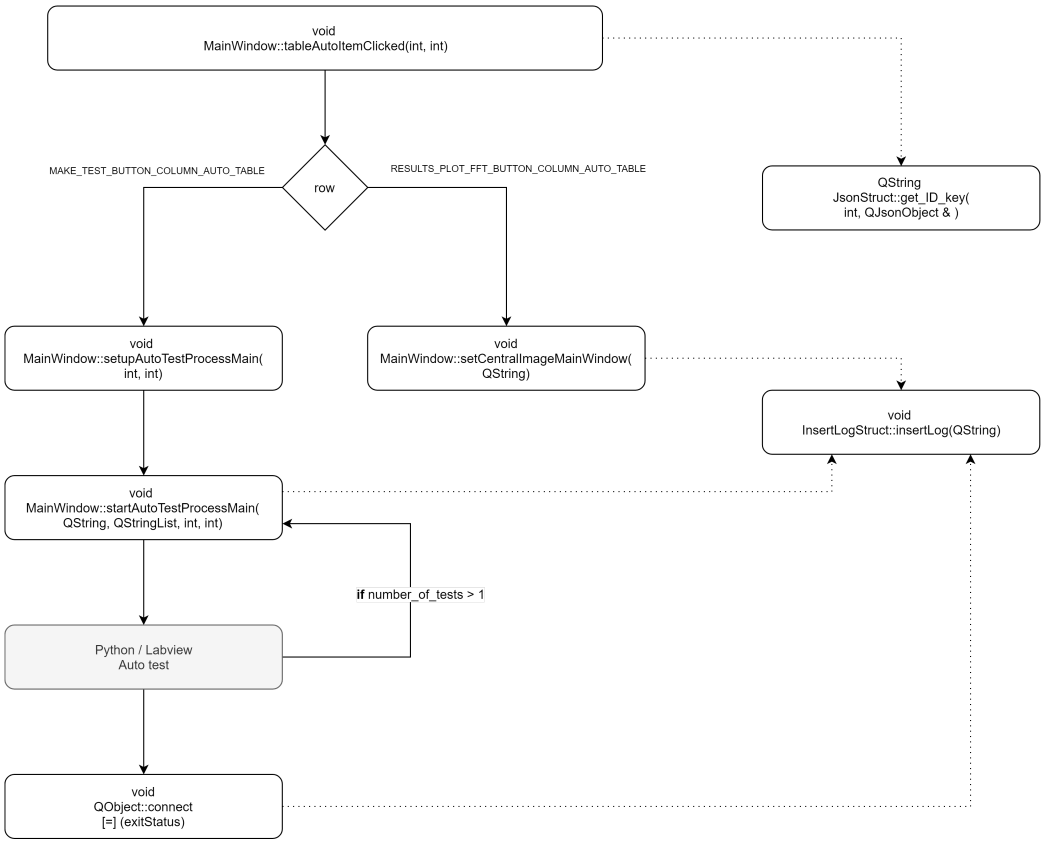

void MainWindow::tableAutoItemClicked(int, int)
MAKE_TEST_BUTTON_COLUMN_AUTO_TABLE
row
RESULTS_PLOT_FFT_BUTTON_COLUMN_AUTO_TABLE
QString JsonStruct::get_ID_key(int, QJsonObject &)
void MainWindow::setupAutoTestProcessMain(int, int)
void MainWindow::setCentralImageMainWindow(QString)
void InsertLogStruct::insertLog(QString)
void MainWindow::startAutoTestProcessMain(QString, QStringList, int, int)
if number_of_tests > 1
Python / Labview Auto test
void QObject::connect [=] (exitStatus)

Figure 45 – Table auto clicked flow diagram.

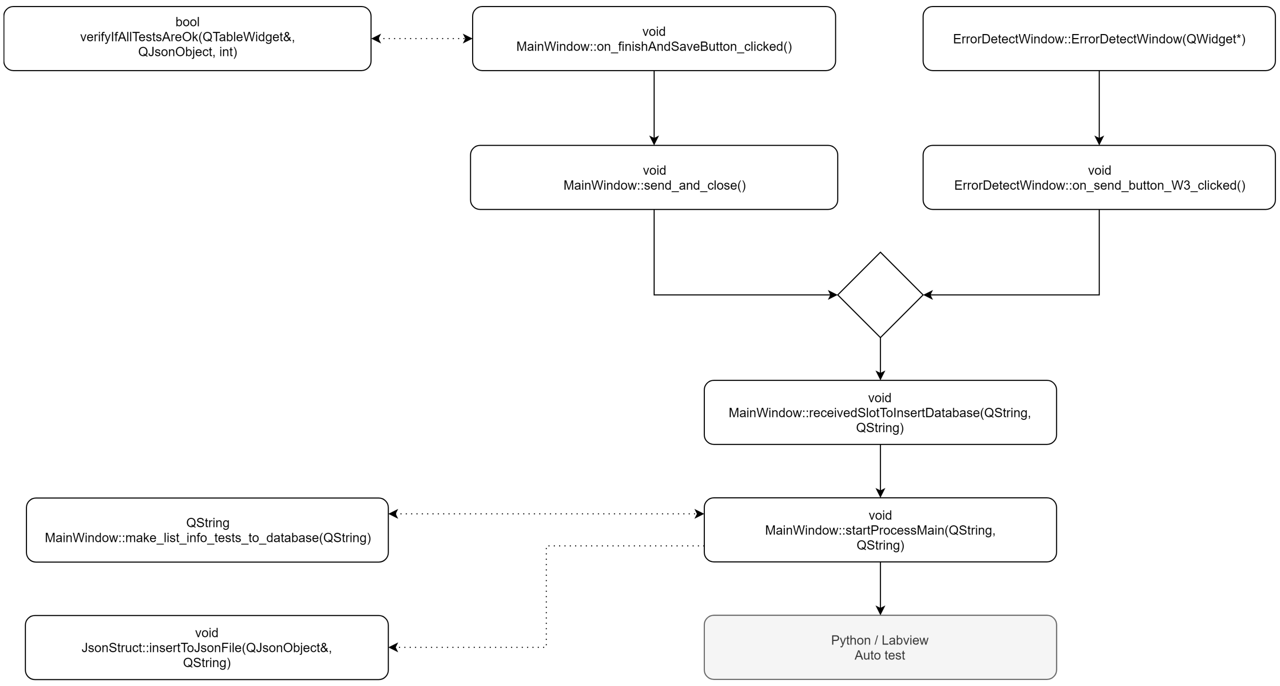

bool
verifyIfAllTestsAreOk(QTableWidget&,
QJsonObject, int)
void
MainWindow::on_finishAndSaveButton_clicked()
ErrorDetectWindow::ErrorDetectWindow(QWidget*)
void
MainWindow::send_and_close()
void
ErrorDetectWindow::on_send_button_W3_clicked()
void
MainWindow::receivedSlotToInsertDatabase(QString,
QString)
QString
MainWindow::make_list_info_tests_to_database(QString)
void
MainWindow::startProcessMain(QString,
QString)
void
JsonStruct::insertToJsonFile(QJsonObject&,
QString)
Python / Labview
Auto test

Figure 46 – Receive slot to insert database function flow diagram.

5.2 PDF report output

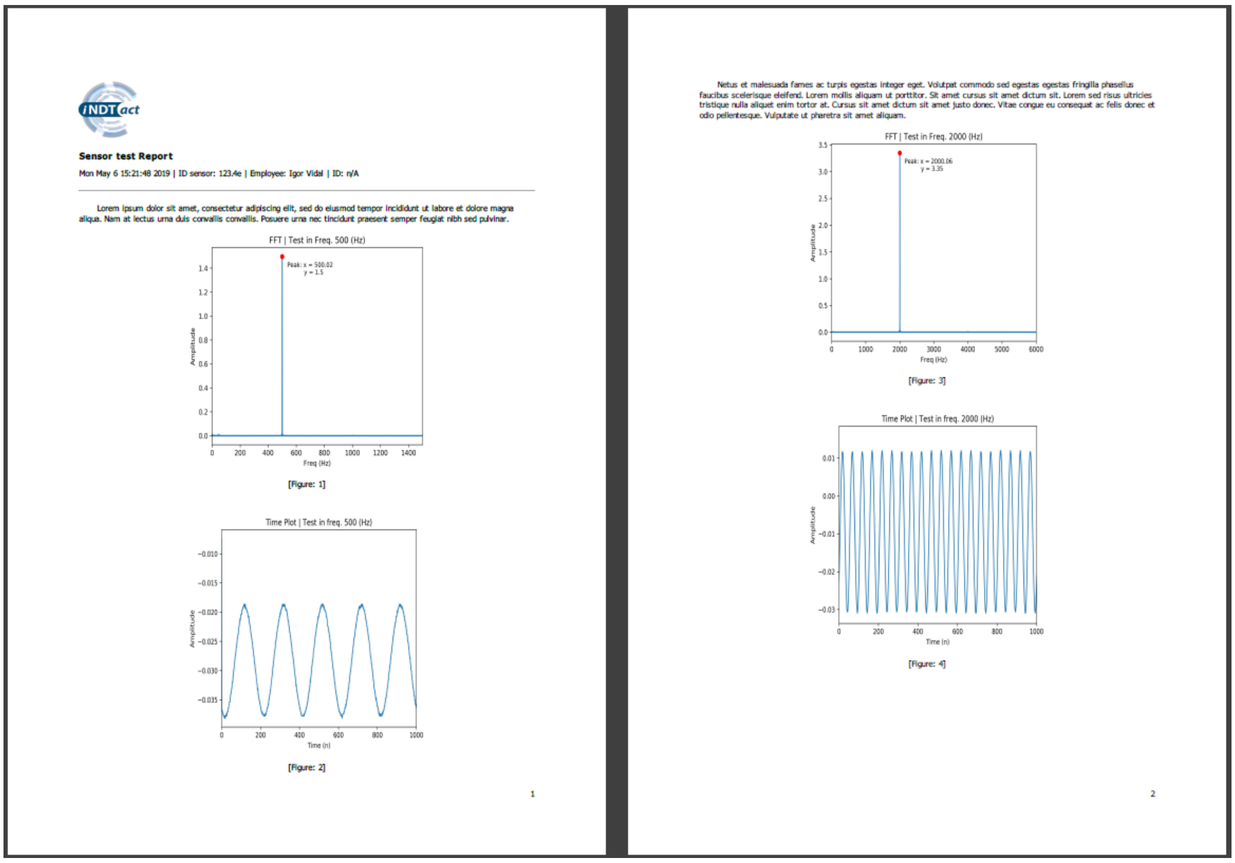
One of the requirements of the project is generate a report about the tests, the report can be used for future analysis or to send to a client to ensure the quality of a sensor. The supervisory provide a function that automatically generate a PDF with the images from the automatic tests available. The user can choose the images that go to the PDF report or no.

The QT/C++ has a *generate_pdf.ui* window in which allow the user to write a message to go to the PDF. There is also an option to select the images to go to the PDF report, the line *<PLOT>figure.png<PLOT>*. This lines with the ID of the figure will appear automatically when open the *generate_pdf.ui* window, all the images available from the automatic tests table will appear. The message below is an example and will generate the PDF on the figure 47.

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Nam at lectus urna duis convallis convallis.
Posuere urna nec tincidunt praesent semper feugiat nibh sed pulvinar.

<PLOT>plot1.png<PLOT>
<PLOT>plot1_time.png<PLOT>

Netus et malesuada fames ac turpis egestas integer eget.
```

```
Volutpat commodo sed egestas egestas fringilla
phasellus faucibus scelerisque eleifend.
Lorem mollis aliquam ut porttitor.
Sit amet cursus sit amet dictum sit.
Lorem sed risus ultricies tristique nulla aliquet enim tortor at.
Cursus sit amet dictum sit amet justo donec.
Vitae congue eu consequat ac felis donec et odio pellentesque.
Vulputate ut pharetra sit amet aliquam.

<PLOT>plot3.png<PLOT>
<PLOT>plot3_time.png<PLOT>
```

Algorithm 5.1 – Input to the *generate_pdf.ui*



Sensor test Report

Mon May 6 15:21:48 2019 | ID sensor: 123.4e | Employee: Igor Vidal | ID: n/A

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nam at lectus urna duis convallis convallis. Posuere urna nec tincidunt praesent semper feugiat nibh sed pulvinar.

[Figure: 1]

[Figure: 2]

1

Netus et malesuada fames ac turpis egestas integer eget. Volutpat commodo sed egestas egestas fringilla phasellus faucibus scelerisque eleifend. Lorem mollis aliquam ut porttitor. Sit amet cursus sit amet dictum sit. Lorem sed risus ultricies tristique nulla aliquet enim tortor at. Cursus sit amet dictum sit amet justo donec. Vitae congue eu consequat ac felis donec et odio pellentesque. Vulputate ut pharetra sit amet aliquam.

[Figure: 3]

[Figure: 4]

2

FFT | Test in Freq. 500 (Hz)
Peak: x = 500.02
y = 1.5
Amplitude
Freq (Hz)
Time Plot | Test in freq. 500 (Hz)
Time (n)
FFT | Test in Freq. 2000 (Hz)
Peak: x = 2000.06
y = 3.35
Time Plot | Test in freq. 2000 (Hz)

Figure 47 – PDF file output. It contains the figures that was selected and the message for the PDF report.

5.3 Log file and supervisory internal error

It is important for a computer program to save the interactions behind the screen, so the user can check the program behavior and the low level iterations that leads for an error. On this project the QT/C++ supervisory provides a log file in which shows all this background data, this means all the: process output and informations when the supervisory open a file or image. It is possible to save on log file just calling *generalSlotsStruct.insertLogStruct.insertLog (str_ to_ log)*. The log file is generated with the date and each entrance has an ID (the input time).

```
$[14:19:41]$ JSON file open <_io.TextIOWrapper
             name='../iNDTact_TestSensorApp/Resources/
             JsonFiles/options_settings.json' mode='r' encoding='cp1252'>
             DWF Version: b'3.9.1'
             Opening first device
             Generating sine wave...
             Starting oscilloscope
             Recording done
             peak_freq_index is at: 6000
             Peak is at: 6000.1800054001615
             Peak is: 7.9652361011962896
             signalsCompareReturnTrue
$[14:19:41]$ Process start with command: python and params:
             ../iNDTact_TestSensorApp/Resources/Python/
             callLabviewByPython.py
$[14:20:00]$ exit status:  exit code: 0
```

Algorithm 5.2 – Log file example

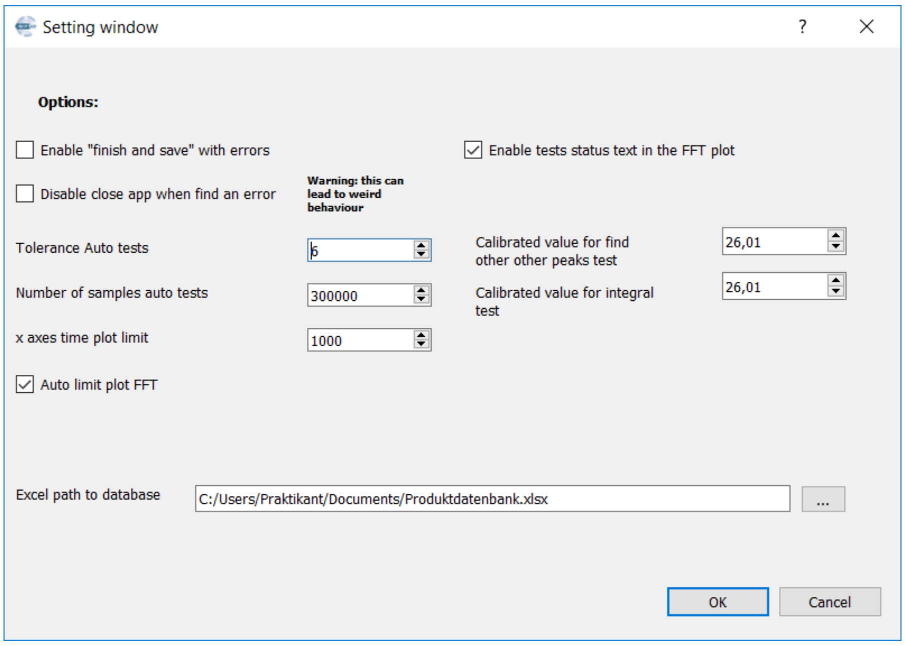
Also the Log file has an important purpose, verify if the program has a top level error, this can be, for example, if the path for a file is incorrect. Once the Log will have all the data about the program behaviour the *generalSlotsStruct. insertLogStruct. insertLog (str_ to_ log)* call *generalSlotsStruct. verifyErrorStruct. verifyErrorAppFunction (str_ to_ check)* in which will check if the string *iNDTact_ TestSensorAppTopErrorDetect* is on the *str_ to_ log* then a error message will appear and the program will stop and close.

5.4 Options window

The options/settings UI (Figure 48) is a window in which allows to choose some important parameters, this can be important once the tests can be different. The way the program does this is to save all the data on a JSON file and this file then is used as database sharing the data with other scripts (python, Labview, ...). The parameters that can be changed are:

- Enable finish and save with test error;
- Disable close app when find a program error (e.g a file can not be open);
- Frequency tolerance of the tests;

- Number of test samples;
- Number of samples time axes on time plot;
- Auto limit plot or Number of samples frequency axes on FFT plot;
- Calibrated value for Local FFT peaks tests;
- Calibrated value for FFT integral tests;
- Path to excel;


Setting window

Options:

☐ Enable "finish and save" with errors

☑ Enable tests status text in the FFT plot

☐ Disable close app when find an error

Warning: this can lead to weird behaviour

Tolerance Auto tests: 5

Calibrated value for find other other peaks test: 26,01

Number of samples auto tests: 300000

Calibrated value for integral test: 26,01

x axes time plot limit: 1000

☑ Auto limit plot FFT

Excel path to database: C:/Users/Praktikant/Documents/Produktdatenbank.xlsx

Figure 48 – Options windows supervisory. This will allow the user to set some settings to the system. All these options are saved on the JSON options file and shared with the Labview and Python scripts.

5.5 Json structure

As explained before the Json database files are important to this project. It is the way that data will be shared between the modules. Also, new tests can be easily inserted, without changes in the QT/C++. This is important to the company to configure a new test without change the main code. It is just necessary to add some lines on the JSON file following the Json structure.

5.5.1 Json Automatic tests

The JSON Auto (Figure 49) is basically a file with many keys, each key has different informations inside. The variable *name* is to show the name of the test on the supervisory.

The variable *language* is the language in which that script will be called, important notice that in the case of Labview language, the supervisory will internally call a python function to call the Labview program. The ID *file path* is where to find the way to the script. The *plot* is where the python will output the figure with the plot, it is possible to insert how many plots are necessary. The easiest way to add a new test is by using the "options" key and fill with the desired frequency to make the test.

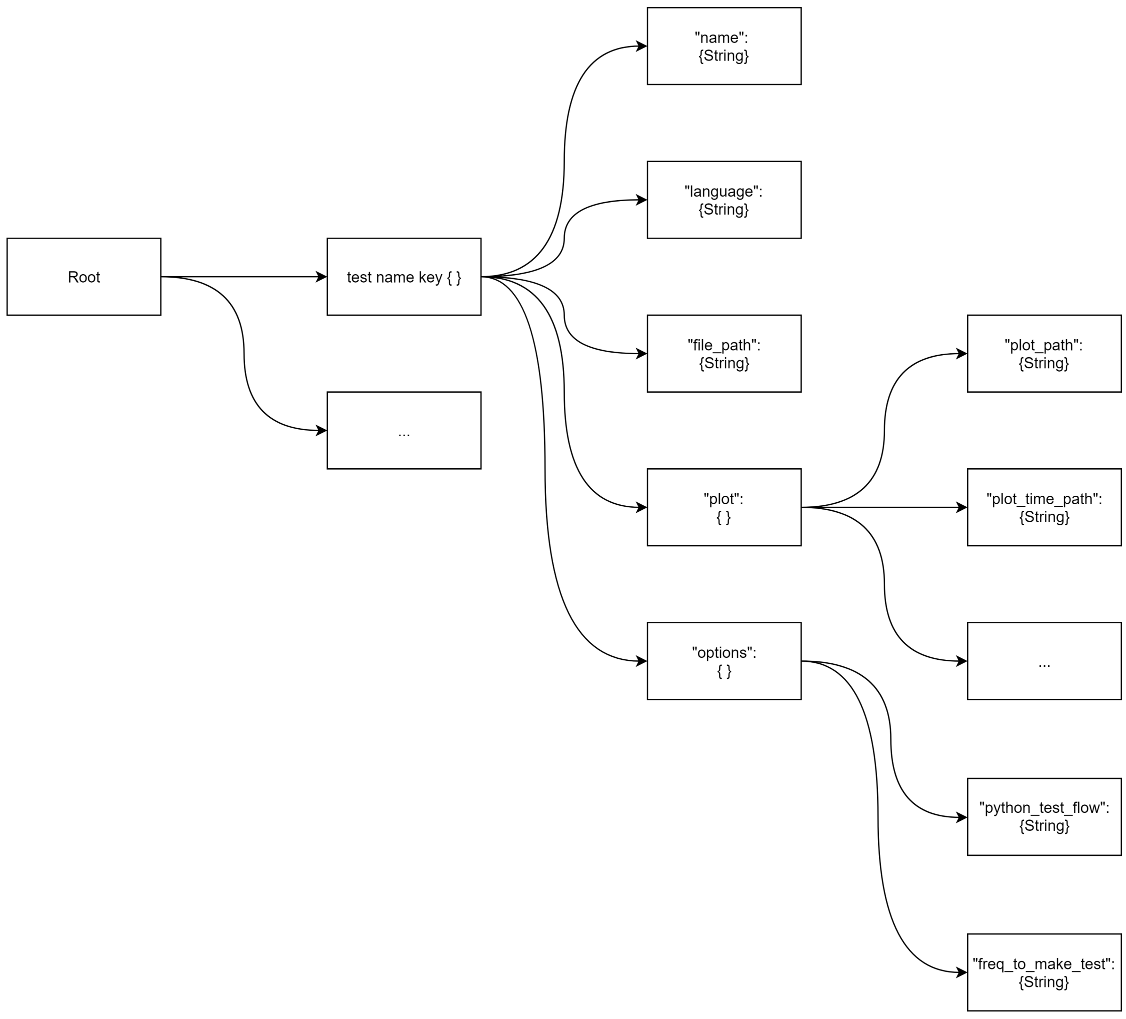


Figure 49 – Json structure automatic tests

5.5.2 Json Manual tests

The manual tests (Figure 50), as the Automatic tests on the subsection above. The variables *Name*, *Help title* and *Help description* are all informations that will appear on the Main Window on the supervisory. The *image path* is the path to the figure for help the user.

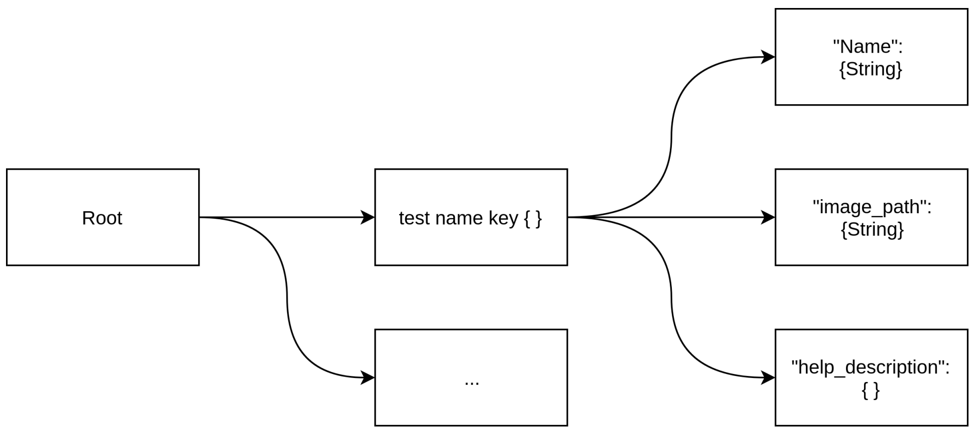


Figure 50 – Json structure manual tests

5.5.3 Json Options

The Json options file (Figure 51) is a kind of database where is saved all the information's about the settings of the supervisory. This allows open and closes the supervisory without change the parameters set. The same of the Json options is used by python/Labview modules to perform the tests, for example, the FFT x-axes limit of the plot can be selected, when python/Labview is generating the FFT figure this information will be checked.

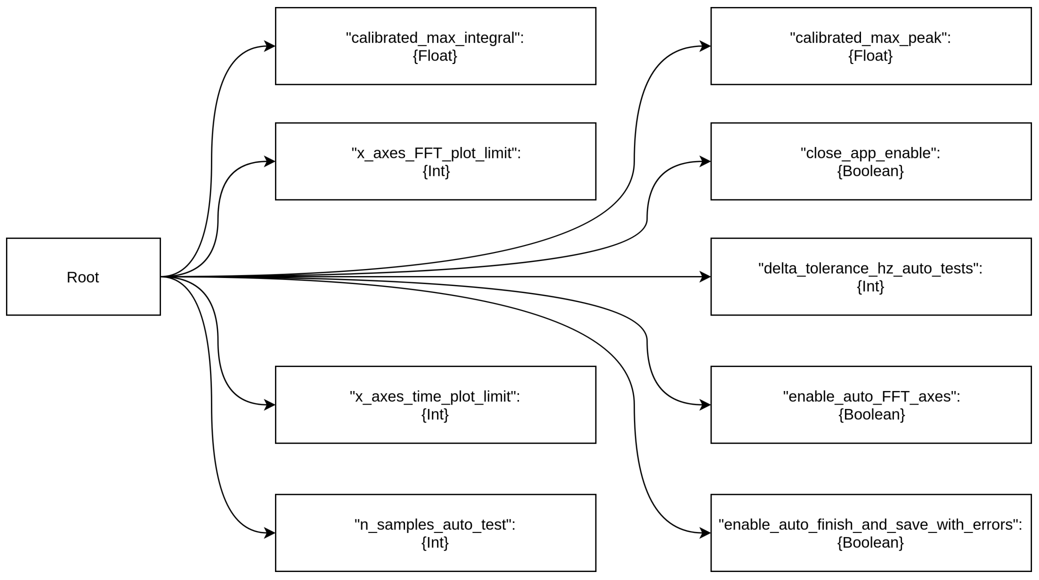


Figure 51 – Json structure options

5.5.4 Json Database

Json provides the possibility to share information between modules. A python module is responsible to send information to the Access database, but the QT/C++ is the one that will join all the informations on two Json files. So this two Json files (Figure 52 and Figure 53) will communicate with this modules.

Two different JSON files are used to save at the database. The *Login Database Json* is responsible to take and share all the initial data about the tests, this is on the Login (initial) window. The *Database Json* is responsible to save all the other details that are produced during the tests.

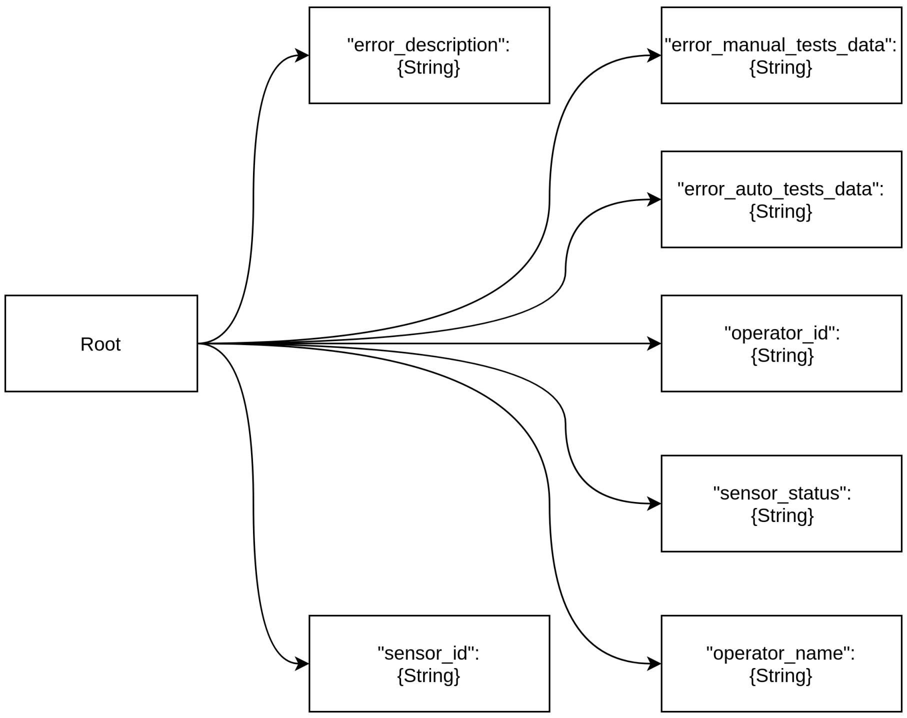

Root
"error_description": {String}
"error_manual_tests_data": {String}
"error_auto_tests_data": {String}
"operator_id": {String}
"sensor_status": {String}
"sensor_id": {String}
"operator_name": {String}

Figure 52 – Json structure database

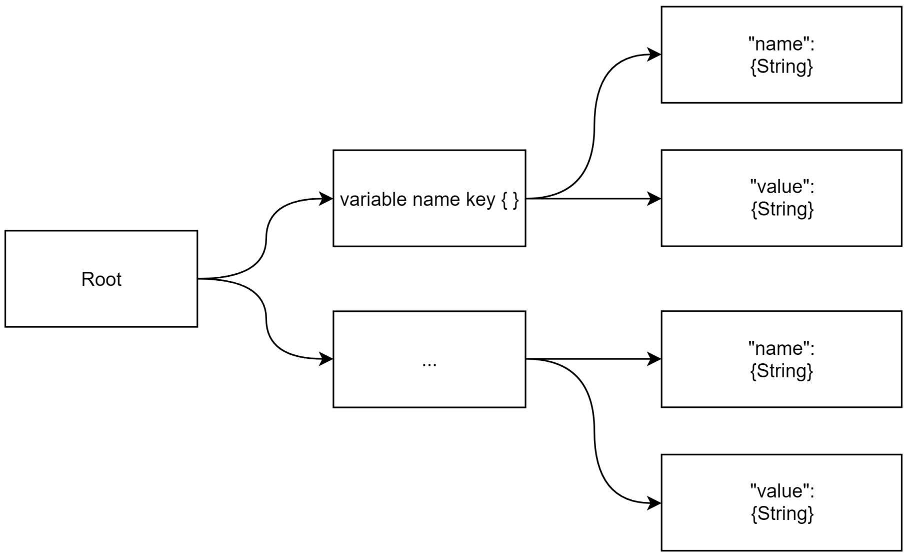


Figure 53 – Json structure Login database

5.6 Results

The automatic tests implemented shows that is possible to analyse if the sensor measure the same frequency that was generated using FFT. The FFT parameters provide ways to analyse if the behaviour of the signal measured match with a sensor well calibrated and working well. The tests showed that analyse the FFT of the signal measured is more reliable than verify the signal in time using cross correlation and RMSE. The smartPredict 2.0 is not available for tests on the moment. However, the tests with the sensor from the company ImpactXs reveal the tests are robust and trustworthy. (Check Appendices C to see the output of the automatic tests).

The proposed method to generate a signal wave worked. The replicable method using a structure born driver (speaker) to generate an audio wave differ from the standard methods. The standard methods are non replicable (e.g.: falling a ball on a structure, breaking a pencil leads, etc). The used method has also the advantage to test each frequency individually, providing a way to increase the quality of the tests.

The supervisory is highly complex with many different modules. Many computational languages was used in order to achieve a functional supervisory using tools that match the requirements of the company and the best for the system. The supervisory connect Python, Labview, QT/C++, Json and Html. The settings of the buttons and labels on the supervisory windows allow easily find the functions, this will guide the user and it will make the user perform the tests as soon as possible. The many functionalities inside the supervisory are a great advantage of the system. The possibility to save on the database the informations of the tests will allow the company to check the records on the future if necessary. The generate PDF function will also help the user to automatically make a report of the tests. It is also easy to change the tests, because the tests are writed inside Python/Labview scripts which the company has background.

The Json files provide a simple way to communicate with other modules. The *automatic tests Json file* and *manual tests Json file* are a great advantage of this platform. With some few lines on this Json files is possible to add a new test. The user does not need to be a programmer to change the manual and automatic tests.

The automatic tests are written outside the QT/C++ supervisory. There are three different ways to make an automatic test: Python, Labview and Python or just Labview. The three different ways to make this tests increase the reliability of the platform, since the company can change the team, however the tests can still be modified, because the chances to find someone with background in Labview or Python is bigger than just one of the tools.

The platform is totally functional and will help the company to test all the new smartPredict 2.0 devices. The quality of the products delivered will increase, since the amount of errors on the products before send to the costumers will decline.

However, it is a hard task to analyze the impact of the platform since smartPredict 2.0 is not ready for tests yet. But the platform is totally functional and without many upgrades, it is possible to start the tests. The platform allows easy changes on the modules and it is easy to add new manual and automatic tests. All the requirements were used to build the platform and many other requirements were introduced by the intern in order to reach the best possible solution. It is important to say that many other solutions were interrupted because of the time constraints, the 6 months was not enough to implement, for example, an Artificial Intelligence automatic test, that could have more accuracy on the analysis, but the actual automatic tests showed to be adequate. The smartPredict 2.0 is a complex device, with the possibility to read many variables, unfortunately not all the variables were actually tested on this platform (e.g.: the temperature), this is because of the time constraints. All the implementation steps were validated by the Indtact experts, even though the intern was free to choose the way. The project was a success from the intern point of view.

6 Conclusion

The project aims to build a platform to test the smartPredict 2.0 sensor. This platform must perform some automatic tests and also guide the employee that is making the tests. So a supervisory must be developed, with the possibility to save the data on a database and generate reports. All the tools used on the platform must follow the company requirements. The solution is a complex software with automatic tests and manual tests. Implemented using many different languages/tools: Python, QT/C++, Lavbview, Json, Html, Access database and others.

This platform allows making tests more quickly than the old system. The old system is basically test everything manually and save all the data on an Excel file. The project makes some automatic tests, but not all the tests are automatic. Visual inspection and current/voltage analysis are manual. A future project can automatize this manual tests, using cameras and robots. But this will increase the complexity of the project and by now (for the actual company amount of sales) it is useless.

Acoustic Emission analysis has a lot to grow and become more popular in the next years. The smartPredict 2.0 is a sensor that matches perfectly with the 4.0 industry, iNDTact GmbH will also grow and the platform of this project will indirectly help the company to grow, since a sensor with better quality will be sent to the customer and in less time.

The automatic tests are robust enough to identify errors on the sensor, but not enough robust to identify small errors. Since Acoustic Emission sensors are super sensitive an Artificial Intelligence approach, for example, a Neural Network, can be interesting in the future, this will ensure that all the sensors are exactly the same. However, it was impossible on this project because there are not enough data available from smartPredict 2.0 at the moment.

The project also was the first iNDTact GmbH using QT/C++, an important tool that has many functionalities and is growing each year. QT/C++ can be important to the company in the future, as an option to Labview.

Many aspects of the project were not introduced here, the size of the project was a challenge for a 6 months internship, but the iNDTact Gmbh support in many aspects of the project. The iNDTact team is highly motivated to provide a great product and will have a promising future.

Bibliography

1 BRITANNICA, i. E. *Elastic wave*. 2016. Accessed on 12/07/2019. Disponível em: <https://www.britannica.com/science/elastic-wave>. (Page 17).

2 TURKASLAN-BULBUL, M. T.; AKIN, O. Computational support for building evaluation: Embedded commissioning model. *Automation in Construction*, v. 15, n. 4, p. 438 – 447, 2006. ISSN 0926-5805. The first conference on the Future of the AEC Industry (BFC05). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0926580505000750>. (Page 18).

3 LI, X. A brief review: acoustic emission method for tool wear monitoring during turning. *Int. Journal of Machine Tools & Manufacture*, 2001. (Page 21).

4 BOYD, J.; VARLEY, J. The uses of passive measurement of acoustic emissions from chemical engineering processes. *Chemical Engineering Science*, v. 56, p. 1749–1767, 03 2001. (Page 21).

5 HELLIER, C. *Handbook of Nondestructive Evaluation, Second Edition*. <country>US</country>: McGraw-Hill Professional, 2012. -1 p. ISBN 0071777148. Disponível em: <https://mhebooklibrary.com/doi/book/10.1036/9780071777131>. (Pages 21 and 42).

6 KEPRT, J.; BENEš, P. A comparison of ae sensor calibration methods. 05 2019. (Page 23).

7 GABAY, J. *Piezo Sensor Fundamentals: Shock and Vibration*. 2011. Accessed on 17/05/2019. Disponível em: <https://www.digikey.com/en/articles/techzone/2011/dec/fundamentals-of-piezoelectric-shock-and-vibration-sensors>. (Page 23).

8 GMBH indtact. *About iNDTact*. 2013. Accessed on 17/05/2019. Disponível em: <https://www.indtact.de/>. (Page 24).

9 GROUP, Q. *About QT*. 1995. Accessed on 17/05/2019. Disponível em: <https://www.qt.io/company>. (Page 31).

10 GROUP, Q. *Signals & Slots*. 1995. Accessed on 17/05/2019. Disponível em: <https://doc.qt.io/qt-5/signalsandslots.html>. (Page 32).

11 JSON. *Introducing JSON*. 1995. Accessed on 17/05/2019. Disponível em: <https://www.json.org/>. (Page 34).

12 NI. *Labview*. 1995. Accessed on 17/05/2019. Disponível em: <http://www.ni.com>. (Page 34).

13 LASI, H. et al. Industry 4.0. *Business & Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK*, v. 6, n. 4, p. 239–242, 2014. Disponível em: <https://EconPapers.repec.org/RePEc:spr:binfse:v:6:y:2014:i:4:p:239-242>. (Page 35).

14 HOEY, B. *6 Important Industry 4.0 Statistics to Know*. 2018. Accessed on 27/06/2019. Disponível em: <https://blog.flexis.com/6-important-industry-4.0-statistics-to-know>. (Page 35).

15 LEAVENS, G. T. et al. *Programming is Writing: Why Student Programs must be Carefully Read*. [S.l.], 1998. Disponível em: <ftp://ftp.cs.iastate.edu/pub/techreports/TR97-23/TR.ps.gz>. (Page 36).

16 DIGILENT. *Analog Discovery 2: Reference Manual*. 1995. Accessed on 17/05/2019. Disponível em: <https://reference.digilentinc.com/reference/instrumentation/analog-discovery-2/reference-manual>. (Page 38).

17 SMITH, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego, CA, USA: California Technical Publishing, 1997. ISBN 0-9660176-3-3. (Page 40).

18 LATHI, B. P. *Linear Systems and Signals*. 2nd. ed. New York, NY, USA: Oxford University Press, Inc., 2009. ISBN 0195392566, 9780195392562. (Page 40).

19 . *Fast Fourier Transformation FFT - Basics*. 2019. Accessed on 17/06/2019. Disponível em: <https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft>. (Page 40).

20 INSTRUMENTS, N. *Understanding FFTs and Windowing*. . Accessed on 24/06/2019. Disponível em: <http://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf>. (Page 40).

21 KIRACOFE, D. *Using cross-correlation to line up two periodic signals*. 1995. Accessed on 17/05/2019. Disponível em: <http://www.mechanicalvibration.com/Using_cross_correlation_lin.html>. (Page 42).

22 . *The Piezoelectric Effect*. 2019. Accessed on 17/06/2019. Disponível em: <https://www.nanomotion.com/piezo-ceramic-motor-technology/piezoelectric-effect/>. (Page 42).

23 AGUIAR, P. et al. In-process grinding monitoring through acoustic emission. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v. 28, 03 2006. (Page 48).

24 SOJER, M. *Reusing open source code: value creation and value appropriation perspectives on knowledge reuse*. Tese (Doutorado) — Technical University Munich, 2011. Disponível em: <http://d-nb.info/100731219X>. (Page 56).

25 GALITZ, W. O. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. New York, NY, USA: John Wiley & Sons, Inc., 2007. ISBN 0470053429. (Page 59).

26 GITHUB @colebemis. *Simply beautiful open source icons*. 2019. Accessed on 10/07/2019. Disponível em: <https://feathericons.com/>. (Page 59).

27 4MATTED. *Controlling a VI using Python using LabVIEW*. 2017. Accessed on 17/05/2019. Disponível em: <https://forums.ni.com/t5/user/viewprofilepage/user-id/315583?profile.language=en>. (Page 99).

28 KAWANO, W. *Migrando de C para C++: Guia Prático de Programação*. 2010. (Page 101).

Appendix

APPENDIX A – Functions and Slots

The following functions/slots are part of the QT/C++ supervisory. The idea of this appendix is to provide an overview of the functions inside the supervisory. Many functions were not included. This is not a datasheet but a way to understand how are the things inside the supervisory.

A.1 Main Window class

startAutoTestProcessMain()

```
void MainWindow::startAutoTestProcessMain(  QString command
                                            QStringList params,
                                            int row,
                                            int number_of_tests)
```

Start some auto test with the parameters *"command"* and *"params"*. The parameters *"row"* and *"number_of_tests"* are important when it want to automatically make all the tests by one time. When a single test is performed can be used the definition *number_of_tests = SINGLE_AUTO_TEST (= 1).*

startProcessMain()

```
void MainWindow::startProcessMain(  QString command
                                    QStringList params)
```

Start an external process as the function *MainWindow :: startAutoTestProcessMain*, the main difference is that this function does not have all the settings to analyze as an automatic test. It is basically used by the supervisory to call the python script to send data to the database.

setupAutoTestProcessMain()

```
void MainWindow::setupAutoTestProcessMain(  int row
                                            int number_of_tests)
```

Setup the environment to call the function *MainWindow :: startAutoTestProcess-Main*. This means that the function will read the JSON key from that test (passed as *"row"*) and take the information's *command* and *params*, necessary to *MainWindow :: startAutoTestProcessMain ()*.

tableAutoItemClicked()

```
void MainWindow::tableAutoItemClicked(  int row
                                        int column)
```

Always that the Auto table is clicked this function will be called, the parameters *row* and *column* will be analyzed to see the actions, the important columns are:

- TEST_NAME_COLUMN_AUTO_TABLE
- MAKE_TEST_BUTTON_COLUMN_AUTO_TABLE;
- RESULTS_PLOT_FFT_BUTTON_COLUMN_AUTO_TABLE;

setImageMainWindow()

```
void MainWindow::setImageMainWindow(  QString image_path
                                      QLabel &label)
```

It is called to set an image on the *Main Window UI*. The label must be passed as a pointer, for example, **ui->label*.

tableItemClicked()

```
void MainWindow::tableItemClicked(  int row
                                    int column)
```

Similar to *MainWindow :: tableAutoItemClicked*, but for the Manual table. The important columns will be:

- STATUS_COLUMN_MANUAL_TABLE;
- HELP_COLUMN_MANUAL_TABLE;

update_screen()

```
void MainWindow::update_screen(   )
```

Will setup all the *MainWindow UI*. This means:

- configure all the images;
- configure the Auto table;
- configure the Manual table;

To setup all the table is necessary to read the relative JSON files and get the information's inside.

receivedSlotToInsertDatabase()

```
void MainWindow::receivedSlotToInsertDatabase(   QString text_error
                                                 QString status)
```

This function will receive the necessary information's to send to the database. The *text_ error* is the description of what is not working provided by the user. The information's will be stored in a JSON file and then a python script is called to send to the database.

on_settingsButton_clicked()

```
void MainWindow::on_settingsButton_clicked(   )
```

Open the *options Window UI*, disable the *Main Window UI*.

on_reportErrorButton_clicked()

```
void MainWindow::on_reportErrorButton_clicked(   )
```

Open the *Error Detect Window UI*, disable the *Main Window UI*.

on_finishAndSaveButton_clicked()

```
void MainWindow::on_finishAndSaveButton_clicked(   )
```

First verify if all the tests are OK, if some of the tests are missing then the supervisory will show a message that something is missing. If everything is OK the function *receivedSlotToInsertDatabase ()* will be called.

getDataFromLoginWindow()

```
void MainWindow::getDataFromLoginWindow(   const QString& sensorID_string
                                           const QString& OperatorName_string,
                                           const QString& OperatorID_string )
```

Take the information's provided by the user on the *Login Window UI* and then show the information's on the respective labels on the *Main Window UI*.

receiveSlotGeneratePDF()

```
void MainWindow::receiveSlotGeneratePDF(   QString textToPDF
                                           QString fileName)
```

This function will make all the necessary steps to generate a PDF file. First, the message that the user wants to write on the PDF is taken and the HTML template file is read. The function joins all this data and saves the PDF file where the user specify.

A.2 Login Window Class

on_buttonBoxW1_accepted()

```
void LoginWindow::on_buttonBoxW1_accepted( )
```

First, check if all the labels are filled by the User. If is not filled than a message of error will appear. If it is OK then the *Main Window UI* is open and the information's of the labels from the Login Window will be sent to the *Main Window UI*.

cleanPlotFolder()

```
void LoginWindow::cleanPlotFolder( )
```

This function will clean the Plot folder, this is important to prevent weird behavior that occurred by tests performed before.

A.3 PDF generator Class

on_saveButton_clicked()

```
void generatePDFwin::on_saveButton_clicked( )
```

Send the *QString* with the message that the User wants to save on the PDF to the *Main Window UI*, also provide a Window to the User select the path and name of the PDF file.

A.4 Settings Class

updateCheckBox()

```
void OptionsWindow::updateCheckBox( QCheckBox & checkBox,
                                    QJsonObject & data_object,
                                    QString JsonKey)
```

Read the JSON file with the settings data and check if the respective *Check Box* widget is already checked or no.

on_buttonBox_accepted()

---

void OptionsWindow::on_buttonBox_accepted( )

Save all the information's set on the Window to a JSON file.

A.5 Error detected Class

on_send_button_W3_clicked()

---

void ErrorDetectWindow::on_send_button_W3_clicked( )

Send the information's of the Window to the *Main Window UI* emitting a signal.

A.6 VerifyErrorStruct struct

verifyErrorAppFunction

---

void VerifyErrorStruct::verifyErrorAppFunction(QString strToVerify)

Verify if the *strToVerify* contains a *iNDTact_ TestSensorAppTopErrorDetect*. If true the supervisory will show an error message and the program will close. This is important to avoid weird behaviors on the program.

A.7 JsonStruct struct

openJsonFileReturnQJsonObject

QJsonObject JsonStruct::openJsonFileReturnQJsonObject(QString file_path)

This open the JSON file, specified at *file_path* and return the content in a *QJsonObject* format.

get_ID_key

```
QString JsonStruct::get_ID_key(   int row
                                  QJsonObject * object )
```

Return the ID *QString* of a specified number ID (*row*) inside a certain ** object.*

get_Json_file_number_tests

```
int JsonStruct::get_Json_file_number_tests(   QJsonObject * object)
```

Return the ID number of ID's inside the parameter ** object.*

A.8 InsertLogStruct struct

insertLog

```
void InsertLogStruct::insertLog(   QString strToLog)
```

Insert the *strToLog* into the log file. It also will generate a new file in case no file match with the current date.

APPENDIX B – The files structure

B.1 iNDTact_TestSensorApp structure

```
/
└─iNDTact_TestSensorApp
   ├─Headers
   ├─Log
   ├─Resources
   ├─Sources
   ├─UI
   ├─README.txt
   └─iNDTact_TestSensorApp.pro.user
```

B.2 Headers

```
/
└─iNDTact_TestSensorApp
   └─Headers
      ├─about_window.h
      ├─error_detect_window.h
      ├─general_iNDTact_TestSensorApp.h
      ├─generate_pdf_window.h
      ├─login_window.h
      ├─main_window.h
      └─options_window.h
```

B.3 UI files structure

```
/
└─iNDTact_TestSensorApp
   └─UI
      ├─about_window.ui
      ├─error_detect_window.ui
      ├─generate_pdf_window.ui
      ├─login_window.ui
      ├─main_window.ui
      └─settings_window.ui
```

B.4 Sources

```
/
└─ iNDTact_TestSensorApp
   └─ Sources
      ├─ about_window.cpp
      ├─ error_detect_window.cpp
      ├─ generate_pdf_window.cpp
      ├─ login_window.cpp
      ├─ main_window.cpp
      ├─ GeneralSlots
      │  ├─ insert_log.cpp
      │  ├─ insert_to_json_file.cpp
      │  ├─ json_functions.cpp
      │  ├─ verify_error_app.cpp
      │  └─ set_image.cpp
      └─ MainWinSlots
         ├─ generate_pdf_main_win.cpp
         ├─ get_data_from_login_window.cpp
         ├─ on_exit_button_clicked.cpp
         ├─ on_finish_and_save_button_clicked.cpp
         ├─ on_new_test_button_clicked.cpp
         ├─ on_report_error_button_clicked.cpp
         ├─ on_settings_button_clicked.cpp
         ├─ process_slots_main_win.cpp
         ├─ received_slot_to_insert_database.cpp
         ├─ set_image_main_window.cpp
         ├─ table_auto_clicked.cpp
         ├─ table_manual_clicked.cpp
         ├─ update_screen.cpp
         └─ zoom_slots.cpp
```

B.5 Resources I

```
/
└─ iNDTact_TestSensorApp
   └─ Resources
      ├─ Html
      ├─ PDF
      │  └─ PDF_template.pdf
      ├─ Json
      │  ├─ auto_table_tests.json
      │  ├─ manual_table_tests.json
      │  ├─ database.json
      │  ├─ options_settings.json
      │  └─ database_info_login.json
      ├─ PCB_images
      │  └─ ...
      ├─ Plots
      │  └─ ...
      ├─ Python
      │  ├─ AutoTest1.py
      │  ├─ ...
      │  ├─ callPythonAutoTest.py
      │  ├─ callLabviewByPython.py
      │  ├─ py_indtact_lib
      │  │  ├─ _init_.py
      │  │  ├─ databaseCommPythonAccess.py
      │  │  ├─ dwfconstants.py
      │  │  ├─ AnalogSensorIn.py
      │  │  ├─ calibrate_auto_test_histogram.py
      │  │  ├─ checkFFT.py
      │  │  ├─ getLabviewDataFromFile.py
      │  │  ├─ jsonFile.py
      │  │  ├─ makeFFT.py
      │  │  ├─ makePlot.py
      │  │  ├─ makeTestAD2.py
      │  │  └─ setupTest.py
      │  └─ ...
      └─ Txt
         └─ About.txt
```

B.6 Resources II

```
/
└── iNDTact_TestSensorApp
    └── Resources
        ├── Labview
        │   ├── checkFFT_subVI.vi
        │   ├── makeFFT_subVI.vi
        │   ├── makeFFTplot_subVI.vi
        │   ├── makeTest_subVI.vi
        │   ├── makeTimePlot_subVI.vi
        │   └── makeTXT_subVI.vi
        └── UI_images
            ├── check.svg
            ├── check_circle.svg
            ├── file.svg
            ├── info.svg
            ├── logo_indtact.svg
            ├── plus.svg
            ├── save.svg
            ├── settings.svg
            ├── slash.svg
            ├── x.svg
            └── zap.svg
```

C Supervisory plots

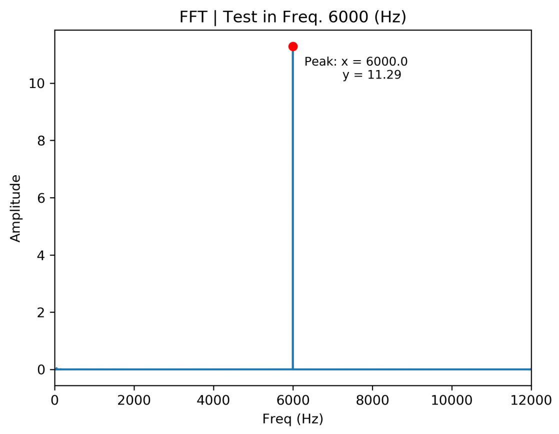


Figure 54 – FFT plot example. This is the output of the supervisory after performing some automatic tests. The image can be observed at the supervisory. The x-axes limits can be changed at the options window.

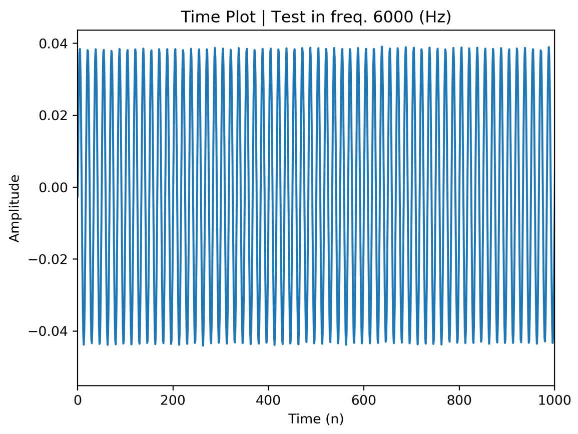


Figure 55 – Time-domain plot example. This is the output of the supervisory after performing some automatic tests. The image can be observed at the supervisory. The x-axes limits can be changed at the options window. Even though all the analysis is on the FFT a time plot can be interesting to the user.

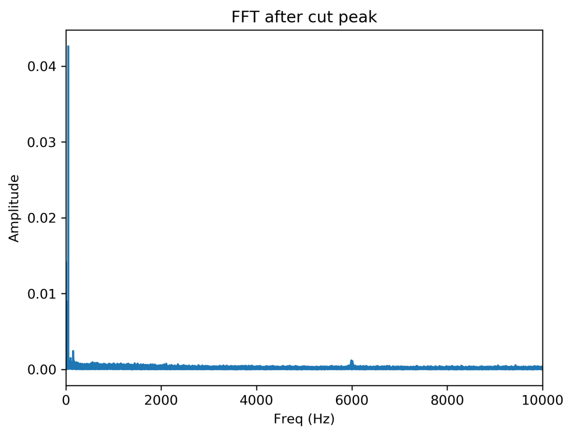


Figure 56 – FFT after cut the global peak. This is used to perform the FFT local peak test and the FFT integral test.

D Algorithms

D.1 FFT script - Python

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.fftpack

def make_hamming(y):

    win = np.hamming(len(y))
    y_win = y * win

    return y_win

# Number of sample points
N = 600
# sample spacing
T = 1.0 / 800.0
x = np.linspace(0.0, N*T, N)
y = np.sin(50.0 * 2.0*np.pi*x) + 0.5*np.sin(80.0 * 2.0*np.pi*x)
y = make_hamming(y)
yf = scipy.fftpack.fft(y)
xf = np.linspace(0.0, 1.0/(2.0*T), N/2)
```

Algorithm D.1 – FFT python example

D.2 Join Labview and QT

Since Labview is not a command-line based code language, it does not provide a friendly way to manipulate Labview codes by other applications. The supervisory program must communicate with Labview programs with the goal to perform the tests on the Analog Discovery 2. The solution was founded by an experienced Labview programmer and that shared the solution at the National Instruments official developers Forum [27]. Since the version 2012 Labview provides ActiveX support, ActiveX is the general name for a range of Microsoft technologies that allows you to reuse code and link individual programs together to suit your computing needs. First, it is necessary to set the Python parameters.

```python
import win32com.client  # Python ActiveX Client

LabVIEW = win32com.client.Dispatch("Labview.Application")
```

```
VI = LabVIEW.getvireference(ABSOLUT_PATH_TO_LABVIEW_CODE)  # Path to
    LabVIEW VI
```

Algorithm D.2 – Join Labview and QT: parameters

And then it is possible to call the Labview code:

```
VI._FlagAsMethod("Call")  # Flag "Call" as Method
VI.Call()  # Run the VI
```

Algorithm D.3 – Join Labview and QT parameters: call labview

It is important to active the ActiveX on Labview at *Tools > Options > Server Options > check ActiveX*.

D.3 Run python scripts using QT

QT provides a library to run scripts outside the main application. This can be interesting in many systems that are impossible to change and have external modules. The platform for this project has many external modules. First of all, it is necessary to include the library:

```
#include <QProcess>
```

Then it is necessary to configure the environment to call the script, on the code bellow a python script will be called with the parameters at the variable *params*.

```
QString  command("python");
QStringList params = QStringList() << "databaseCommPythonAccess.py" <<
                                      ui->labelProductID->text() <<
                                       "Not working " <<
                                       ui->operatorName->text() <<
                                       ui->operatorID->text() <<
                                       text_error;
```

Algorithm D.4 – Configure Process in QT

Basically is doing the same as calling the function by the terminal, using:

```
python databaseCommPythonAccess.py params1 params2 ...
```

With the environment configured the process can be created and called:

```
QProcess *process = new QProcess();
process->start(command, params);
```

Algorithm D.5 – Run process

With the *process->waitForFinished(-1)* QT blocks the thread until the process is over and then read all the output. It is also important to close the project.

```
process->waitForFinished(-1);
QString output(process->readAllStandardOutput());
qDebug() << output << endl;
process->close();
```

Algorithm D.6 – Python example

The *process->waitForFinished(-1)* is not always good once if the script do not return the QT application will collapse. To avoid this a *connect* can be implemented as follow:

```
connect(process, QOverload<int,
        QProcess::ExitStatus>::of(&QProcess::finished),
    [=](int exitCode, QProcess::ExitStatus exitStatus){
        QString output(process->readAllStandardOutput());
        qDebug() << "OUTPUT:" << output << endl;
        process->close();
    }
```

Algorithm D.7 – Python example

Always that the process is finished this *connect* is called. This kind of function declared on the same line of the connect declaration is a lambda function [28], first defined at C++ 11.

D.4 Join Python and Database

Communicate with the database is one of the most important functions of the supervisory program. Once save the informations of each test is one of the main purposes of this system. The Supervisory wrote in QT will call a python script and this python script will communicate with the database. The database is written in Microsoft Access and it is being developed by another department of the iNDTact company. The QT supervisory will first write everything in a JSON, so all the data that must be stored in the database will be on a JSON file. As follow:

```
with open('test.json') as json_file:
    data = json.load(json_file)

sensor_ID = data['sensor_id']
sensor_status = data['sensor_status']
operator_name = data['operator_name']
operator_ID = data['operator_id']
error_description = data['error_description']
```

Algorithm D.8 – Read Json file using python

Then connect with the database, with a code that is basically provided by the *accdb* assistant:

```
conn = pyodbc.connect('DRIVER={Microsoft Access Driver (*.mdb)};UID=admin;
    UserCommitSync=Yes;Threads=3;SafeTransactions=0;PageTimeout=5;
    MaxScanRows=8;MaxBufferSize=2048;{FIL=MS Access};DriverId=25;DefaultDir=
    C:/Users/Praktikant/Documents;DBQ=C:/Users/Praktikant/Documents/testdb.
    mdb;')
```

Algorithm D.9 – Connect python with database

And then insert on the connect database in a SQL format:

```
cursor.execute(
    """
    INSERT INTO sensorTestDatabase
    (sensor_ID, sensor_status, operator_name, operator_ID,
    error_description)
    VALUES (?,?,?,?,?)
    """,
    (sensor_ID, sensor_status, operator_name, operator_ID,
    error_description)
)
```

Algorithm D.10 – Python insert data to database

The message will be stored in the database as follow:

sensor_ID	sensor_status	operator_name	...	error_description
...	...	...	...	...
"sensor2019.1"	"Not working"	"Employee name"	...	"Capacitor missing"

Table 1 – Database Access example, the new lines will be inserted at the end of the database, keeping the old entrances on the top.

D.5 Json and QT

Reading and Writing information's on a JSON file is important to reach the desired system. QT provide a library that make this task. The libraries: *QJsonArray*, *QJsonDocument*, *QJsonObject* and *QJsonValue*.

D.5.1 Reading JSON file

First, the script must open the JSON file and read the content. The content is saved on a string. The string is then converted to a JSON format file. A JSON object is a list of key-value pairs, where the keys are unique strings and the values are represented by a *QJsonValue*.

```
QJsonObject MainWindow::openJsonFileReturnQJsonObject(QString file_path){
    QString val;
    QFile file(file_path);
    file.open(QIODevice::ReadOnly | QIODevice::Text);
    val = file.readAll();
    file.close();
    QJsonDocument data = QJsonDocument::fromJson(val.toUtf8());
    return data.object();
}
```

Algorithm D.11 – QT read JSON file

Then the value inside each part of the JSON can be accessed. For example, on the script bellow the JSON object *data_object_ManualTableJson* is used to convert to an array and take the size of this array.

```
MainWindow::data_object_ManualTableJson = openJsonFileReturnQJsonObject("
    ../data_table_automated_tests.json");

int json_idlist_size = data_object_ManualTableJson["IDlist"].toArray().size
    ();
```

Algorithm D.12 – QT using the Json value

D.5.2 Writing a JSON file

Write a JSON file that will be important to save the informations on the database and also to the settings JSON file. The QT also provides a library to help this task. First, it is necessary to create an object to insert the information on the specific JSON key:

```
QJsonObject recordObject;
recordObject.insert("sensor_id", QJsonValue::fromVariant(ui->
                    labelProductID->text()));
recordObject.insert("sensor_status", QJsonValue::fromVariant(status));
recordObject.insert("operator_name", QJsonValue::fromVariant(ui->
                    operatorName->text()));
recordObject.insert("operator_id", QJsonValue::fromVariant(ui->
                     operatorID->text()));
recordObject.insert("error_description", QJsonValue::fromVariant(
                    text_error));
```

Algorithm D.13 – Creating an object

Then the information is recorded inside a file. The way the information is saved is similar to a normal *.txt* file.

```
QJsonDocument doc(recordObject);
QFile jsonFile("../test.json");
jsonFile.open(QFile::WriteOnly);
jsonFile.write(doc.toJson());
jsonFile.close();
```

Algorithm D.14 – QT writing on JSON file

D.6 Export to PDF

A way to export as a PDF file is important to generate reports, to print and to share with clients. QT provides many different ways to perform this task. On this project was used an HTML script which is transformed into a PDF file. The HTML script is a file apart from the QT code, this allows the company to change the HTML template without change the QT code.

The HTML template is written in order to have the following setup:

- a header with the informations of the sensor and the test;
- a text that the user can set;
- the plots of the automatic tests performed until that moment;

The HTML template is on the following code. All the *%1 %2 %3 %...* are the places where the QT will change for an argument.

```
<p>
    <img src= '../iNDTact_TestSensorApp/UI_images/logo_indtact.png' alt=''
    width='100' height='100' />
</p>
    <h3>Sensor test Report</h3>
<p>
    %1 | ID sensor: %2 | Employee: %3 | ID: %4
</p>
<hr />
<p style='text-align: justify;'>
    %5
</p>
```

Algorithm D.15 – HTML template

E Json examples

E.1 Automatic tests

```
{
  "AutoTest1": {
      "name": "Test 1",
       "language": "python",
       "file_path": "../iNDTact_TestSensorApp/Resources/Python/AutoTest1.
  py",
       "plot":{
           "plot_path": "../iNDTact_TestSensorApp/Resources/Plots/plot1.
  png",
           "plot_time_path": "../iNDTact_TestSensorApp/Resources/Plots/
  plot1_time.png"
       }
   },
  "AutoTest2": {
       "name": "Test 2",
       "language": "python",
       "file_path": "../iNDTact_TestSensorApp/Resources/Python/AutoTest2.
  py",
       "plot":{
           "plot_path": "../iNDTact_TestSensorApp/Resources/Plots/plot2.
  png",
           "plot_time_path": "../iNDTact_TestSensorApp/Resources/Plots/
  plot2_time.png"
       }
}
```

Algorithm E.1 – Python example

E.2 Manual tests

```
{
  "Test1": {
       "Name": "Test 1",
       "image_path": "../iNDTact_TestSensorApp/PCB_images/PCB_example_1.
  jpg",
       "help_title": "Test 1 help",
       "help_description": "Pellentesque in consequat nulla, nec iaculis
  velit In lobortis neque et malesuada consequat. Maecenas sed neque augue
  ."
       },
   {
        [...]
    }
}
```

Algorithm E.2 – Json manual tests example

E.3 Database

```
{
    "error_auto_tests_data": " Test 1: ok ; Test 2: not tested ; Test 3:
   not tested ; Test 4: not tested ; Test 5: not tested ; Test Labview: not
    tested ; Test 7: not tested ; Test Labview 2: not tested ;",
    "error_description": "error test",
    "error_manual_tests_data": " Test 1: ok ; Test 2: ok ; Test 3: ok ;
   Test 4: ok ; Test 5: ok ; Test 6: ok ; Test 7: ok ; Test 8: ok ; Test 9:
    ok ;",
    "operator_id": "none",
    "operator_name": "Igor",
    "sensor_id": "1234",
    "sensor_status": "It is not working"
}
```

Algorithm E.3 – Json database example

E.4 Database Login

```
{
    "_Artid": {
        "name": "ARTID",
        "value": "1"
    },
    "_Charge": {
        "name": "Charge",
        "value": "E"
    },
    "_Empfanger": {
        "name": "Empfanger",
        "value": "Intern"
    },
    "_Employee_id": {
        "name": "Employee ID",
        "value": "1234.3"
    }
}
```

Algorithm E.4 – Json login database example

E.5 Options

```
{
    "calibrated_max_integral": 26.01,
    "calibrated_max_peak": 26.01,
    "close_app_enable": "false",
    "delta_tolerance_hz_auto_tests": 6,
    "enable_auto_FFT_axes": "true",
    "enable_auto_finish_and_save_with_errors": "false",
    "n_samples_auto_test": 300000,
    "x_axes_FFT_plot_limit": 7000,
    "x_axes_time_plot_limit": 1000
}
```

Algorithm E.5 – Json options example