Waqas Shakeel

Automated testing with Wireless Communication in the digitalised industry

A case study of Mirka Oy

School of Technology and Innovation
Industrial Digitalisation (InDi)

Vaasa 2020

ACKNOWLEDGEMENT

The feeling that this master's degree is coming to an end gives me immense relief and joy! It is like a dream coming true. Looking back, I can say with certainty that leaving the comfort of homeland, leaving the loved ones behind, and moving to a new land to fulfil the long-standing desire of completing higher education has paid off well. This achievement would not have been possible without the support and assistance from a number of people.

Firstly, I would like to pass my gratitude to my supervisor Professor Mohammed Elmusrati for his support and guidance. The opportunity I got to learn from him - within and outside of the classroom- has been very beneficial. I cannot thank him enough for his valuable suggestions, availability, and reviewing the work often at short notices. I am also thankful to the other faculty members and staff at the University of Vaasa.

I would also like to thank Mirka Oy for giving me an opportunity to work with them and write this master thesis. The work culture, creativity, zeal, and enthusiasm are surely the trademark of Mirka. My first and foremost gratitude is for Caj Nordström and Veli-Pekka Västi for believing in me, giving this opportunity, and ensuring the close interaction during the work. In the same breath, I would also like to acknowledge Andreas Storbjörk - my former colleague at Mirka - for sharing his knowledge and keenness to help whenever needed. A very special thanks go to my instructor Mårten Wikman for guiding this work, mentoring, reviewing, and providing invaluable suggestions throughout this dissertation work. The completion of this work would not have been possible without him.

Moreover, I would like to register my appreciation for the whole Mirka family for their welcoming spirit. The rejuvenating discussions during the office hours, coffee, and lunch breaks were revitalizing and sharpened my perspective. I cannot resist thanking them all for taking every possibility of teaching me a word or two of Swedish. I am looking forward to speaking with you all in Swedish very soon!!!

Lastly, I would like to thank my family for being supportive to me in all phases of my life especially to my brother and father, who are the biggest motivation behind all of my achievements.

Waqas Shakeel
20 April 2020

UNIVERSITY OF VAASA
School of Technology and Innovation
Author: Waqas Shakeel
Title of the Thesis: Automated testing with wireless communication in digitalised industry: A case study of Mirka Oy
Degree: Master's Programme in Industrial Digitalisation
Programme: Industrial Digitalisation
Supervisor: Professor Mohammed Elmusrati
Evaluator Professor Timo Mantere
Instructor: Mårten Wikman
Year: 2020 **Pages**: 80

ABSTRACT:

Advanced automation technologies are changing the dynamics of the process and manufacturing industries. Product development processes are becoming smarter with the application of intelligent solutions and automated testing. The industry 4.0 concept of centralized control for industrial devices results in a rapid increase in the demand for the industrial Internet of Things (IoT) and cordless machines. Wireless communication protocols are integral to the functioning of such devices.

This thesis work is performed with Mirka Oy during the development process of a smart industrial cordless tool. Various available short-range wireless communication protocols are studied to find out the best possible solution to match the product requirements. Besides, an automated testing platform is developed to verify and validate the functional description of the devices. All the stages, starting from the types of embedded system testing, device test requirements, test case designing leading to a comprehensive testing platform are explained. Results generated by the automated platform are analysed, which shows that all the test execution is successful.

The successful implantation of this automated testing platform would significantly increase the efficiency of the development and testing process. Moreover, this dissertation highlights further development in terms of the application of the Artificial Intelligence (AI) and Machine learning (ML) technique for smarter testing processes and increase the overall performance of the testing framework.

KEYWORDS: wireless communication, embedded system testing, automated testing, robot framework, industrial automation, Industrial digitalisation, industry 4.0

Contents

Pictures

Figures

Tables

Abbreviations

AI	Artificial Intelligence
API	Application programmer interface
AP	Access Point
BSS	Base Service Set
CI/CD	Continuous Integration/ Continuous Deployment
Cmd prompt	Command Prompt
COM port	Communication Port
CPU	Central Processing Unit
FFD	Full Function Device
HTML	Hypertext Markup Language
IAA	Industrial Automation applications
ICTs	Information and communication technologies
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of things
ISM	Industrial, scientific and medical
I/O	Input/output
IP	Internet protocol, intellectual property
IPv4	Internet protocol, version 4
ML	Machine Learning
PC	Personal computer
RFD	Reduced Function Device
RF	Robot Framework
Wi-Fi	Wireless Fidelity
WNs	Wireless networks
XML	eXtensible markup language

1 Introduction

The industrial sector takes a major part in the economic growth and development of any country. In the recent past, traditional industrial communication standards such as data buses like Modbus and Profibus are replaced by Ethernet. Due to easy configuration, low cost, and a supportive environment for an extensive range of applications, it becomes a universally supported solution for industrial use. Moreover, it can also encapsulate various protocols for industrial equipment, which makes it suitable for applications with short-range communication requirements. However, despite these advantages, having drawbacks like cabling and limited range restrict its use for smart cordless industrial tools. Wireless communication can provide a solution to this problem. Wireless communication is helping to automate industry in several ways. On one hand, these solutions are providing mobility and ease of installation while on the other hand, they are cheaper compared to the wired solutions.

The concept of “industry 4.0” is also encouraging to utilize advanced information and communication protocols to develop a more digitalized manufacturing process. Devices and machines are becoming smarter and intelligent with the capability to communicate with each other by using wireless communication protocols. Industry 4.0 is a platform where the combination of sensors, actuators, controllers, and communication technology along with cyber-physical systems forms industrial devices that have the potential to feed information to the system results in a more efficient manufacturing process (Aiman et al., 2016). The small industrial cordless machine usually consists of various sensors, actuators, and power sources that require wireless connectivity to send and receive a stream of small data packets. There are various wireless solutions available to fulfil requirements for short-range communication with diverse characteristics. To concur later communication issues i.e. range, data throughput, etc. it's highly important to consider these factors in the planning phase.

Likewise, Automation is also widely in use for validation and verification of the response received from various industrial machines. Automated testing helps to get rapid testing

feedback and results compilation, right from the development to deployment phase. Nowadays, automated testing frameworks are a major part of the firmware development process for embedded machines. During the firmware development phase, it helps to find bugs and error on the initial stage, whereas at later stages it is used for system testing. Acceptance testing helps to validate machine functionality during and at the end of the development process.

1.1 Objective

The objective of this thesis is to develop an automated testing framework to perform acceptance testing for embedded devices with wireless communication capabilities. Moreover, various short-range communication protocols are conversed to help in the selection of suitable wireless protocol according to device requirements during the planning and development phase.

Technically these embedded devices contain sensors, actuators, power source, and Microcontroller with wireless communication protocols to send data and receive the instructions.

The summary of the thesis objectives can be stated as:

1. Design and plan an entire testing platform including the interfaces for serial and wireless communication to machine
2. Design the basic working structure of the test framework
3. Designing and implementation of the Robot Framework test library (explained in section 2.6.1)
4. Execute functional testing on the machine under test, through serial and wireless communication protocols between the machine and the automated testing framework.

Figure 1 shows the basic concept of testing framework where the device under test and testing library communicating through wireless and serial protocols.

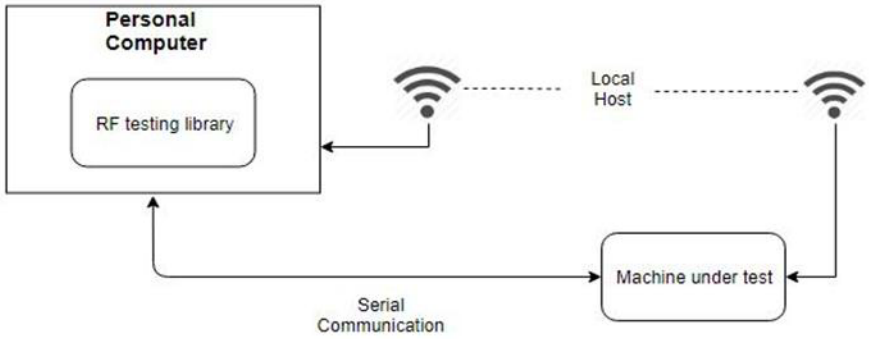


Figure 1. The Basic Architecture of the System

1.2 Structure

The structure of the thesis consists of mainly three parts, starting from the theoretical background, followed by the functional testing requirements and designing of test cases, lastly the test implementation part. In the theory part, relevant theory and different possible solutions to the problem are studied to find a concrete and generalized solution with the capability to extend it for later development.

The 1st chapter presents an introduction to the wireless automation and automated testing along with a generalized overview of the basic system architecture. This chapter also highlights the intended contributions of this study. Chapter 2 is divided into two parts. The first half provides an overview of different available short-range wireless communication protocols. In the second half, embedded system testing is explained along with different types and levels, followed by a suggestion on an appropriate framework for a testing platform, according to requirements. Also, an example test is executed to describe the working of the suggested automated testing platform, Robot framework.

In chapter 3 the target hardware machine is introduced along with its different functionalities. The full process flow is discussed in terms of previous and the updated working approach. Whereas, after establishing the description of the functionality of the device

needed to be tested, chapter 4 forms a test plan that helps in the designing of the Robot Frame testing library.

Chapter 5 includes the test implementation phase along with an analysis of the results across the functional requirement of the machine. Finally, chapter 6 presents the conclusion of the study, followed by further scope and suggestions in the field of automated testing.

1.3 Contribution

This research is done to highlight various available wireless communication protocols suitable for small cordless industrial machines. Also, their range, data throughput, pros, and cons are discussed in detail. Furthermore different types & approaches to testing are explained here especially related to embedded systems. This master thesis would help to figure out the best possible wireless communication protocol among the available options as per device functional requirements. Moreover, it will also serve as a guideline for designing various stages of automated testing framework for embedded devices such as defining the test requirement, test designing and implementation, and validation of results.

2 Wireless Automation and Automated testing

2.1 Wireless automation - introduction and applications

Wireless automation is an alternative to the traditional automated systems, where instead of wires, wireless communication protocols and devices are adopted as a medium of transferring data and instruction to linked devices and sub-systems. In recent years "Industrial Wireless Automation" has shown vast market growth, and with the advancement, in Wireless networks (WNs) many applications have started utilizing it. On one hand, using WNs for automation helps to reduce the installation issues, resulting in simple architecture while on the other hand, it offers easy mobility to devices. In the future, WNs would take a major part in the concepts of industry 4.0 (Li et al., 2017). Gnad et al. (2008) suggest that wireless communication, in itself, is sensitive to various parameters if compared to wired communication. It becomes even more delicate and requires a considerable understanding of parameters and their effects when it comes to the use of wireless communication in an industrial setting. These parameters have enormous influence to evaluate the reliability of device features for different wireless solutions.

With the development of industry 4.0 concept, new frameworks such as smart factories, and networking manufacturing are evolving. Also, advanced information and communication technologies (ICTs) are introduced to manufacturing industries to fulfill higher productivity demands and to assimilate with green production concepts. Some new ICTs are Wireless cloud networks, industrial IoT, big data, and high performance embedded systems (Li et al., 2017).

In general, traditional wired communication is failing to fulfill the latest industrial requirements such as mobility and flexibility. It is much easier to utilize the cordless sensors and actuators to collect data and handling of the system due to their easy movability and installation procedure.

In Industrial Automation Applications (IAA), wireless communication systems are already in use for more than a decade. These applications can be further categorized in terms of

the utilization in the process automation industry and discrete factory automation. The traditional use of wireless systems is to make the connection with non-stationary parts or the linking of mobile machines to the central control network. Moreover, they are also used to connect and operate the machines remotely, deputed in dangerous areas (Frotzscher et al., 2014). In recent times “smart industrial wireless machines” are also widely in use. Along with mechanical parts, it also contains a power source, sensors, communication protocols & programs, memory, and a processor to integrate these parts (Kreibich et al., 2014).

To sum up, for modern industrial automation solutions, these machines required to be smarter, more flexible, and portable, than existing traditional solutions. There are many wireless automation technologies available in the area of industrial automation. It is required to select any solution according to device requirements and description i.e. range, signal strength, and power consumption to avoid later issues.

2.2 Wireless communication protocols

Wireless communication technologies are taking a major part in the development of advanced industrial and process automation systems. Data can be transmitted from sensors to control systems or to any database for further operations by using these protocols.

Various wireless communication protocols are available with having their pros and cons. It is always critical to choose a single protocol that matches the requirements. For instance, by using the Cellular networks 3/4/5Gs, a large number of phones can be connected but it is not possible to use this for real-time processes. Likewise, well-known solutions such as WirelessHART (IEC62591) and ISA100.11a (IEC62743) follows IEEE802.15.4 standard requires a specialized gateway to get internet access (Goursaud & Gorce, 2015).

Some Short-Range Wireless Communication Protocols are studied and analyzed here across the requirements of the machine. Protocols such as Wi-Fi, Bluetooth, ZigBee, and Z-Wave are the most common ones for short-range communication.

2.2.1 Bluetooth (IEEE 802.15.1 standard)

Bluetooth comes under IEEE 802.15.1 standard and initially, it was implemented by Ericsson. It's based on the wireless radio system. It is used to communicate with other Bluetooth-enabled devices based on a client-server architecture where the client starts the connection while the receiver of the connection is called the server (Singh et al., 2011). Bluetooth is used for short-range communication. It utilizes personal network applications, generally called Wireless Personal Area Network (WPAN). Bluetooth connectivity topologies can be categories as "piconet" and "scatternet". While building a network "piconet" term is used for the one which forms WPAN while interconnected networks of these piconets are called scatternet. The devices discovered by the "piconet" comes under the slave category which can only perform point-to-point communication (P2P). In contrast, a master can communicate in both ways either point-to-point or point-to-multipoint (Naidu & Kumar, 2019).

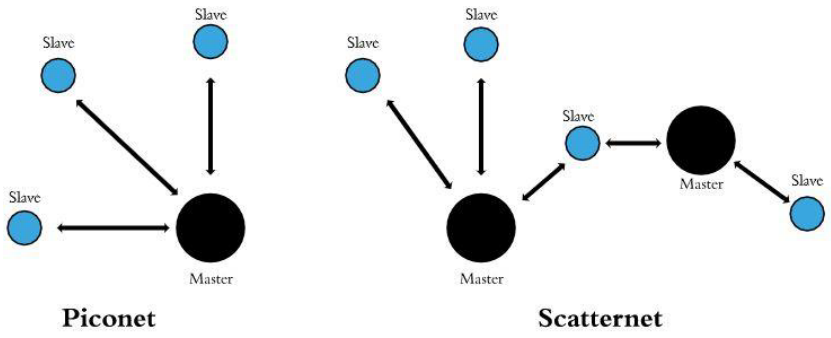


Figure 2. Piconet and Scatternet

Bluetooth low energy (BLE) also called Bluetooth smart is ultralow-power, wireless personal networking technology that is used by modern wireless devices. Formerly called Wibree, it was developed at Nokia but later it becomes part of Bluetooth 4.0 specifications. It consumes very low energy as compared to traditional Bluetooth with the same communication distance capabilities. It is utilized for many IoT applications due to its low bandwidth and low latency capabilities (Salman & Jain, 2017). BLE operates from 2400 to 2483.5 MHz with data speed up to 1 Mbps, and line of sight (LoS) range for 10 m. By

having all these features, Bluetooth is considered a very handy solution to utilise with home or industrial IoT devices (Naidu & Kumar, 2019).

2.2.2 Cellular Communication

The cellular protocol consists of Low Power Wide Area Network (LPWAN) standards. Besides, it can also utilize GSM/3G/4G communication competencies to achieve rapid internet access and connectivity. Cellular technology is very good for applications requiring high throughput data over the longer distance. However, its high power consumption makes it less beneficial for small battery-powered IoT and sensor-based devices (Alsarawi et al., 2017).

Cellular networks attain incredibly good availability and latency factors. With the introduction of fifth-generation (5G) technology, cellular networks are considered important concerning industrial automation. By using multi-beamforming technology which helps to reduce co-channel interference, improved link-quality, and reliability, 5G networks seem to be a major part of industrial IoT (Cheng et al., 2018). However, the 5G system is under development and the standards for IoT are still evolving yet.

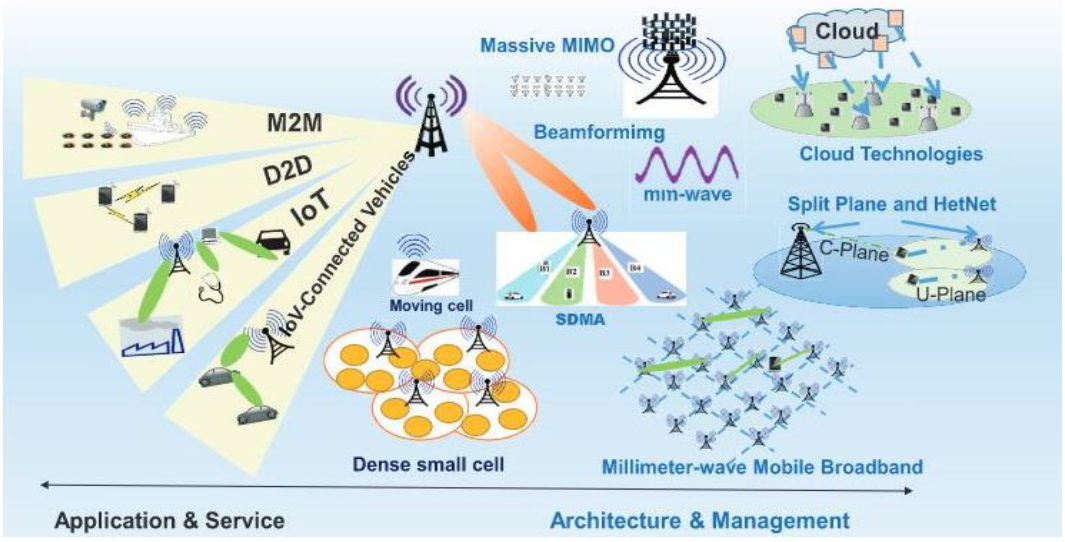


Figure 3. 5G Network Schematic Diagram (Agiwal et al., 2016)

2.2.3 LoRa

LoRa is a newly developed protocol, mostly used for, line of sight, and long distances communication such as kilometers, with great signal strength and high sensitivity. These characteristics also make it considerable for short-range communication in the industry where high noise and radio-sensitivity exist.

LoRa, produced by Semtech, works with spread spectrum technology which makes it very robust to external narrow-band signals and interference. Moreover, it uses the ISM band with different regional standards, i.e. for US 915MHz and in Europe 868 MHz, which means it is independent of crowded 2.4 GHz range which is utilized by common protocols such as Bluetooth, Wi-Fi, and ZigBee (Tessaro et al., 2018).

The constraint of using LoRa in the industry for short-range communication is related to duty cycle limitations that result in low throughput. By using the different combinations of parameter it is possible to obtain 21875bps, which inadequate its use to non-critical, low data industrial applications (Tessaro et al., 2018) but less efficient where high throughput is required.

2.2.4 Wi-Fi (IEEE standard 802.11n/a/b/g)

Wi-Fi is the abbreviation of Wireless Fidelity that follows IEEE standard 802.11n/a/b/g for Wireless Local Area Network (WLAN). It is mostly utilized wireless communication protocol where the user is connected to an Access Point (AP) or in Adhoc mode having access to internet browsing and cloud services provided by the network vendor or administrator at network broadband speed (Naidu & Kumar, 2019). It can operate on both 2.4 GHz and 5 GHz bands by using IEEE 802.11a and IEEE 802.11n standards respectively. Both stated bands are license-free worldwide which makes Wi-Fi suitable for different applications and solutions. In Adhoc mode, stations communicate to each other in a peer-to-peer manner also known as Wi-Fi direct. While in Access point mode, networks have an access point where all the client devices are connected to it, which is known as infrastructure mode.

Infrastructure mode consists of an access point (router) and wireless stations (client). Collectively this setup is called BSS (Base Service Set). Wireless stations accomplish connection to the internet through their associated AP, which contains own Service Set ID (SSID) for identification. These several AP can be connected through a wired distribution system, which results in an extended connected network stated as an Extended Service Set (ESS) (Naidu & Kumar, 2019; Rahman, 2015). The basic infrastructure is shown in figure4.

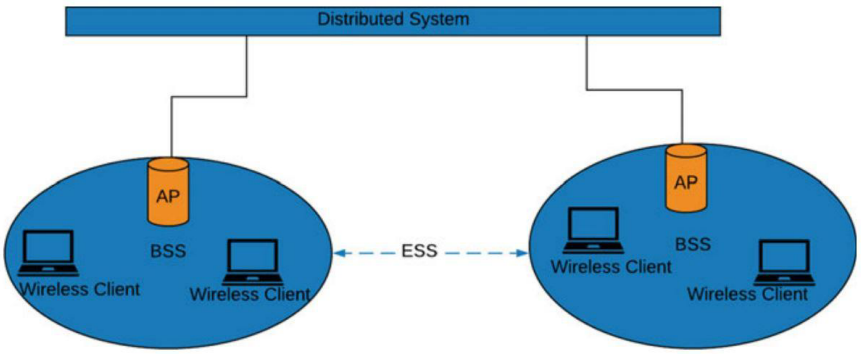


Figure 4. Wi-Fi infrastructure (Naidu & Kumar, 2019)

Adhoc Mode also called Wi-Fi Direct, allows devices to communicate with each other without establishing a connection to any AP. Wi-Fi Direct mode allows devices to search for each other and create a peer-to-peer group where one node that is publishing its information for others, act as AP. It's also called AP-soft generally (Khan et al., 2017). The following figure 5 shows both Wi-Fi connection modes.

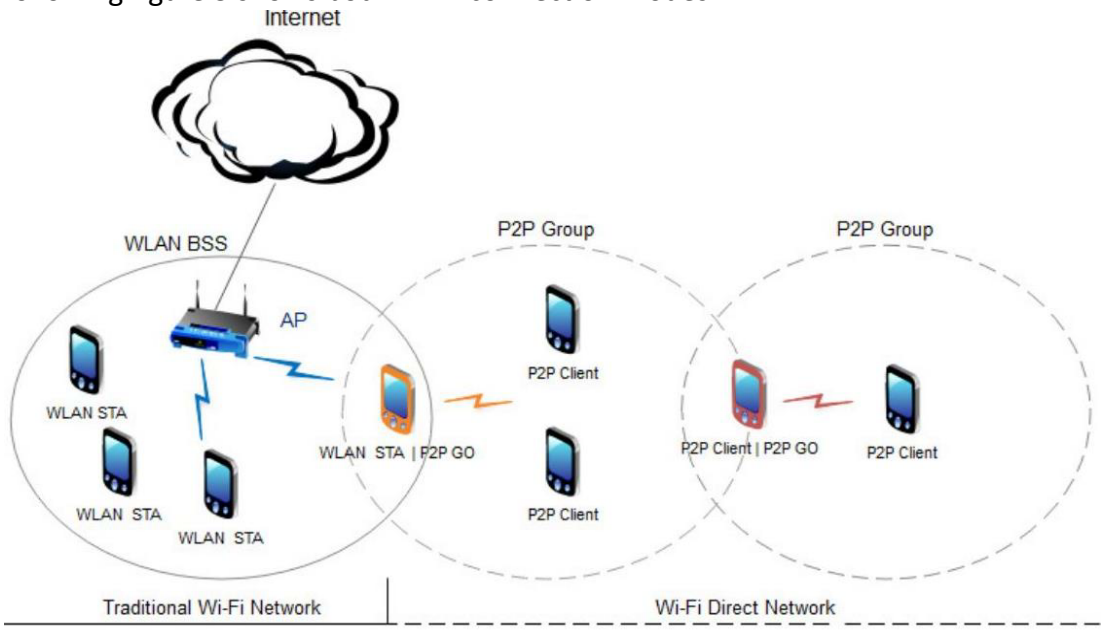


Figure 5. Wi-Fi and Wi-Fi Direct Network Architectures (Khan et al., 2017)

Wi-Fi protocol allows selecting different levels of security from “open network” to the most secure one stated as Wi-Fi Protected Access 2 (WPA2). Wi-Fi can achieve up to 100m range for outdoor communication while around 35m for indoor communication due to obstacles. It can achieve a very high data rate, in comparison to other available solutions that is up to 11–867 Mbps. On the other hand, this high data rate results in higher power consumption (Rahman, 2015).

2.2.5 ZigBee (802.15.4 standard)

ZigBee is a low-cost, low-power, and low-data-rate protocol for wireless personal area networks. It follows the IEEE 802.15.4 standard, maintained by the ZigBee Alliance. While working with the Media-Access-Control layer (MAC) of the network, it can function on different frequencies and data rate ranges like 868 MHz, 915 MHz, and 2.4 GHz with 20 Kbps, 40 Kbps 250 Kbps respectively with the dynamic routing protocol. The features like low-data rates and low-power consumption make it more suitable for short-range and personal network (home automation) applications (Al-sarawi et al., 2017).

It contains range up to 10-30 m normally but can be increased by using star, mesh, and cluster tree network topologies. When forming a network, ZigBee devices are divided into two categories, a router or coordinator or Full Function Device (FFD), and a Reduced Function Device (RFD) also called an end device. RFD can just be used as an end device while in contrast, FFD can be used as a coordinator or router and as an end device as well (Naidu & Kumar, 2019).

In conclusion, RFD or end devices are actuators or sensor which are connected to a single coordinator. Whereas the coordinator can initiate the networks as well as can add more coordinators or router or end devices to the network. These routers, added by the FFD are limited to perform routing operations without initiating a new network. As shown in figure 6, in star network there is a single FFD with multiple end devices/RFD, the tree network contains one coordinator with three independently connected routers called nodes with their end devices. In contrast, mesh connection contains multiple routers that are interconnected to each other as well. Each node is capable to communicate with

other nodes having at least two pathways. This results in forming a network that can handle more end devices in a broader range.

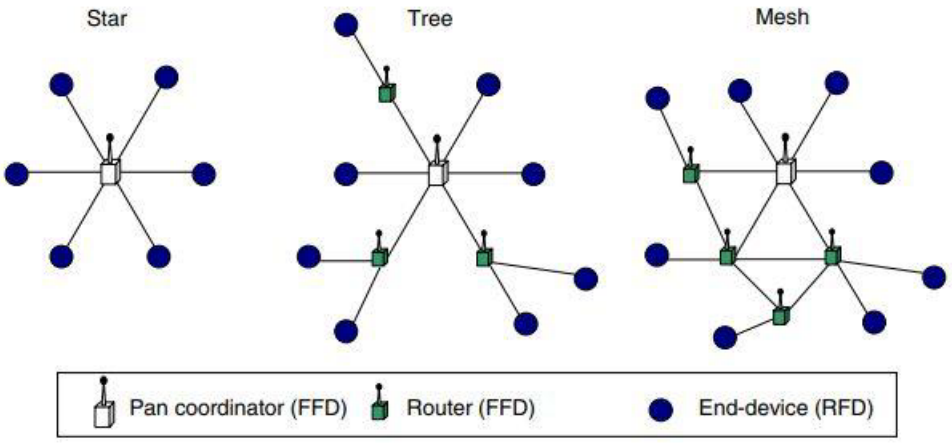


Figure 6. Network Topologies in ZigBee (Baronti et al., 2007)

2.2.6 Z-Wave

Z-wave is an IoT protocol used for home automation and small-scale industrial applications, operates on two different standards for the US and Europe, 908.42 MHz, and 868.42 MHz respectively. It was initially designed by Zensys, than later Sigma design took control of it in 2008 (Badenhop et al., 2017). Z-Wave follows a proprietary standards protocol, with the controller-slave based framework. In each Z-Wave network, there is always a controller having all the routing details of connected salve devices, while slave devices follow the instruction received from it. Single network support is limited to a maximum of 232 devices or slaves. The advantage Z-Wave contains is to work on radio frequency (RF), which results in avoiding intervention with other prominent protocols using 2.4 GHz bandwidth such as Wi-Fi and Bluetooth.

2.2.7 6LoWPAN

6LoWPAN stands for “IPv6 protocol over low-power wireless PANs” is developed by Internet Engineering Task Force (ITEF). It is used for short-range communication where its basic architecture consists of three elements called nodes (sensors and actuators), routers, and edge routers or main routers. Nodes, receive instruction, or send sensor data to

routers, which works as a bridge and forward it to the main router (Edge router). All this communication is performed by IPv6 protocols that run over IEEE 802.15.4. The main router is connected to an external network (usually the internet) through IPv4 addresses. This architecture supports both point-to-point and mesh network configuration where each element has its unique IPv6 address (Al-Kashoash & Kemp, 2016).

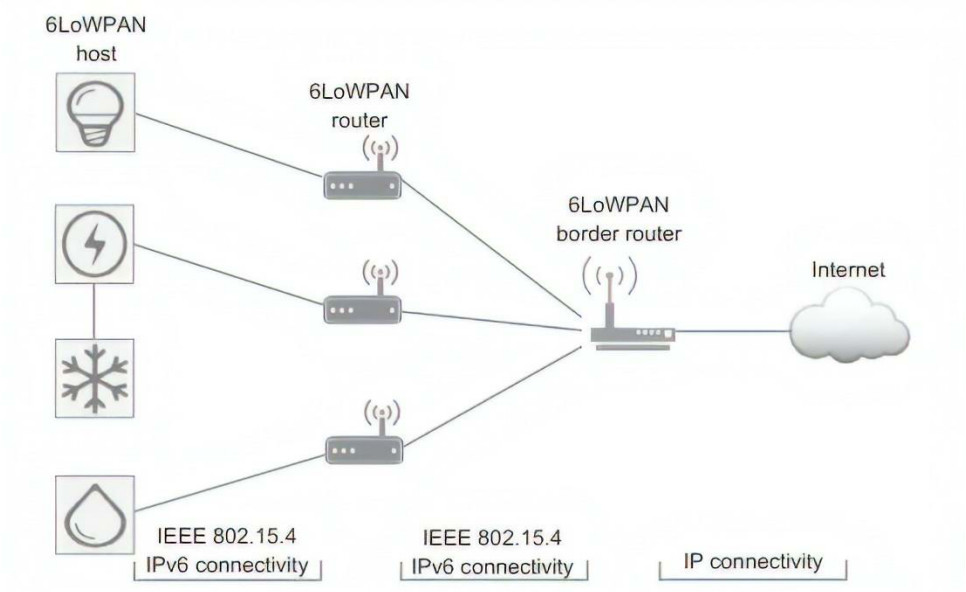


Figure 7. 6LoWPAN Network Architecture (Kushalnagar et al., 2007)

On one hand, low power consumption while using RFC4944 standard makes it a good choice to be used with IoT and small sensor-based devices, on the other hand, a compulsory requirement of internet connection limit its use at an instant.

2.3 Comparison of wireless communication protocols

There are different parameters to compare different wireless communication solutions i.e. range, speed, cost, installation complexity, latency, network infrastructure, etc. The absolute communication protocol or solution is always decided according to project requirements and descriptions. From the different solutions stated in section 2.2, some of these have certain limitations when it comes to using them for short-range communication in the industry with sensor-based machines. For example, Cellular-based solution

requires addition connection and they consume much more power compared to other solution (Al-sarawi et al., 2017). Whereas LoRa attains low data throughput due to some limitations of duty cycles (Tessaro et al., 2018). Although 6LoWPAN is a very handy solution for short-range communication but the requirement of internet connection, make it inadequate where internet connection is not available (Al-Kashoash & Kemp, 2016).

To conclude, the remaining solutions Bluetooth, Wi-Fi, ZigBee, and Z-Wave are compared in detail with their pros and cons to developing a clear understanding.

Z-Wave runs on a different frequency band than these other solutions which all works on 2.4 GHz bands, resulting in less interference and clear communication. Its range can also be increased according to requirements by using mesh structure. However, it has a major drawback, i.e. the Z-Wave devices are designed by following the country-specific radiofrequency guidelines which result in limiting their usability to just that country (Badenhop et al., 2017).

ZigBee is a very useful communication protocol for home automation and lightweight industrial applications with the ability to works on different available frequencies. But as stated earlier in section 2.2.5, it is designed for lightweight industrial applications. When it comes to handling a large amount of data with high data throughput, it results in a slow process and latency that is not acceptable for most of the real-time sensor-based machines (Al-sarawi et al., 2017).

Bluetooth is one of the most popular solutions for short-range wireless communication. Bluetooth LE is very efficient at power consumption and can be configured in sleep mode until communication is initialized. Its range is limited to 10m which makes it less efficient for cordless industrial machines. Moreover, not all routers and HUBs are Bluetooth compatible, which results in the necessity of an additional Gateway other than the machine to get internet access in case of data-logging to an external or cloud database (Zachariah et al., 2015).

Wi-Fi is the most used wireless communication protocol with various advantages to other solutions in terms of device availability, range, data throughput, and installation

procedure. Moreover, mostly devices follow the same country-specific guidelines for radiofrequency which makes it a generalized solution. Although Wi-Fi protocol has some drawbacks such as high power consumption compare to other solutions (Rahman, 2015) but it can be compensated when the wireless machine contains a powerful battery. Moreover, it utilized the crowded 2.4 GHz band which results in interference and latency but this can be handled through better system architecture as well.

To summaries, from different studied wireless protocols, Wi-Fi is the best option to be utilized with the specific requirements of high data-throughput. It is a generalized solution, in terms of advance digitalized industry with communication capabilities to local and cloud databases without using any additional gateway. Figure 8, shows an overall comparison of different above compared wireless solutions.

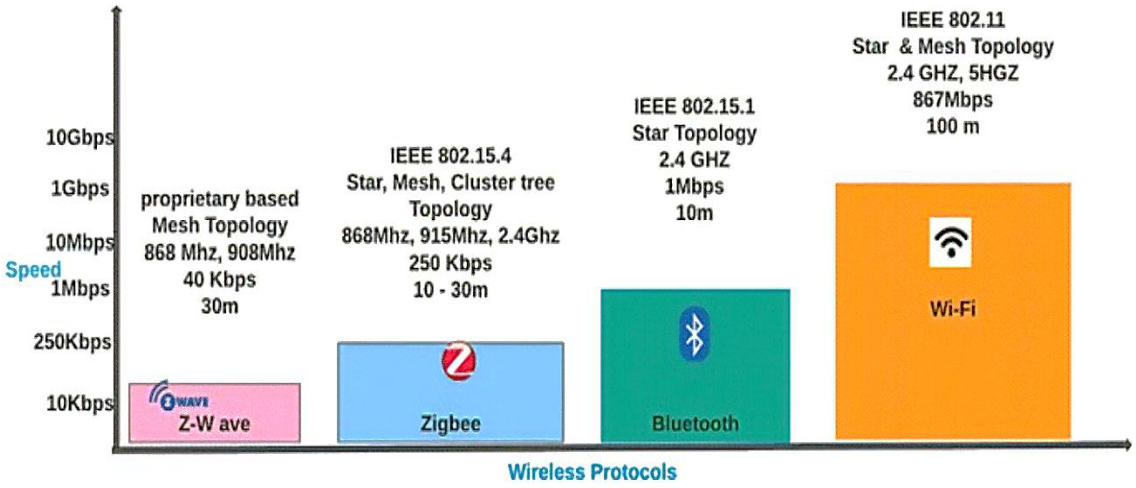


Figure 8. Wireless Protocols Comparison (Bluetooth, Wi-Fi, ZigBee, and Z-Wave)(Naidu & Kumar, 2019)

2.4 Embedded systems & testing

The Embedded system consists of both hardware (i.e. sensor, actuators, power source, and Microcontroller) and software (device firmware) and designed to achieve stated goals to meet requirements from the user. Embedded devices contain complex system architecture and they are made to perform specified tasks. While performing those tasks,

defects can result in major damages such as life-threatening situations, delays in the process, and deficiencies in the production quality (Ebert & Jones, 2009). The comprehensive testing process for embedded systems is very important before their production and deployment. Generally, embedded system testing contains the verification and validation phase for both software and hardware. According to the IEEE Standard Glossary of Software Engineering Terminology, verification is stated as "*The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase*" while validation as "*The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements*" (Paul & Jeff, 2008: 12).

2.5 Types of embedded system testing

Testing real-time embedded systems is always difficult because of their complexity, hardware interaction issues, and complex behavior. Generally, the embedded system requires a series of testing such as which-box (structural) and black-box (functional) testing. Depending upon requirements, module and integration testing is also part of this testing sequence. In general, function and integration testing are more important than structural and module testing (Tsai et al., 2005).

2.5.1 Types of testing approaches

- **Test-driven development (TDD)**

Test-driven development is an incremental way of writing the code where writing a test case is a prerequisite for starting writing the actual code. Initially, after writing a failed test, the actual code is written and tested until it passes the pre-written test (Grenning, 2011: 30). In other words, the actual function is written after writing the test code for it, which reduces the errors and saves time. According to Kent Beck's book, there are five general and simple steps to perform TDD

I. Creating a small test case

II. Run the test and waiting for to be failed
III. Write/modify the code as per functional requirements
IV. Run the test and waiting for it to pass
V. Refactoring the code for improvement and remove duplications (Ashbacher, 2002)

- **Behavior-driven development (BDD)**

Behavior-driven development is a specification based driven development where the basic principle is to follow specifications rather than a test procedure. It works opposite to TDD. BDD follows this described procedural form:

- Given: some preconditions are introduced at this stage
- When: waiting for some response from any other scenario
- Then: Any pre-described event which would occur after fulfilling both initial conditions (Grenning, 2011: 316)

2.5.2 Levels of testing

- **Unit testing**

Testing of an individual unit of the software is called unit testing which leads to the validation and verification of its functionality. Unit testing is premeditated to find the bugs into the independent small units of the software. A single unit can be a function, a class/object, or a component, from a whole library that is tested to validate its functionality according to its specifications. Test codes written by developers or tester are used for this initial level of testing. The process includes the creation of stub and drivers. Drivers used to call the components under test while stub works as the components to be tested (Myers et al., 2011). Unit testing also involves different stages from its planning, test designing, and implementation. After that, the next step is the validation of the recorded results across the expected output of the unit under test. The results are analyzed to determine whether the test is passed or fail.

- **Integration testing**

Integration testing is performed to verify the functional interaction and intercommunication between the different distinct units of the codes. By using these smaller units, a more complex and broader component is formed to test and validate their reciprocal action. There are different approaches to perform integration testing including two more generous approaches bottom-up and top-down (Myers et al., 2011).

The top-down integration approach is expected to be started from the top higher level module followed by its integration with the next most important module and continues to do so. Integration tests are performed on all the stages while adding already tested small functional units to the modules. The advantages include the requirements of fewer drivers, Also, having sample software right from the start for the testing. The disadvantage is that basic functionality is tested at the end-stage (Myers et al., 2011).

In contrast, in the bottom-up integration approach, the integration starts with the lower small units and the system is under development until the last unit is added to the system. In this approach, the lower functions are more frequently tested compared to the top-level functions but the whole system is not present until the addition of the last module. It becomes more time-efficient because testing starts as the basic functions, and they are tested and simultaneously integrated (Myers et al., 2011).

- **System testing**

System testing is related to the testing of the entire system across its known specifications, involving functional and behavior base testing along with system performance tests concerning resource utilization and response time (Briand & Labiche, 2002). In general, system testing is the next stage after completion of unit testing and integration testing. The system is formed after the integration of different subsystems having their own smaller functional units. Generally, system test requires more resources for the testing such as laboratory equipment and testing software. System testing also takes a longer time than the unit and integration testing. Some general types of system tests are discussed below (Freeman, 2002).

- Functional testing
- Performance testing
- Stress testing
- Configuration testing
- Security testing
- Recovery testing

- **Acceptance testing**

Acceptance testing is an advanced form of system testing where the features and functionalities of the device are verified across the user requirements for it. Sometimes it is also called user acceptance testing. It commits a suite of tests on the outright system (Miller & Collins, 2001). Acceptance tests perform the following three types of actions:

- Verify the completion of user/device functional requirements and measure how well they are satisfied
- Find the bugs missed in Unit and integration testing
- Provide an indication of what is “done” regarding system

Boundary values testing is also a type of acceptance testing where the received response is validated across some known boundary limits.

2.6 Automated Testing Framework for Embedded systems

According to Lee (2000: 20) “*A framework is a set of constraints on components and their interaction, and a set of benefits that derive from those constraints. A framework defines a model of computation, which governs the interaction of components*.” In other words, the framework performs the interaction between the different components to get some useful output. Embedded system testing frameworks can be explained as the framework which is used to test interaction and relation between the different components of the system under test and produce a useful output in terms of results.

There are various Test Automation Frameworks such as Robot Framework, Cucumber, Testim, and Gauge with their pros and cons. Due to limitations of thesis work only Robot Framework is discussed here, which is also chosen for this project work. Robot Framework is a preferred choice because of being open-source, cross-platform, easy keyword-driven testing (KDT) approach, and the support for different languages such as Python, Jython (Java) and IronPython (.NET). Moreover, libraries like "Selenium Gold" and "Pabot" aids in parallel execution of tests, which is very helpful in embedded system testing.

2.6.1 Robot Framework automated testing

Robot Framework can be defined as: "*Robot Framework is a generic open-source automation framework for acceptance testing, acceptance test-driven development (ATDD), and robotic process automation (RPA)"* (Robot Framework, 2020). Robot framework is a keyword-driven, tubular syntax-based testing solution with the competency to increase usability by introducing new custom based keywords (functions) and libraries. Libraries can be written in Java and python. New higher-level keywords can be written by utilizing pre-existing keywords from different available libraries. Robot Framework also provides luxury to write whole new libraries from scratch (Robot Framework, 2020).

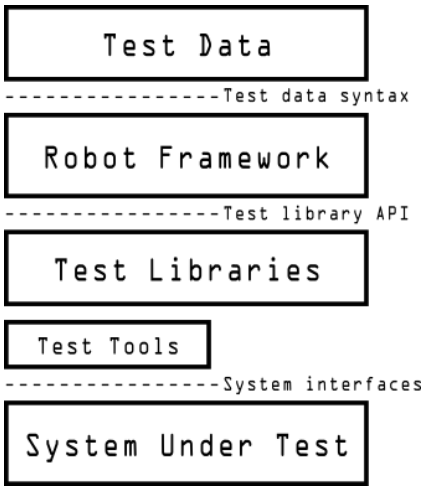


Figure 9. The architecture of Robot Framework (Robot Framework, 2020)

In other words, Robot Framework is a platform that provides a whole testing framework to perform automated testing by interacting with the application programmer interface

(API) which implements the given test data syntax to verify the received response from the system under test. Figure 9 shows the basic architecture of the Robot Framework where after receiving the test data, it communicates directly to the system under test by utilizing the provided test libraries. Usually, test libraries have direct interaction with the system under test by using given high-level keywords but in some cases, some libraries also require some prerequisite additional drivers to establish the connection such as drivers for different databases and web browsers.

Robot Framework libraries are categories into two types which are standard and external libraries. Each of these libraries is used to achieve unique testing goals. The libraries which are included in the Robot Framework are called standard libraries such as Built-in, Operating system libraries, strings, and Remote library. Contrarily the libraries created by the users to achieve the specified unique requirements to perform certain tests are called external libraries. Some of the famous external libraries are Selenium Library (Web testing), Apium, and Android (Device testing), MQTT-library, JSON-validator, and Database library.

Robot Framework also generates results and logging files after conducting tests, which is a key feature of the Robot Framework, regarding the development of a fully automated testing platform. During the test time, test feedback is provided by the terminal, then after the completion of test HTML (Hypertext Markup Language) and XML (eXtensible Markup Language) files are generated to analyze the behavior of the test. These files can also be saved for later use (Robot Framework, 2020). Following is the example of a Robot Framework test file utilizing the MQTT library that is an external library for Robot Framework. It is open-source software under Apache License 2.0, which provides the Keywords for performing the various operations involving MQTT broker (robotframework-mqttlibrary, 2019).

Robot Framework test sequence file follows a specialized writing pattern and it is divided into four different sections. These sections are called tables and defined as Settings, Variables, Test cases, and Keywords.

- **Settings**

  In this section different libraries are defined which would be utilized during running the test sequence

- **Variables**

  Variables are defined in this section to make test execution more generalized and simpler. Variables are not only used in test cases with defined value but they are also used as an argument while defining new high-level keywords. Robot Framework has its particular types of variables such as scalars, lists, or dictionaries using syntax ${SCALAR}, @{LIST}, and &{DICT}, respectively (Robot Framework, 2020).

- **Test cases**

  Test cases section includes the unique names given to the tests which consist of different user-defined high-level keywords or keywords from the libraries.

- **Keywords**

  This section is used to create new high-level keywords by utilizing existing keywords. The syntax of creating user-defined keywords is close to case syntax but also, they require some arguments along with the keywords from libraries (Robot Framework, 2020).

```
*** Settings ***

Documentation   Test-suit for Demonstration
Library        MQTTLibrary

*** Variables ***

${BROKER_URI}      localhost
${BROKER_PORT}     1883
${SUBSCRIBE_TOPIC}      test/example
${PUBLISH_TOPIC}        test/example
${expected-payload}    test-msg

*** Test Cases ***
```

```
Connect to broker and Publish

     [Documentation]      Connect to broker and Publish to
     given topic

     Connect_to_broker     ${BROKER_URI}     ${BROKER_PORT}

     Publish_to_Topic       ${PUBLISH_TOPIC}    test-msg

subscribe and validate payload

     [Documentation]     Receive the payload and validate it
     w.r.t time and expected Pay-load

     validate_payload       ${SUBSCRIBE_TOPIC}    0    ${ex-
     pected-payload}   5 sec

     [Teardown]  Disconnect

*** Keywords ***

Connect_to_broker
    [Arguments]  ${BROKER_URI}  ${BROKER_PORT}

     [Documentation]      User-defined keyword to make con-
     nection with the broker

    Connect     ${BROKER_URI}   ${BROKER_PORT}

Publish_to_Topic
    [Arguments]  ${PUBLISH_TOPIC}  ${TEST_PAYLOAD}

    [Documentation]     User-defined keyword to Publish Pay-
    load to topic
    Publish     ${PUBLISH_TOPIC}   ${TEST_PAYLOAD}

validate_payload
    [Arguments]  ${SUBSCRIBE_TOPIC}      ${qos}       ${ex-
    pected-messages}      ${timeout}
    [Documentation]     User-defined keyword to Receive pay-
    load and validate it

Subscribe and validate      ${SUBSCRIBE_TOPIC}    0    ${ex-
pected-payload}   ${timeout}
```

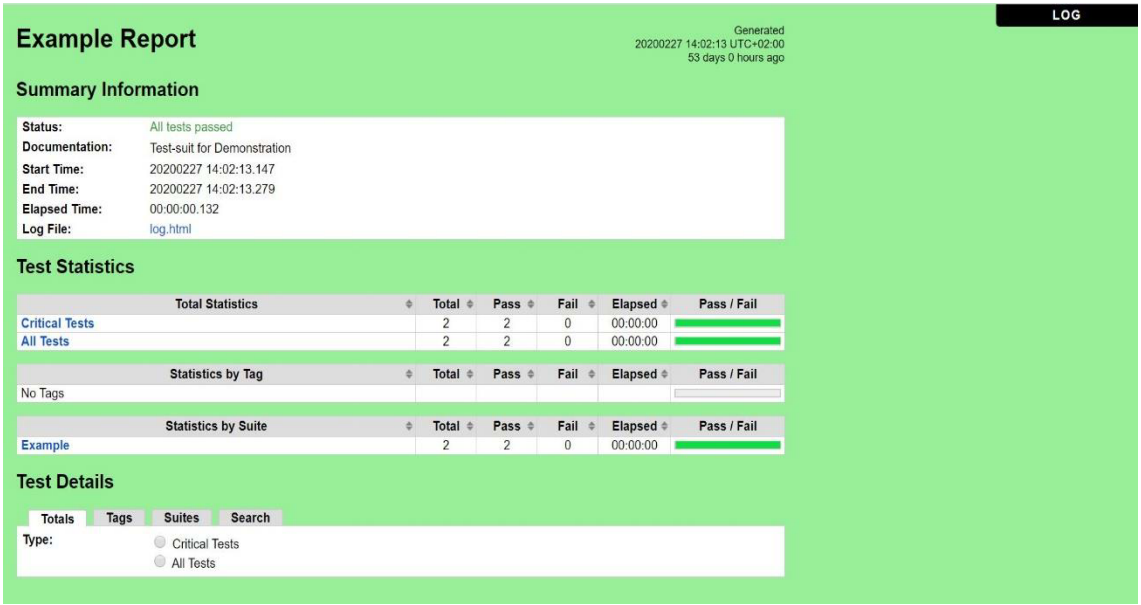
Algorithm 1. Test Sequence File, for example.robot, RF test

The MQTTLibrary utilized in this test sequence is implemented as a python class that containing functions to drive the keywords. After running a test sequence in the Robot framework, it refers to the mentioned source files or the libraries from the settings table

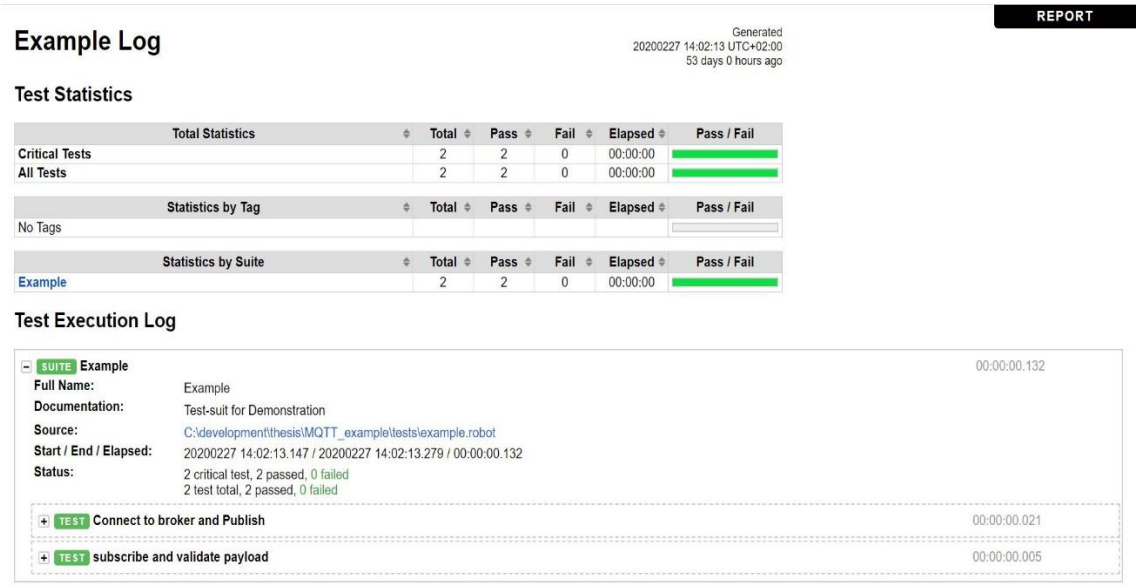
and called required python function against a given keyword. Robot Framework also supports Java libraries and function but that is out of scope to this thesis work. This test sequence can be run by fulfilling some pre-requirements such as the installation of Python and Robot Framework along with the external library MQTTLibrary. To run the test sequence from Pycharm following command can be run from the project directory.

"robot tests/example.robot"

Where "robot" indicates that the python module is called with Robot Framework where "example.robot" is the name of the test sequence which is in the "tests" directory. The test execution results are shown in the terminal, in terms of fail or pass and complete test execution logs file and reports are created which was mentioned earlier. These sample logs and result files can be seen in the pictures below.



Picture 1. The Screenshot of the Test Report File Generated by RF



Picture 2. The Screenshot of the Test Log File Generated by RF

From the different available test suit writing styles, the above-mentioned style containing python keywords (function) and libraries is implemented from further development of the automated testing platform for the device under test.

3 Device Under Test (Case Study)

3.1 Introduction to company case study

This master's thesis is done for Mirka Oy[1] that is a family-owned Finnish company and a part of the KWH group. Mirka is a world leader when it comes to abrasives technology and innovation, with being the only company that develops and produces abrasives, tools, and polishing compounds under the same roof. Mirka aims to provide complete sanding solutions and optimize them (Mirka Oy, 2020b). Mirka's sanding and polishing tools portfolio consists of electric and pneumatic sanders, polishing machines, dust extractors, equipment, and tools for sanding walls and ceilings" (Mirka Oy, 2020a).

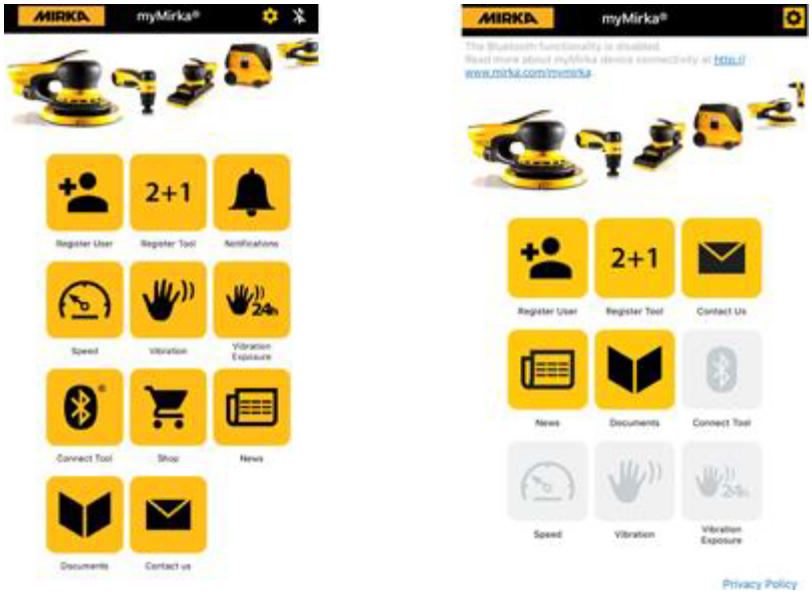
Picture 3. Mirka tools

3.2 Basic Working of tools

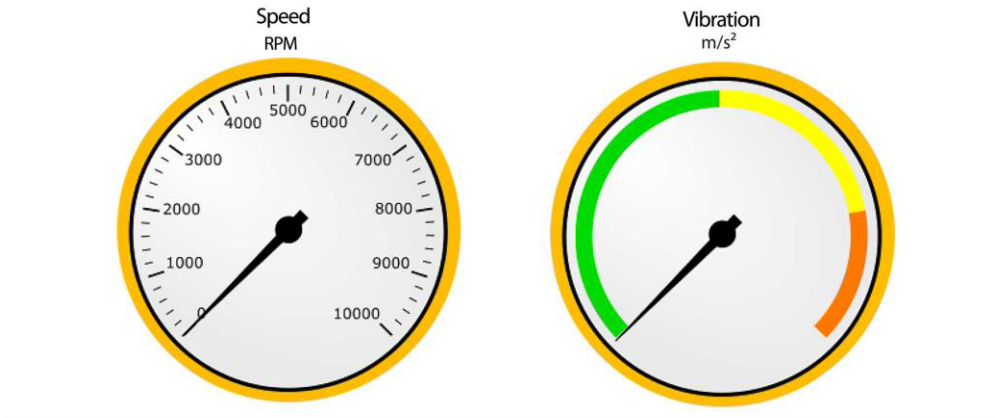
Mirka's cordless sanders and cordless polishers enriched with advanced features like RPM range management (locking or limiting speed), Interval setting / Auto-stop function (setting of runtime), and Vibration measurement (current and Daily exposure for follow-

[1] The author of this thesis is part of an ongoing development project at Mirka Oy. The necessity of an automated testing platform was realized to verify and validated the different functionalities of the device under development. These tests are planned to be performed during the product development and firmware up-gradation phase to validate device functionality and updated features. The author conducted research for the suitable testing framework, against project requirements, having considerations about the further scope for the company and to develop it as a fully functional automated testing platform.

up) (Mirka Oy, 2020c). Mirka provides an app named “myMirka” for the Bluetooth capable machine for connectivity and optimization. Within the app, speed monitoring along with vibration measurement feature is available. Vibration is measured according to ISO 5349-1:2001(E) standard (Mirka Oy, 2020c).



Picture 4. myMirka App



Picture 5. Real-time Speed and Vibration Monitoring

3.3 Background study of previous working approach

Mirka's cordless tools family utilize Bluetooth protocol for communication to myMirka app. Sensor-based devices using Bluetooth as communication protocols require an additional gateway to access the backend server. myMirka app also works as a gateway to transfer data to the Mirka dashboard for later use and optimization. Figure 10 shows a Bluetooth enabled cordless polisher with multiple connectivity approaches to the backend. It can utilize either myMirka app installed in any mobile device or an additional gateway or to send data or receive numerous operational instructions.

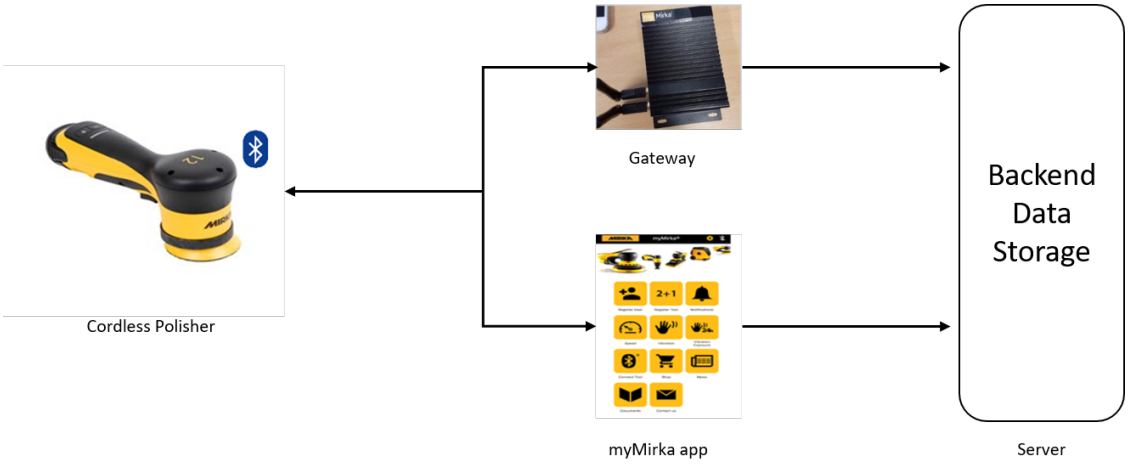


Figure 10. Recent Working Architecture

However, using a mobile app or an additional gateway is a good solution but it doesn't fit for all. In many cases, using a mobile phone is not an option, also there are some restrictions about gateway standards. So it requires a more generalized solution for these kinds of use cases.

3.4 Updated architecture and working approach

Mirka came up with an idea of a new tool with Wi-Fi communication capabilities including several new features and functionalities, which is under development and testing phase. Implantation of Wi-Fi communication protocol would make these devices more

suitable and fitting into the digitalized industry concept. Using Wi-Fi as a communication protocol would offer the following key advantages compared to the previous approach:

- High data throughput compared to Bluetooth
- More than one devices can be attached to a single network
- Support for both wireless internet and local network
- Enhanced connectivity range compared to the previous approach
- More suitable to industry 4.0 concept
- No additional gateway requires to access the backend server

Figure 11 shows the basic working block diagram of updated architecture. An external user interface module is also shown here. It can be installed or accessed by any PC/ mobile device that will help to provide network credentials to the machine.

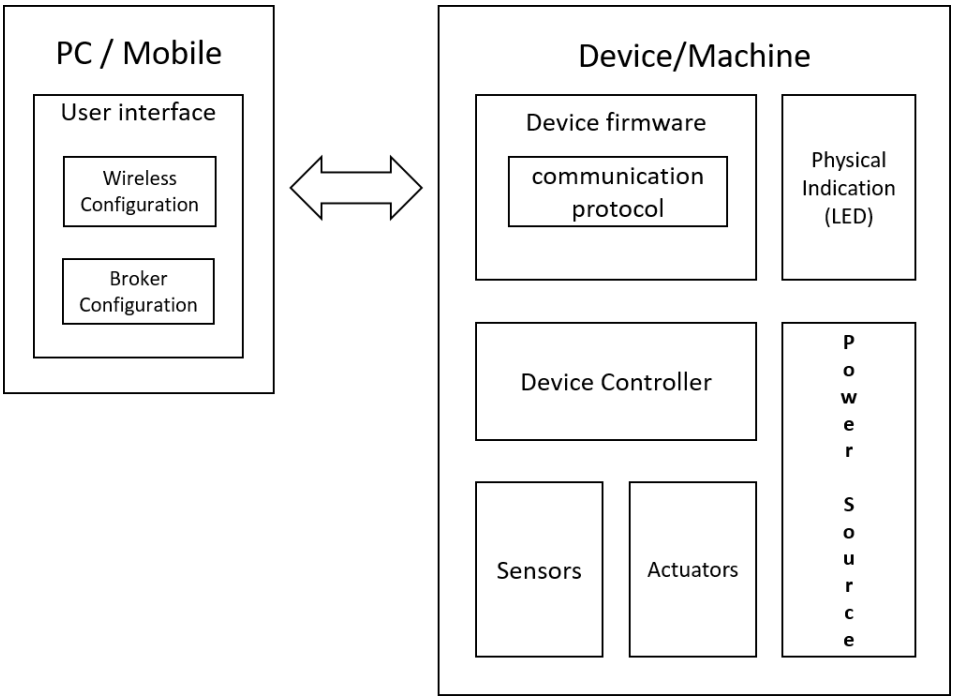


Figure 11. Updated Device Architecture

3.4.1 Configuration of device

According to the basic working principle when the device is switched on for the first time or has been reset, it would turn on in Access Point (AP) mode. By using an external interface through PC, tablet, or mobile device, credentials of the local wireless network are stored into the device. After getting these credentials, the device switches to the station mode and gets access to the internet or local network where it can send data and receive instructions. All these modes are explained in section 2.2.4. Figure 12 shows both stages where initially the device is in access point mode while after getting credential of the wireless network through a user interface, the device is switched to station mode. To update wireless network credentials, a complete user interface is designed. However, it is not included here as it is out of the scope for this thesis work.

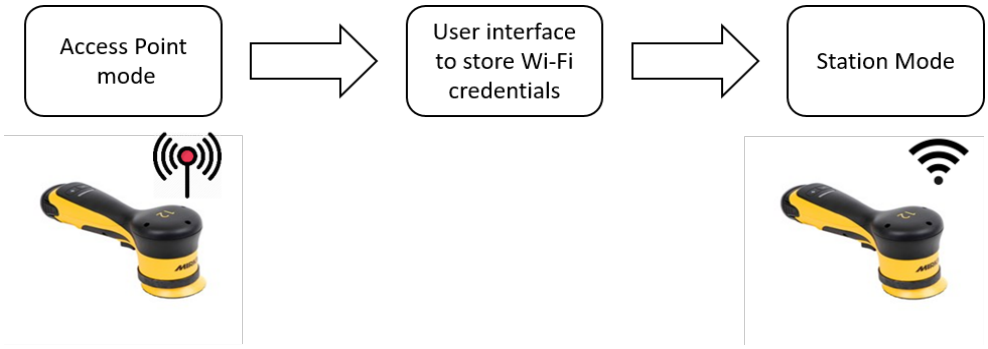


Figure 12. Wi-Fi Modes

3.4.2 Working of device

Once the connection is established to the wireless network, the device starts sending sensor data to the backend database every n^{th} second. Figure 13 shows the working design architecture of the device. As shown, a wireless access point where two different polishers are connected - each working independently. Here the wireless network can be a local network or linked to the internet. Polisher would send data, gathered by various sensors, to MQTT broker that can be monitored and optimized by using any user-defined interface, or can be store to any backend server for later use. it is also possible

to send back some working instructions to the machine by using the same MQTT protocol as the device is capable of two-way communication explained earlier. This instruction can be such as a change in data logging interval or fixing the machine speed to a certain RPM.

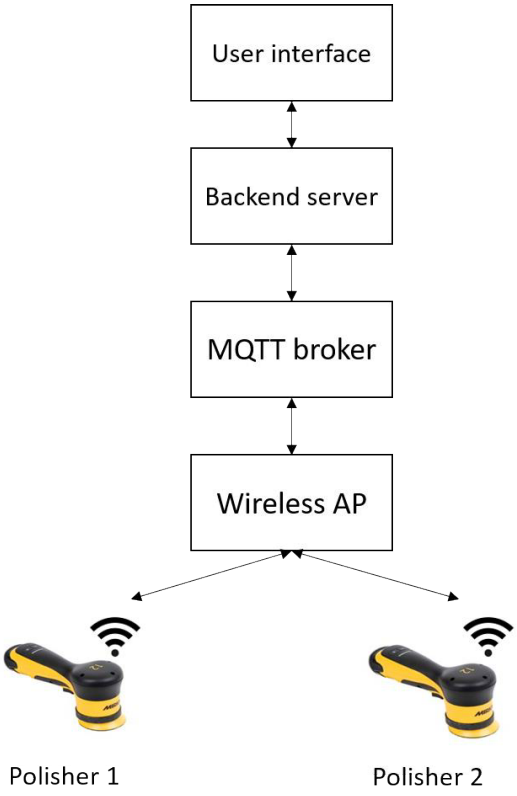


Figure 13. Device Working Architecture

The entire System flow is summarized in figure 14. Before that, some of the key blocks are defined here to get a better overview.

Wireless Access Point (AP)

A wireless access point is a hardware networking device that allows other devices with the Wi-Fi capabilities to connect and share data with it. Here wireless Access point is used to form a connection between the device and network.

MQTT Broker

MQTT broker is a software running on a computer or cloud to handle the MQTT message transmission protocol. Various open-source and paid MQTT brokers are available such as Mosquitto, MQTT-explorer, cloudMQTT, etc. They are local and cloud-based. It is also possible to utilize a self-built broker. Here MQTT broker is used for sending and receiving messages from the device.

Back-end server

The backend server is a combination of a server, an application, and a database. Here backend server is used to store messages received from the device for the optimization or later use.

User interface

The user interface is a device/software designed to facilitate human-computer interaction. Here the graphical user interface is required to get a graphical representation of data, also this interface is used to send instructions toward the device.

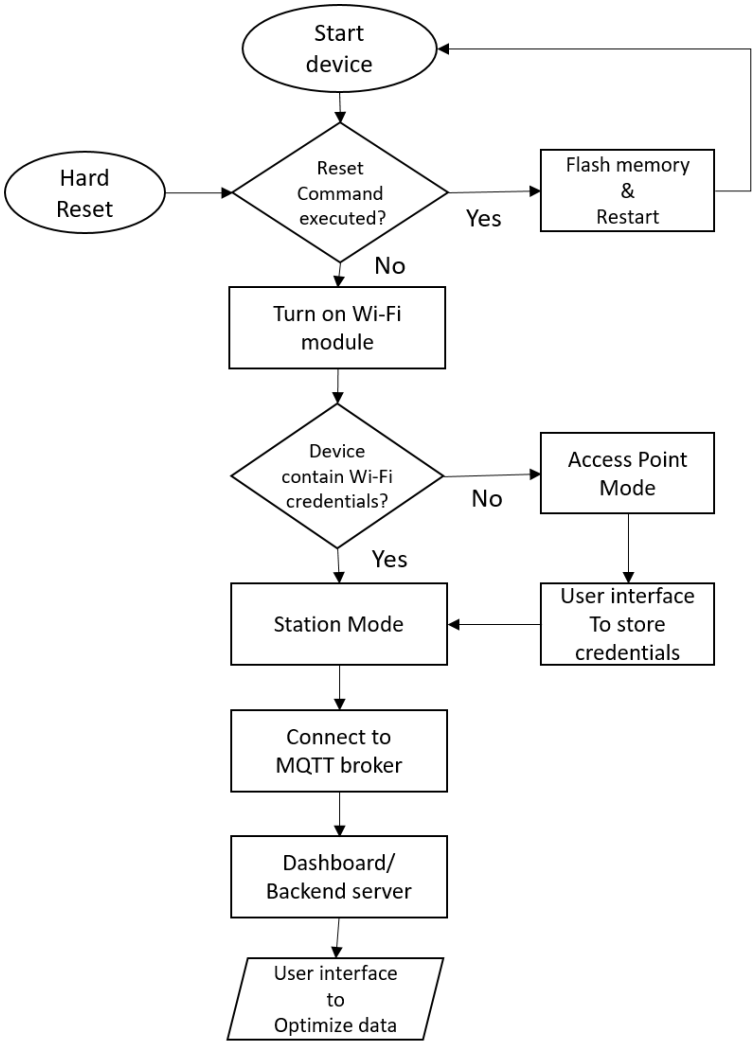


Figure 14. System Workflow Design

4 Testing: Verification and Validation

The testing approach adopted for this project is system testing. More specifically “Acceptance testing” is performed on the device under test which is an advanced form of “system testing” as explained earlier in section 2.5.2. In other words, the features and functionally of the device is tested against its expected behavior. Device functionality is described with details in section 3 of this thesis. By following that, a generalized test sequence is designed that is shown in the figure below.

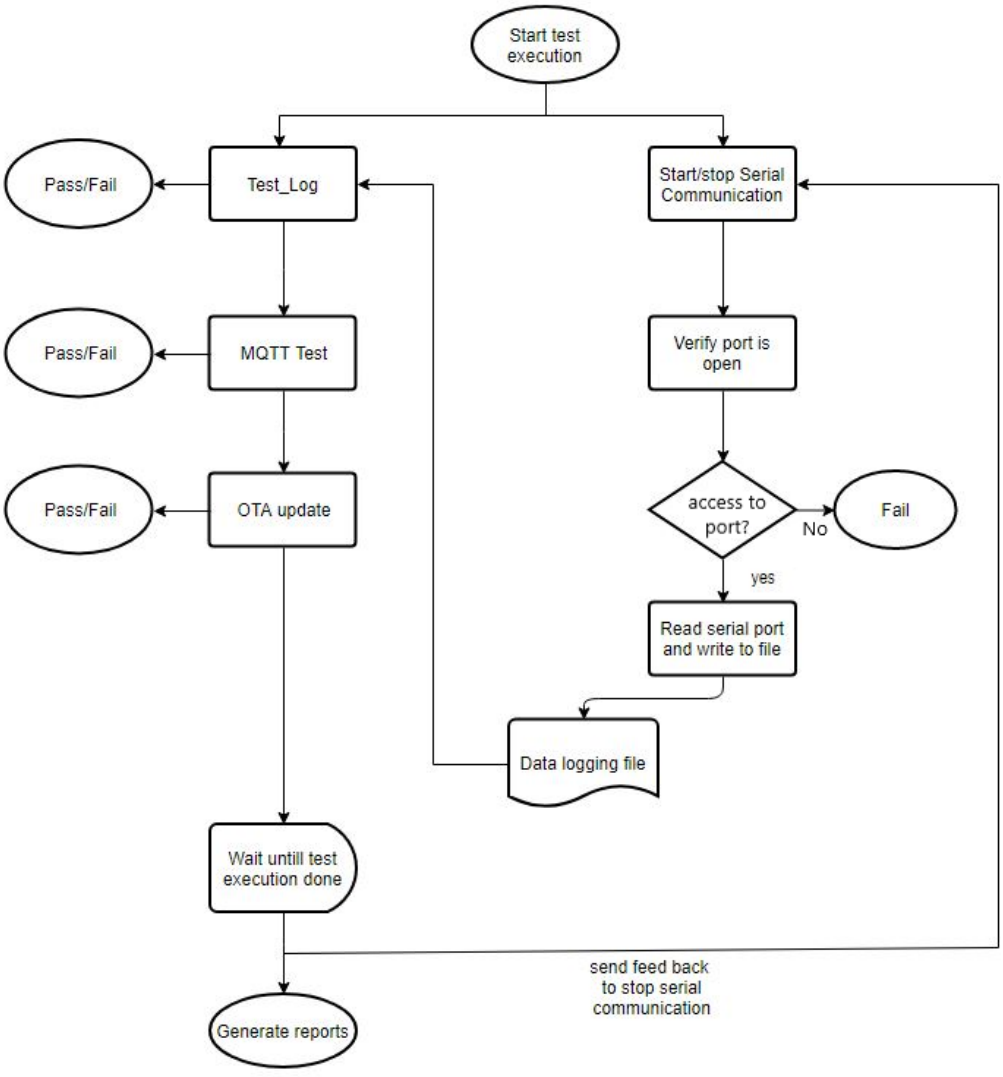


Figure 15. The Flow Chart of a Basic Working Architecture

By following the described test flow these can be some helpful guidelines regarding various test sequences.

Guidelines regarding Expected tests

- Verifications and validation of serial data-logging, by directly reading from the serial port (requires a separate program i.e. python script to read serial port)
- Verify, the connection is established to MQTT-broker
- Verify, MQTT broker receives payload every n^{th} second
- Store the received “Payload” as well for later use
- Validate, parse “JSON-String”
- Publish & Validate, Directives to change data-logging interval and validate the next payload received
- Implement, “OTA update to machine firmware” and validate that firmware is updated successfully
- Compare, Received “Payload” against pre-defined limits from a “text file” in terms of “less than”, “greater than”,” should be equal to”, “true”,” false” etc.
- Validate, Setting the machine to a specific RPM, for a specific period
- Validate, stored “Payload messages” from storage against pre-defined limits
- Close serial connection after completion of the test sequence
- Generates, test results for process verification of the process and later use
- Generate additional reports or files i.e. graph if required

4.1 Test design phase

According to the functional test requirements of the device, a basic test template is designed that is showing test details, expected results, and some prerequisites to perform that test.

Table 1. Test Template Serial Tests

Test #	Test Details	Expected Results	Prerequisites
1	validation_of_serial_logs		
i.	Read the Logs	Find some keywords	Serial port data logging interface is running
ii.	Wait Until Keyword Succeeds	Find keyword: "device name"	Same As above

Table 2. Test Templet for MQTT Tests

Test #	Test Details	Expected Results	Prerequisites
2	MQTT _Tests (Validation of the Data received)		
	Connect_to_broker		
i.	Connect	Connection established	Broker address, port number
	subscribe_and_validate_payload		
i.	Subscribe to topic	Subscribed and receive a message	Connection to broker
ii.	Parse JSON data	Parse different values from JSON string	Getting machine data
iii.	Validate different values across known limits	Values are validated	Predefined limits for these values
iv.	Validate latency	According to the allowed maximum delay	Predefined maximum delay and latency
	publish_configurations_to_timer		
i.	Publish to machine "change data logging time"	Change in data logging interval	Connection to broker
ii.	Validate updated time intervals	According to time limits	updated time interval limits

Table 3. Test Templet for OTA Tests

Test #	Test Details	Expected Results	Prerequisites
3	OTA_update (Validate over the air updates)		
i.	Publish configuration to machine	Device receive payload	Connection to broker
ii.	Verify update initialize	Update start	The connection between broker and backend server
iii.	Wait for device response	Update completed	Access to serial log data
iv.	Validate firmware is upgraded	Firmware version details updated	Connection to the broker to validate it from the payload

4.2 Designing Automated testing system

The basic structure of the testing system consists of these major parts

- a serial communication set up between the device and testing platform
- local network host to support wireless communication between device and testing platform
- database/data-log file to store payload for later use
- test library to conduct all these tests

The architecture of the testing system is shown below in figure 16.

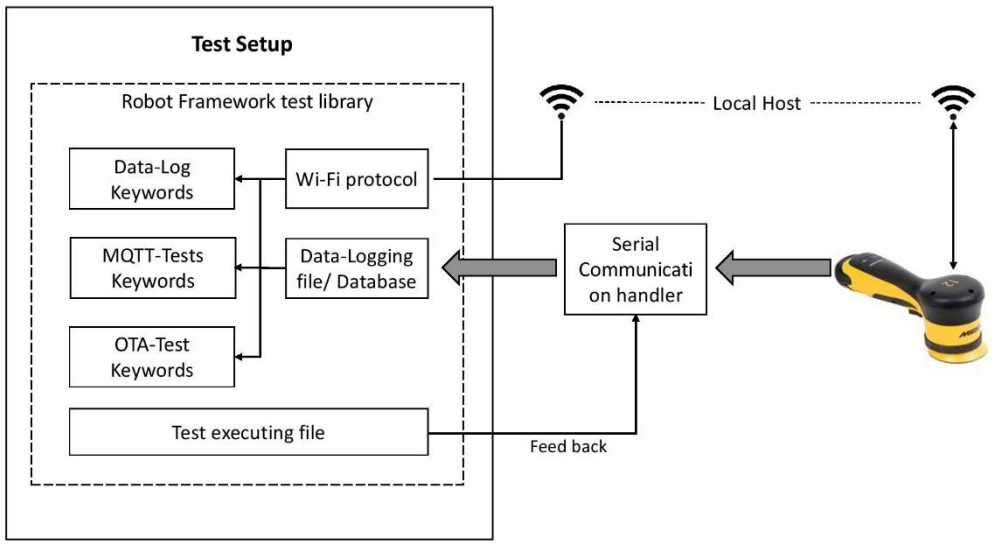


Figure 16. The Architecture of the Test System

This test setup contains three major parts.

- Device under test
- Serial and wireless communication between device and test library
- Test setup containing a test library with additional data logging files and databases

As explained in chapter 3 that device utilizes sensors to collect data and publish it by using the MQTT communication protocol. Also, it receives instruction via MQTT subscribe protocol. Serial communication is required between the test library and device to receive log messages from the device. Test setup contains a test library which includes different keywords driven files, designed according to functional testing requirements. All three parts are explained here in detail to get an insight regarding the system.

4.2.1 Device under test

A basic device architecture is shown in the figure below. As explained earlier, the device contains some sensors and actuators, controlled by the device controller. Moreover, it utilizes serial and wireless communication protocols to interact with the system or to any

backend to receive and send information. As a wireless communication protocol, the device uses the MQTT communication protocol as explained earlier in section 3.

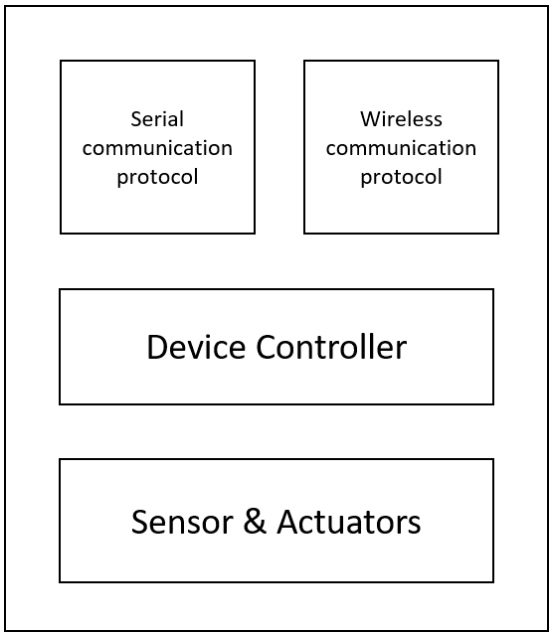


Figure 17. Device Basic Architecture

4.2.2 Design for Serial communication process

Serial communication between the testing setup and the device is a basic requirement of the testing framework. The serial communication module follows these given requirements.

- Open the serial port on which device is connected
- Reading data from the serial port and write it to data-logging file at the same time
- Look forward to the "*stop*" signal from the test set up during the described testing process
- Close the communication, and the serial port, on receiving the stop signal

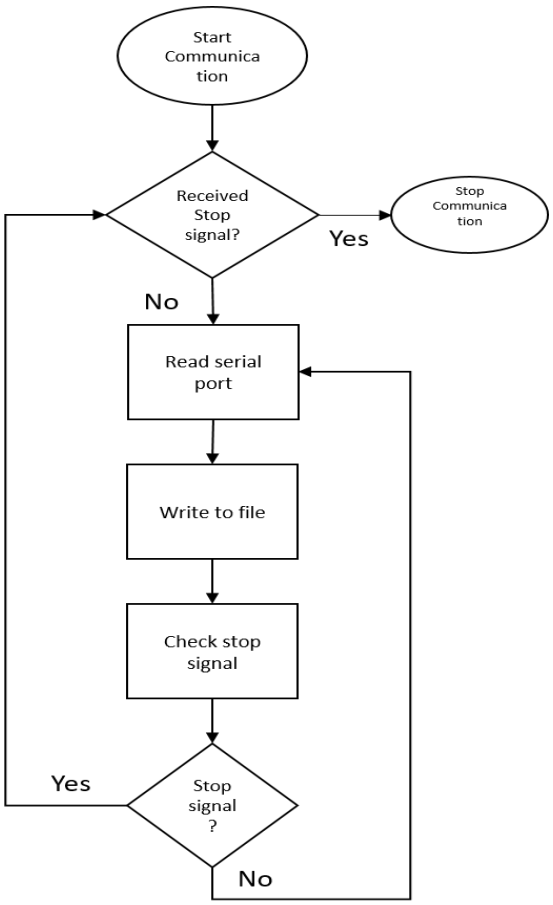


Figure 18. Flow Diagram of Serial Communication

Figure 18 shows the process flow of the serial communication procedure during the test library implementation phase. A separate python module is designed to accomplish the task that is executed from the robot framework test library. Besides, the test library also produces a stop signal on the completion of the test process which results in stopping the serial communication.

4.2.3 Robot Framework test library

Robot Framework test library consists of multiple other test libraries to execute different test cases along with a *"test_suite"* folder or templet to execute these test libraries. This "*test_suite*" folder consists of two separate tests running files for device tests and serial

port tests respectively. Both test running modules are executed in parallel with the help of the “Pabot” library.

Pabot library is a parallel test executor for Robot Framework test files. It provides test level split, which results in parallel execution of test cases, results in well-structured testing. Besides, also reduces test process time. It is an open-source python library with an apache 2.0 license (Pabot, 2019). In this testing project, the parallel operation helps to read serial data and execute tests at the same time. The “test_suite” folder follows the structure described below.

Test_suite -|

*|- **Device_tests_module** -|*

|- Device_tests_resource file

|- Log test keywords

|- Mqtt test keywords

|- Ota test keywords

*|- **Serial_port_tests_ module** -|*

|- Serial_port_tests-resource file

|- serial_test keywords

Algorithm 2. Test Suit Architecture

The test suite includes two main test modules mentioned below

- ***Device test module***
- ***Serial port tests module***

Both are executed in parallel with the Pabot library. Moreover, both module contains resource files containing additional keywords to run the test cases. Test_suite is utilizing various built-in and external libraries to run these keywords. These libraries or either required to be installed or may be placed into a separate folder within the test directory.

The working structure of both main test modules *“device test module”* and *“Serial port tests module*” is explained in detail.

4.2.3.1 Device_tests_module

This module contains further three major user-defined higher-level keywords representing three different tests enclosed with multiple keywords from various libraries to perform functional testing. The test file for the device_tests_module is shown in algorithm 3 along with the explanation of defined keywords.

```
*** Settings ***
Library         MQTTLibrary
Library         Process
Library         Collections
Library         OperatingSystem
Library         String
Library         Collections
Library         SerialLibrary


Resource  ../Resources/MQTT/Mqtt.robot          # Code for running
all MQTT tests

Resource  ../Resources/LOGGING/logging.robot    # Code for running
all device log tests

Resource  ../Resources/OTA/Ota.robot      # Code for running all
OTA tests

*** Keywords ***

     Test_log

         logging.test_logging    #--# 1      # Code for this test
         is in Resources/LOGGING/logging.robot

     MQTT

         mqtt.Connect_to_broker   #--# 2     # Code for this test
         is in Resources/MQTT/Mqtt.robot

         Repeat Keyword      3 times     mqtt.subscribe_and_vali-
         date_payload    #--# 3          # Code for this test is
         in Resources/MQTT/Mqtt.robot

         mqtt.publish_configurations_to_timer   #--# 4          #
         Code for this test is in Resources/MQTT/Mqtt.robot
```

```
OTA

    Ota.publish_configuration_to_firmware            # Code
    for this test is in Resources/OTA/Ota.robot
```

Algorithm 3. Test file for Device_module_tests

I. validation_of_logs

This keyword works in two steps, the initial task is to get “Data-logging-file” and in the subsequent step, find some prescribed strings from it. Test results are dependent upon the obtainability of those marked specific strings. i.e. a keyword from the Built-in library, *“Should Contain Any”* can be utilized to find these strings. A code snips from the library is shown below

```
test_logging    #--# 1
${TextFileContent}=    Get File     ${LOG_FILE_URI}
Log    ${TextFileContent}
Should Contain Any       ${TextFileContent}      ${keyword}
```

Algorithm 4. Log Test Code Snips

II. MQTT_test

A major part of the testing library consists of wireless communication tests, where MQTT is used as the underlying protocol for communication between the test library and device. This module contains various keywords from the Robot Framework MQTT library explained in section two under the Robot Framework introduction topic. It is a part of the “device test module”, which works as resource files containing the keywords to help in the test process.

This module is designed according to device functionality. The working principle and testing architecture are explained here in detail showing three key steps.

- Connect to broker and validate connection is established
- Subscribe to the topic and receive payload, then validate it
- Publish new configuration to the device and validate its updated functionality

The flowchart is showing the test design architecture that follows the functional key test requirements.

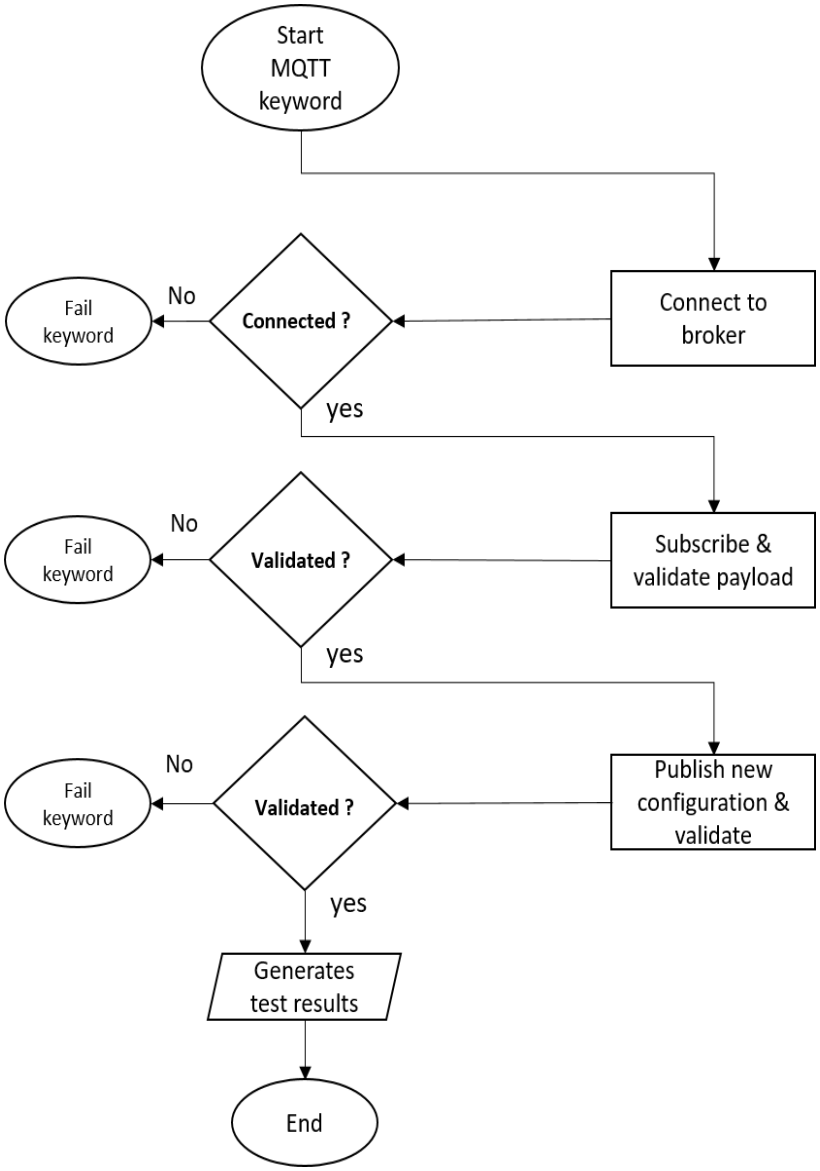


Figure 19. The basic Structure of the MQTT Test Module

One major block of this architecture is related to receiving and validation of payload is further explained. This stage consists of various important sub-steps highlighted here.

- Subscribe to the topic and receive payload
- validate the first key-value by applying boundary test with already known limits
- Initiate any action if required before proceeding to the next value
- validate the next key-values until last one by applying boundary test with already known limits

An example of payload and a validation data set is shown to develop a clear understanding of the validation process in picture 6.


```
{
  "topic":  "topic/DeviceData",
  "type":  " DeviceData ",
  "timestamp": 1587041108,
   "data": {
          "formatted": {
          "vbusMin": 15,
          "vbusMax": 10.99,
          "vbusMean": 10.99,
          "lever": 0,
          "tempPcb": 29.86,
          "tempMotor": 23.86,
          "tempBat": 23.69,
           }
       }
 }
```

```
{
  "topic": "topic/DeviceData",
   "type": " DeviceData ",
   "data": {
          "formatted": {
               "vbusMin": {
                   "greater-than": 8.0,
                   "less-than": 13.0
              },
                "vbusMax": {
                    "greater-than": 8.0,
                    "less-than": 13.0
               },
                 "vbusMean": {
                     "greater-than": 8.0,
                     "less-than": 13.0
               },
                  "lever": {
                      "greater-than": -10,
                      "less-than": 100
                },
                  "tempPcb": {
                      "greater-than": 0,
                      "less-than": 100
                },
                  "tempMotor": {
                      "greater-than": 0,
                      "less-than": 100
                },
```

Payload Received

```
                "tempBat": {
                    "greater-than": 0,
                    "less-than": 100
                }
            }
        }
    }
}
```

Boundry test file

Picture 6. Payload Example and Expected Boundary Limits

This process is described in more detail with the help of a flowchart below.

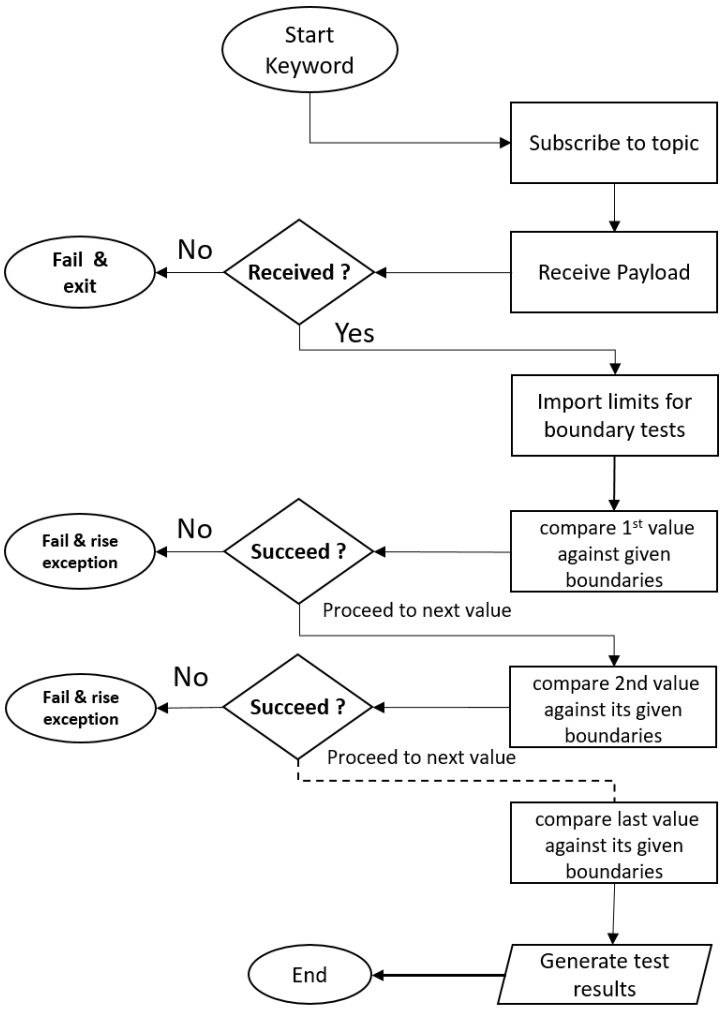


Figure 20. Architecture of Payload Validation Block

A user defined keyword from MQTT resource file is shown in algorithm 5

```
Connect_to_broker
     Connect     ${BROKER_URI}    ${BROKER_PORT}
subscribe_and_validate_payload
     Device_data
     Device_Statistics
     Battery_indecators
     Device_diagnostic
     Device_config
publish_configurations_to_device
     Publish       ${PUB_TOPIC_TIMER}  ${Payload}
Subscribe and validate      ${SUB_TOPIC}    0    ${messages}
${time_interval}
```

Algorithm 5. Code Snips from MQTT Resource File

III. OTA_update

As described earlier, the device uses MQTT messaging protocol for communication. The Over the Air (OTA) feature is designed as, the machine receives firmware update instruction along with WebHost address containing updated firmware file, and starts the firmware update. The process is designed with special keywords strings via serial logs such as "*Start Device OTA", "Device_ota_begin succeeded", "Firmware upgrade completed"* when upgraded successfully or *"Firmware upgrade failed"* in case of any interruption. In case of successful upgrading, the device restarts, and the firmware version updated. Moreover, this new firmware version information is also published through messages under the topic, configData. Accordingly, the test follows these designated steps.

- Publish JSON message to the machine and start monitoring serial data logs
- Find indication keywords string related to the upgrading process highlighted earlier
- Wait until one of those Keyword succeeds, then terminate the test

A code snips from the test library is shown below.

```
*** Keywords ***

Publish    ${PUB_TOPIC_FIRMWARE}      ${firmware_string}      0

${TextFileContent}=    Get File    ${LOG_FILE_URI}

Wait Until Keyword Succeeds   1min   5 sec   get_start_device_ota

Wait Until Keyword Succeeds    1min    5 sec   get_Firmware_up-
grade_failed

Wait Until Keyword Succeeds         1min    10 sec      get_de-
vice_ota_begin

Wait Until Keyword Succeeds        1min    10 sec     get_Firm-
ware_upgrade_completed

Disconnect
```

Algorithm 6. OTA Test Code Snips

For a clear understanding, the process is explained with the help of a flow chart.

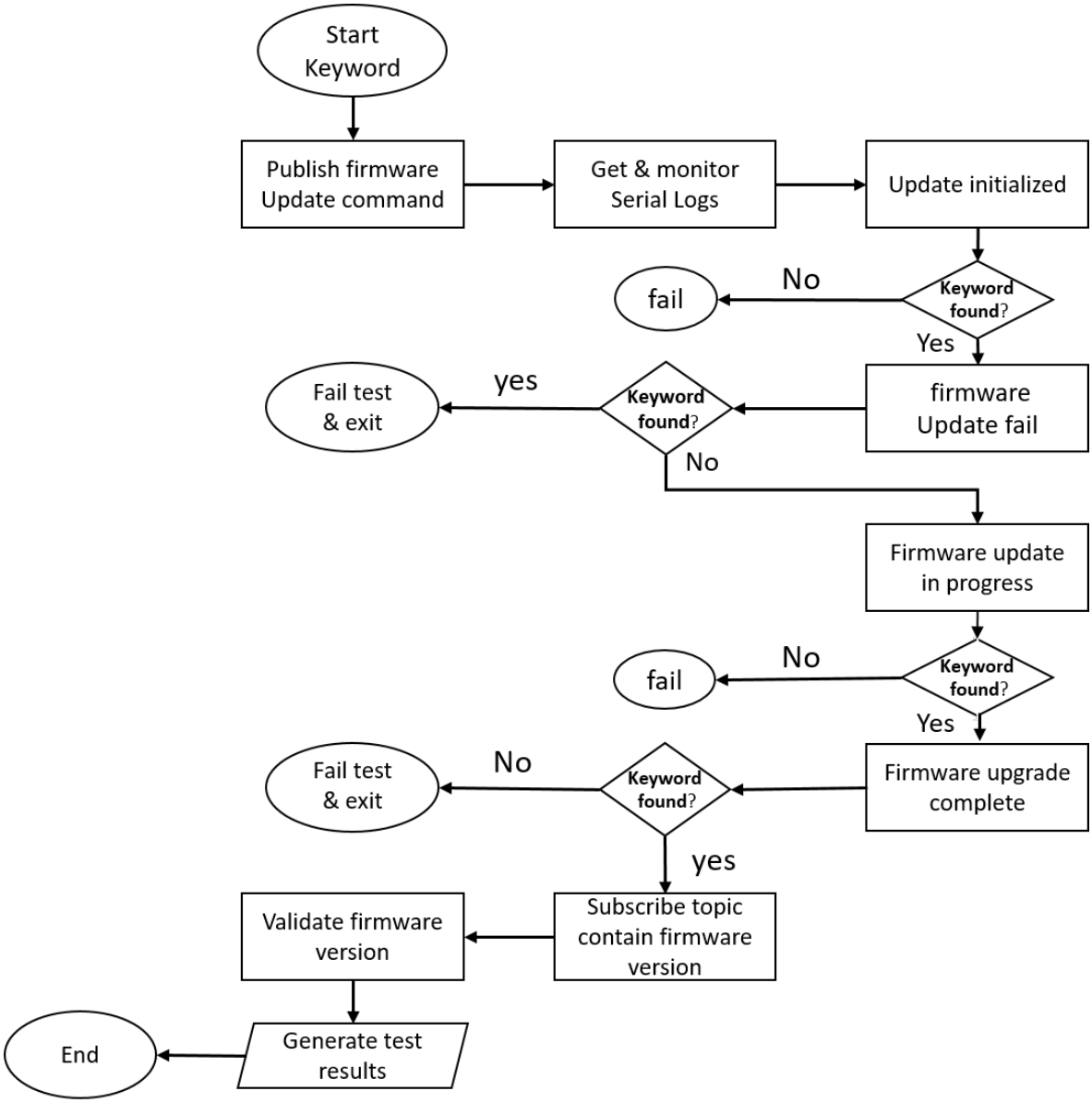


Figure 21. OTA Process Generalized Structure

4.2.3.2 Serial test module

The serial test module contains two major executable higher-level keywords.

I. Check if port is open

The initial step is to monitor if the given “COM Port” is available for communication or not. Such as in some cases it could be already occupied by any other program which is already running.

The keyword “*Check if port is open”* is a user-defined higher-level keyword that utilized two other given keywords from an external Robot Framework library called “Serial Library”. These keywords are “Add Port” and “Port Should Be Open”. Code snip or Robot Framework keyword along with python functions from the library is given below that is handling described keywords.

```
Check if port is open
    Add Port    ${SERIAL_PORT}
    Port Should Be Open
    [Teardown]    Delete All Ports
```

Algorithm 7. Robot Framework Keyword

```
def port_should_be_open(self, port_locator=None):
    """
    Fails if specified port is closed.
    """
    asserts.assert_true(
        self._port(port_locator).is_open,
        'Port is closed.'
    )
```

Algorithm 8. Code Snippet from the Serial Library (Seriallibrary, 2017)

II. Monitor serial port

This keyword is also part of the serial port test module. To start and monitor serial port communication, a separate python module is written named “run-serial.py”. Robot Framework is capable to run an external python module with the help of a built-in “Process Library”.

```
*** Settings ***
Library    OperatingSystem
Library    String
```

```
Library    Collections
Library    BuiltIn
Library    Process
Library    SerialLibrary

*** Keywords ***
     Check if port is open

          Add Port    ${SERIAL_PORT}    loop://

          Port Should Be Open

          [Teardown]    Delete All Ports

     Monitor serial port

           Process.Start Process     run-serial.py   cwd=C:/devel-
           opment/RFW-testing-Project                   shell=True
           stdout=stdout.txt      alias=process1
```

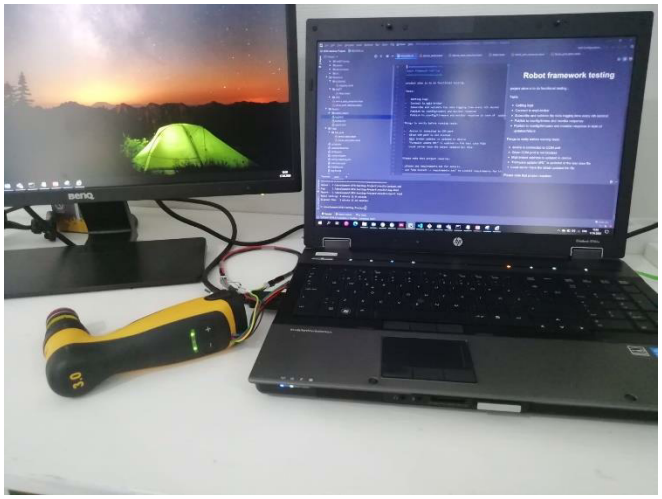
Algorithm 9. Robot Framework Keyword for Serial Tests

Here in Algorithm 9, “Start process” is a keyword from process library to run any process, “run-serial.py” is an external module, “cwd” stand for the current working directory, while “shell=True” is a special notation in Robot Framework that either is it required to run the external module with CMD prompt or not. Program flow is the same as explained in section 4.2.1 earlier.

5 Test Implementation

5.1.1 Preparing the testing setup

The testing setup consists of a system containing a Robot Framework test library, serial cable, and a device under test. Picture 7 shows the testing setup. Test library that includes test files is ready to initialize for running the test sequence. Moreover, a local wireless host is required as mentioned earlier in section 4.2, for wireless connectivity between the system containing the Robot Framework test library and the device under test. Therefore, both connected to the same wireless network.



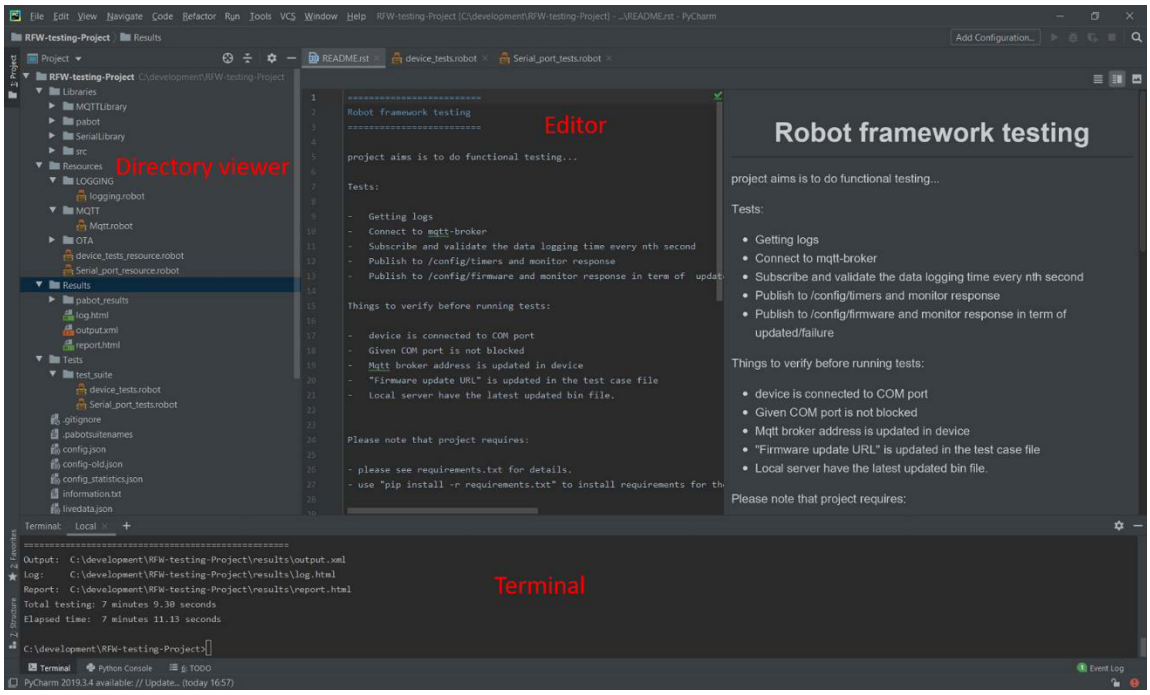
Picture 7. Test Setup

5.1.2 Execution of the testing setup

As explained earlier the “pabot” library is utilized for parallel execution of test files. There are different methods to launch the test case such as running the test from the command line, by using any IDE i.e. Pycharm and RIDE, from task manager to running as per schedule, or running remotely through CI/CD platform like as Gitlab and Travis. In this particular scenario, Pycharm IDE is utilized for running the test sequence. A screenshot of Pycharm IDE, with the ongoing testing project, is shown in picture 8.

The user interface consists of three main parts.

- Directory viewer to access the different files and folder within test library
- Editor to write and modify test cases
- Terminal to start and monitor the testing process.



Picture 8. Screenshot of pycharm IDE

The suggested structure for the test library is highlighted in picture 9. It mainly consists of these sub-directories.

- Tests

Tests folder contains the test case files to be executed. Here the *test_suite* folder contains the test files required to be executed in parallel by using the Pabot library.

- Resources

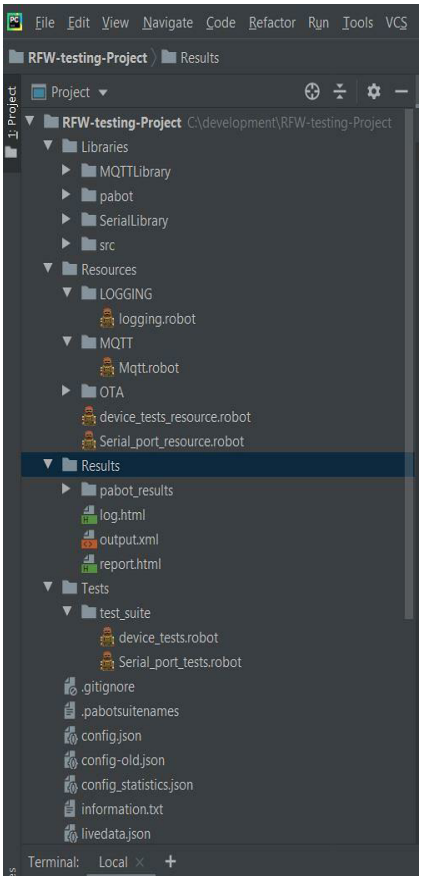
The resources directory contains all the resource files having user-defined keywords to support the test execution files. This architecture results in a simpler and understandable test case files.

- Libraries

This folder contains all the external libraries used in this project. These libraries contain the high-level keywords utilized in writing test cases for the device.

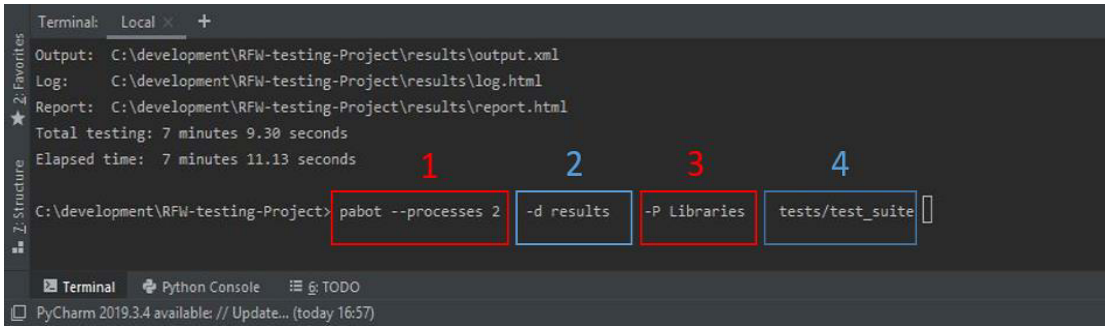
- Results

All the results generated by the Robot framework, along with some additional reports produced by the Pabot library are stored in this folder. These results can be stored in any other place as well for later reference.



Picture 9. Suggested Directory Structure

As told, pycharm terminal is used to implement the test cases in this project by executing the specific command show in picture 10.



Picture 10. Terminal to Execute Test Sequences

The command mentioned below consists of four parts.

"pabot --processes 2 -d results -P Libraries tests/test_suite"

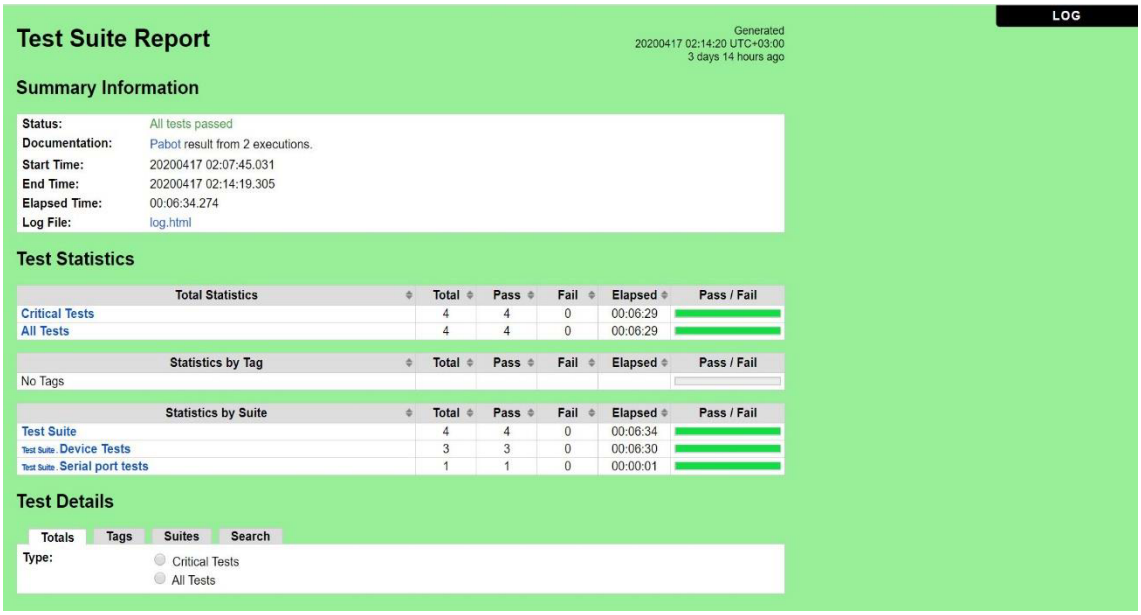
1. **"***pabot --processes 2***"** shows that the pabot library is utilized to execute 2 processes in parallel.
2. *"-d results"* represents the folder where the results would be stored.
3. *"-P Libraries"* is mentioning the directory where all the external library resources are placed.
4. *"tests/test_suite"* is heading towards the directory where all the test executable files are stored.

5.2 Results

Test run with Robot Framework results in generating three files.

1. **Report.html**

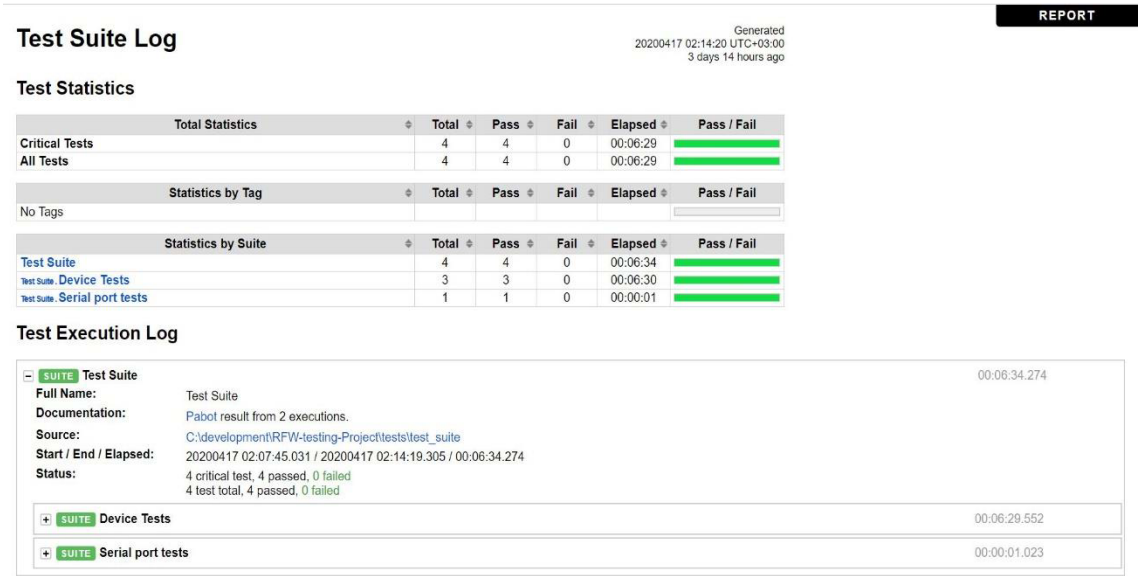
This file contains a summary of the executed tests. It includes basic information like test execution time, and the name of the tests executed with results in terms of pass/fail.



Picture 11. Screenshot of Report.html file

2. Log.html

This file contains more comprehensive results with the details of each of the tests.

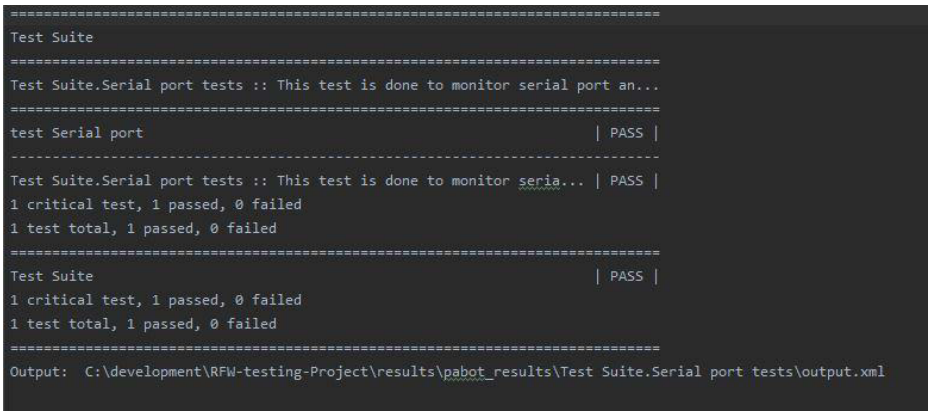


Picture 12. Screenshot of Log.html file

3. Output.xml

This file contains the same information as log.html but in a more machine-readable format.

In addition to these result files generated by Robot Framework, Pabot library also generate an additional "*pabot_results"* folder containing the out.xml files for each of its test process. Also, snips of the terminal are stored in the same folder. An example of a terminal snip shown in picture 13.

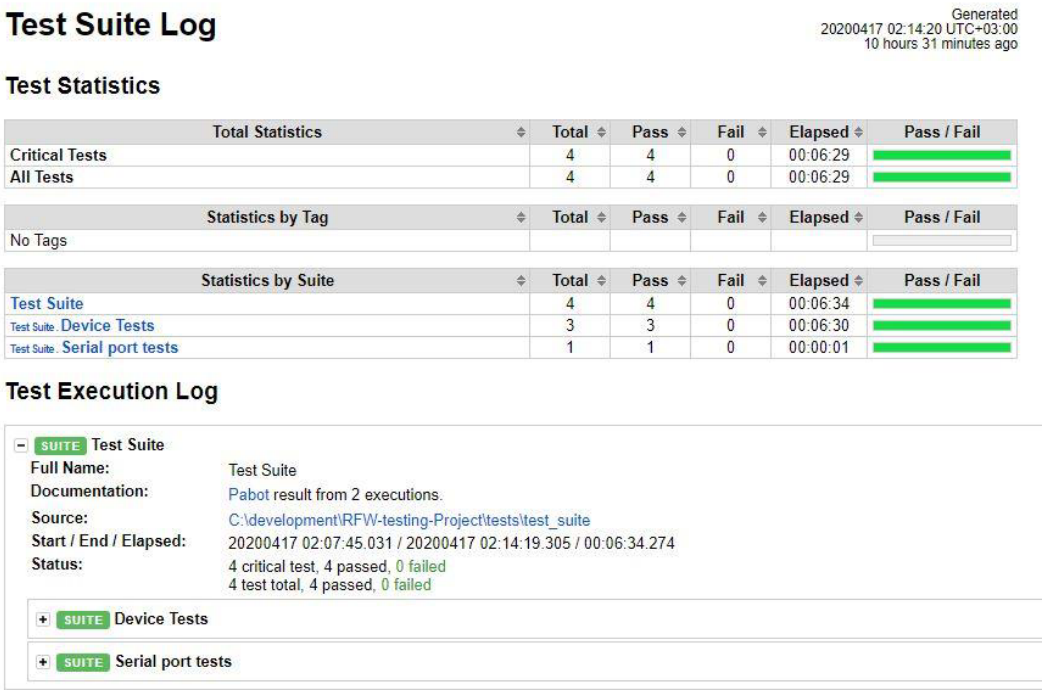


Picture 13. Pabot Results for the Serial Test

In section 4.2.3 during designing the Robot Frame testing library, it is mentioned that the test sequence is divided into two sections.

- Device test module
- Serial port tests module

Both sections are executed in parallel, results in positive output as seen in a partial screenshot in picture 14.



Test Suite Log

Generated
20200417 02:14:20 UTC+03:00
10 hours 31 minutes ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	4	0	00:06:29	
All Tests	4	4	0	00:06:29	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Test Suite	4	4	0	00:06:34	
Test Suite . Device Tests	3	3	0	00:06:30	
Test Suite . Serial port tests	1	1	0	00:00:01	

Test Execution Log

SUITE Test Suite
Full Name: Test Suite
Documentation: Pabot result from 2 executions.
Source: C:\development\RFW-testing-Project\tests\test_suite
Start / End / Elapsed: 20200417 02:07:45.031 / 20200417 02:14:19.305 / 00:06:34.274
Status: 4 critical test, 4 passed, 0 failed
4 test total, 4 passed, 0 failed

SUITE Device Tests

SUITE Serial port tests

Picture 14. Partial Screenshot of log.html file

5.3 Analysis of test results

The results of both parallel executed tests are analyzed separately. As the test designing being already discussed in the previous chapter, here only the results are analyzed.

5.3.1 Serial port tests

As explained in section 4.2.3, the serial port test is implemented to access the serial port and initiate a serial connection with the device under test. Besides, read serial port data, store it to data logging file and terminate the communication on completion of the testing process. It is evident from the result file below that all the tests are executed successfully. Further details are also accessible by expending any of these test sub-sequences in the log.html file.

Picture 15. Partial Screenshot of Serial Port Tests

5.3.2 Device test module

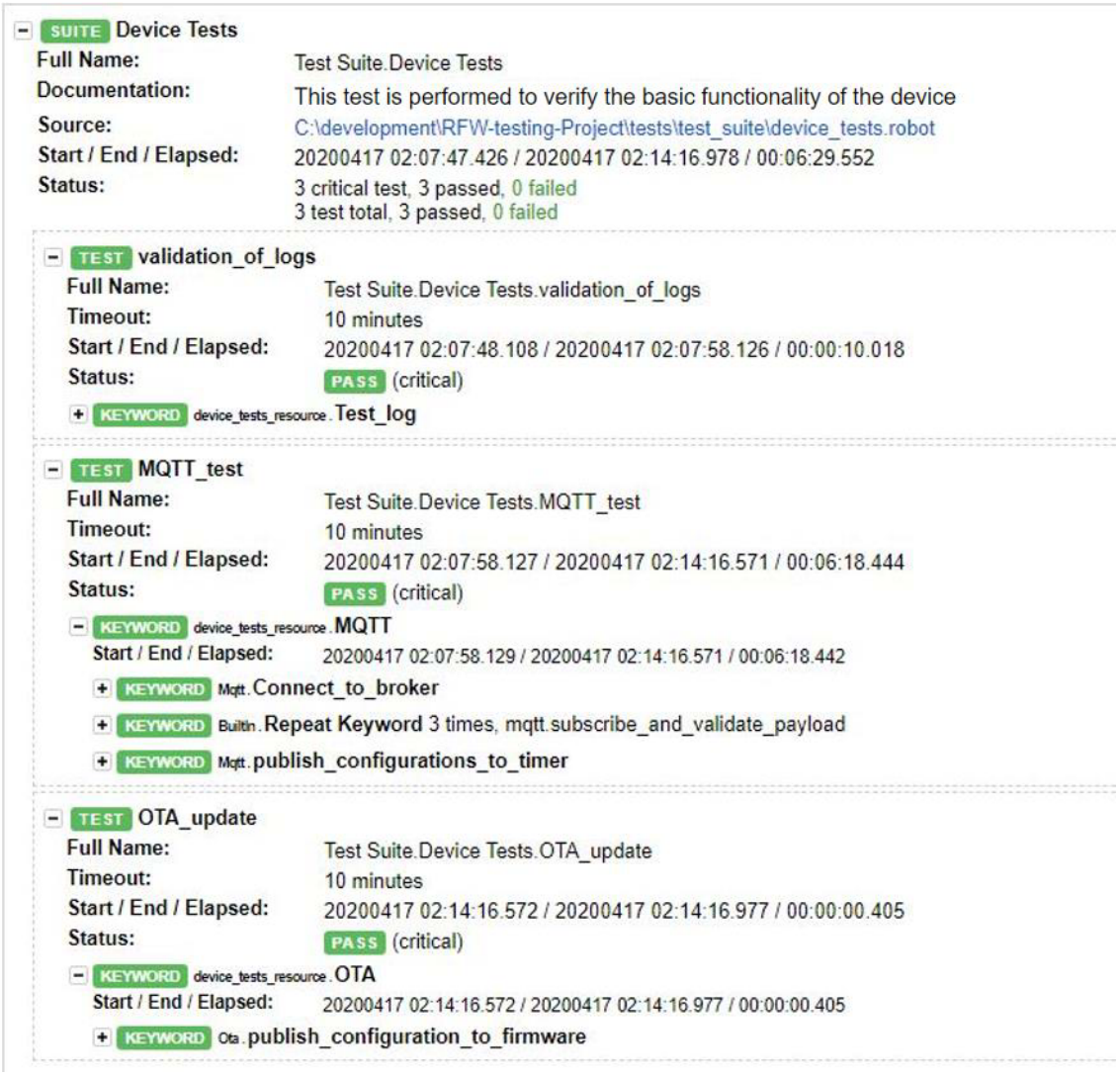
The device test module consists of three sub-modules, explained in section 4.2.3. Partial screenshot of the log file below shows the results of all three sub-modules, Validation of logs, MQTT tests, and OTA update along with their sub-sequence tests. The results of all these three sections are discussed here.

1. **Validation of logs**

Serial logs, received from the device are validated across the specified keywords. All the tests are executed successfully and result in Pass as shown in picture 16, section one.

2. **OTA update**

Over the air update (OTA) test design, and the process flow is already described in the Robot Framework testing library designing section 4.2.3. Here from the partial screenshot of the log.html file picture 16, it can be observed that the OTA update test is conducted successfully. After getting a firmware update, the device would restart automatically with the updated firmware.



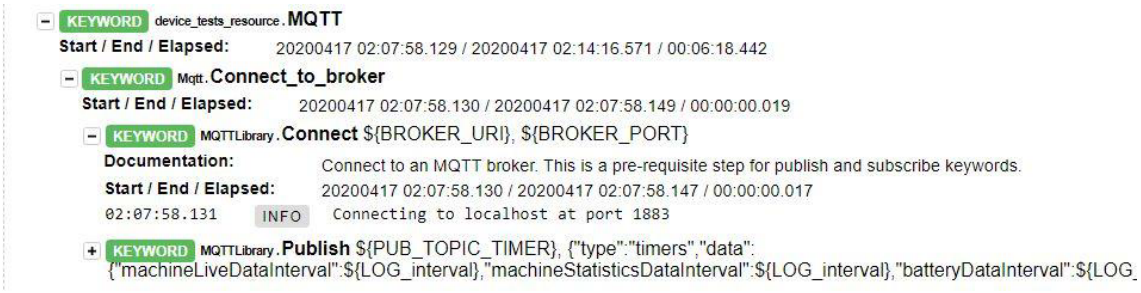
Picture 16. Partial Screenshot of Device Test Module

3. **MQTT tests**

MQTT tests are one of the main section of the test library that is required to be explained with more details. MQTT test sequence furthermore consists of three sub-sequences, Connect to the broker, Subscribe to a specific topic and validate the received payload, and Publishing some configuration to device and validate the device response. All these tests are already explained in the test library designing section. The results of these sections are analyzed here.

1. Connect to the broker

Partial screenshot from the log.html file clearly showing the keywords used in the test sequence and their test output. The connection to the broker is established successfully.



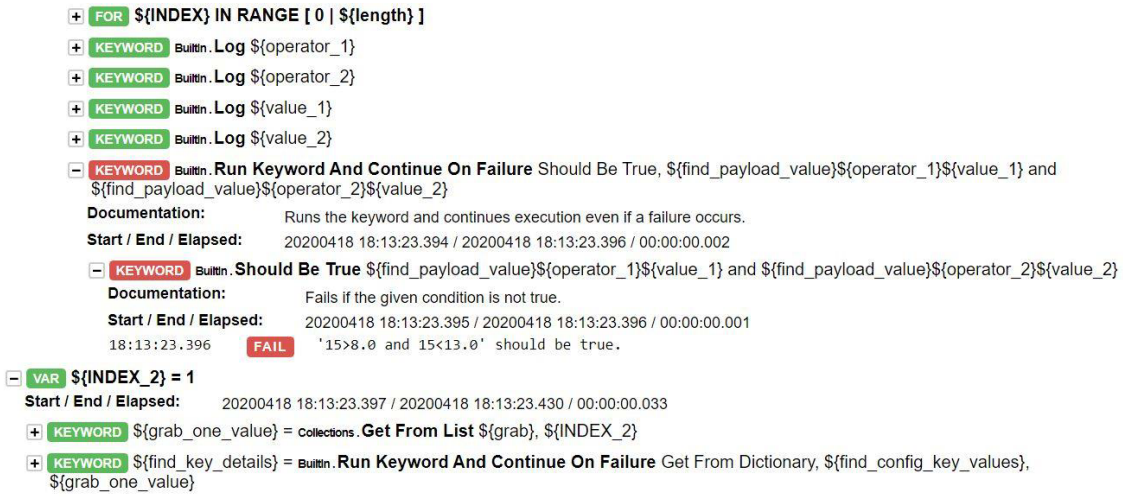
Picture 17. Partial Screenshot Showing Connect Keyword

2. Subscribe and validate the payload

Here the purpose of the test is to firstly, make sure that the payload is received within a specific period than validate the received payload across some already predefined boundary limits. In section 4.2.3, figure 20 shows the test architecture of the payload validation test. Also, an example payload and a validation data set are shown to develop a clear understanding of the validation process. During the test, one after another each value is verified where in case of test failure, the exception is rise. In picture 18 below, all the keywords highlighted by the arrows are a different kind of data sets under the different topic name, received from the device under test. One of those data set “Device data” is expended to highlight in more detail. Moreover, this keyword “Subscribe and validate payload” is executed three times to make the testing process more comprehensive and satisfactory. Three different payloads concerning time intervals are validated.

Picture 18. Partial Screenshot of Payload Validation Test

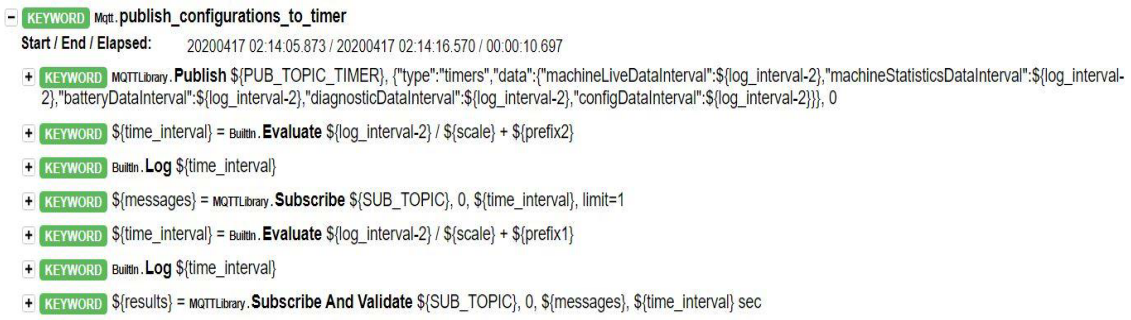
During the validation process, in the case of a boundary test failure, an exception is rise before going to the next value. Picture 19 shows a partial screenshot in case of test failure.



Picture 19. Partial Screenshot of a Failure Test

3. Publish instruction to the device

This test is performed to verify that the device receiving the instruction from the broker and implementing those instructions successfully. Here the instructions are sent regarding the change in data-logging interval, and verification of change is verified. Both tests are executed successfully. A partial screenshot is attached as well.

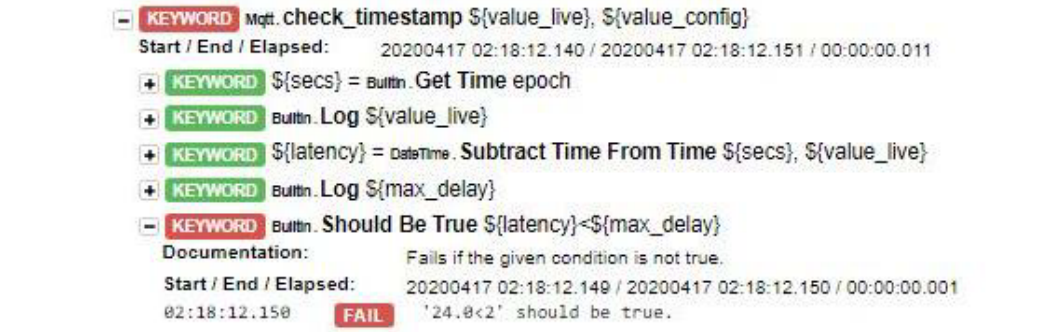


Picture 20. Partial Screenshot of the Publish-Configuration Test

5.3.3 Latency test

Latency is the delay caused in the transmission of payload from one point to another. Although this test is not highlighted in the log.html file, however, it is the part of the testing sequence as well. The basic concept of this test is to get the epoch timestamp by using Robot Framework DateTime library and compare it with the timestamp encompassed into each payload received from the device. The result of the tests is measured against the maximum delayed cause during the process. The partial screenshot for both cases, pass and fail are provided below.

Picture 21. Latency Test Executed Successfully

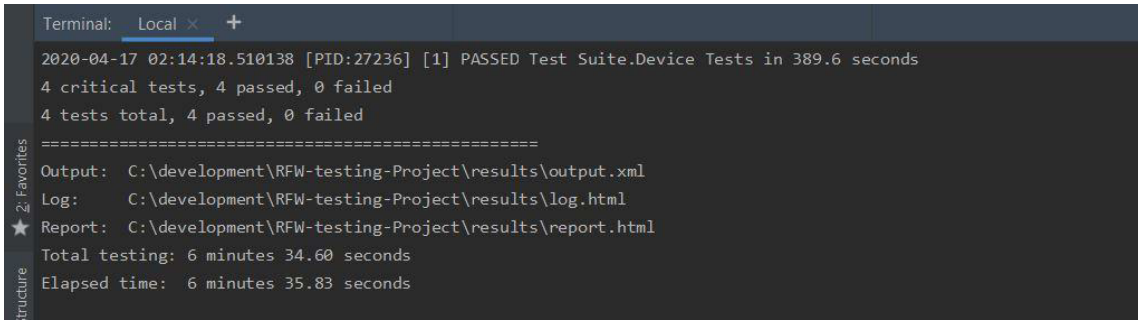


Picture 22. Latency Test Failed

5.4 Conclusion about the testing process and its limitations

The following conclusion is made from the experience gathered during the test implementation phase, as well as after analyzing the generated results.

- Test setup using serial and wireless communication protocol to interact with the device works according to expectations.
- Robot Frame Test library is designed according to machine functionality works as expected.
- For some specific tests, an additional file is required, containing boundary limits that work fine if it's according to a structure.
- It already explained earlier the three collections of payload received are tested during the process, even than the total testing time is quite low compared to manual testing.



Picture 23. Screenshot of the Terminal after Test Execution

To summaries, the results of the different tests performed are given below in table 4.

Table 4. Test Conducted and their Results

Test name	Result
Test serial port	Pass
Monitor serial port	Pass
Validation of logs	Pass
Connect to the broker	Pass
Subscribe and validate the payload	Pass
Publish instruction to the device and validate	Pass
Over the air (OTA) updates	Pass
Latency test	Pass

However, some limitations realized during the testing process are also mentioned here.

- The file containing boundary limits should be specifically structured, otherwise, the test would fail, or consequences in false results.
- Test case writing is hard, a smaller change in device functionality requires changes in the testing library.
- One device can be tested at a time with the system having the Robot Framework test library.
- The testing platform not supported for remotely testing at this stage, introducing CI/CD to the automated testing platform can help to increase this capability.

6 Conclusion and Further Scope

The development of any platform from scratch is always challenging and often requires a significant amount of research and planning. The process also presents with many learning opportunities and an introduction to new technologies. The focus of this thesis is to establish an automated testing platform to perform the functional testing against device description. Robot Framework is utilized as the testing platform by the implementation of various built-in and external libraries.

Firstly, after the introduction to wireless automation and automated testing, various available short-range wireless communication protocols for small industrial cordless devices are reviewed. Following the functional description of the device under development, *Wi-Fi* has been suggested as a protocol. After that embedded system testing is introduced along with its various types than the automated testing platforms are discussed and Robot Framework is proposed as the suggested testing platform.

Secondly, after completion of a theoretical background in the first part, the functional testing requirements and designing of test cases began. It starts from the introduction to the machine under test with the discussion about its functionalities that leads to the test planning phase. Robot Framework test library is designed according to the test requirements defined previously. Thirdly, the implementation phase started where after establishing the test setup, the test sequences from the Robot Framework test library executed against the real hardware device. Results and reports generated by Robot Framework are analyzed and discussed concerning expected device behavior.

As the project is still under development and up-gradation phase, some limitations related to the automated testing platform would be covered in the next phase. Especially during the functional testing of the device, some tests required the physical interaction with the device under test such as pressing the button or monitoring the Led lights. Somehow the requirements of physical interaction limit the utilization of the automated testing platform. The next stage of this project is to form a physical setup, consisting of sensors and actuators to perform physical interaction, linked with the automated testing

platform. This hardware setup would help to introduce the DevOps and CI/CD to firmware development and testing setup to add remote testing capabilities to the platform. This would be the last stage of a fully automated testing platform.

6.1 Future scope

In the case of the automated testing platform, it is always important to consider the latest technologies from the field and learn from the experience of the others as well. Besides, the platform should be adaptable to the latest technologies at any stage. It would be helpful to enhance the performance in the future and to increase the functional capabilities of the platform. Introducing Artificial Intelligence (AI) and Machine learning (ML) technologies to the existing testing platform can result in faster, efficient, and reliable with numerous advanced functionalities. A short introduction of these technologies is provided along with their scope in terms of the current project.

6.1.1 Artificial Intelligence (AI)

Artificial Intelligence technology is related to the development of intelligent machines and systems. Machines with AI, capable of utilizing intelligence techniques, developed by humans to identify and solve numerous problems. AI application can be categories to two major sections, a pattern tries to simulate the human brain process called Artificial Neural Networks (ANNs) and the other that follows and uses the cognitive patterns stated as the Conventional Artificial Intelligence (Gharbi & Mansoori, 2005). Introducing Artificial intelligence to the automated testing platform would make it capable to take intelligent decisions without human intervention to handle minor changes in the code and regarding further actions in case of failure of a test case.

6.1.2 Machine learning (ML)

Machine learning is a computer application for performance optimization against a benchmark using exemplary data or on the bases of past experience. The parameters of a given model can be optimized using training data or previous knowledge. The model can be used to predict the future called *predictive* or gain knowledge from the received

data called *descriptive* or sometimes with both functionalities (Alpaydin, 2014 : 3). Some main classes of machine learning are mentioned here.

- **Supervised Learning**

In supervised learning, a supervisor is responsible for feeding the system with the training sets of labeled data consists of the real observation on input and output values. After that machine tries to form a certain model based on received training data. Regression and classification methods are an example of supervised learning (Alpaydin, 2014: 3-7).

- **Unsupervised Learning**

In unsupervised learning, instead of providing the training set, input values are provided where the machine tries to find regularities between these values. Repetition of certain patterns from the input is monitored along with their response, the generated results in the form of statistics are known as density estimation (Alpaydin, 2014: 11-13).

- **Reinforcement Learning**

In some specific scenarios, the output of a system is dependent upon several steps. In these cases, rather than a single action, the whole sequence of steps matters which generates a good result. Here the purpose of introducing the machine learning is to generate a good working policy, learn from the previous action with good output (Alpaydin, 2014: 13).

Introducing these technologies to the automated testing platform would help in many ways. As an example, some use cases are mentioned here.

I. Test case writing

It is quite an intensive process to write test cases to perform testing on actual code. Sometimes changes are required to test case script, even if a small change is made in the actual code. The automated test platform should be intelligent enough to handle these smaller changes in the actual code. Machine learning (ML) and artificial intelligence (AI)

algorithm can be included in the automated testing platform to write smart test cases, intelligent enough to handle minor changes in code.

II. Defining boundary testing values

Currently, the data collected from the sensors is validated against some predefined boundary values provided through an additional configuration file. Moreover, a specified format of the configuration file is required as well. AI can be used here to form an intelligent test sequence to grab boundary values even without having a specified file format. Besides, ML algorithms can also help here to train test setup against the normal values of the sensor. In case of some other values, different than past samples, test sequence would raise an exception which would work as boundary value for tests.

Fault Approximation

As explained earlier, the data collected from the machines is stored in a database. This data can be optimized and analyzed later for various purposes. For instance, in case of any machine failure, its previous data is analyzed. Stored data helps to get an overview of machine behavior before its failure. Such as high vibration values show irregularity in machine usage or issues with machine assembly, whereas higher current values indicate some faults related to specific electronic components. To summarize, stored data helps to get an idea about the fault accrued. Here machine learning can help in a very efficient way where the stored data and feedback from the testers can be used for the training purpose. In the case of machine failure, the machine learning algorithm can highlight the approximate reason for failure. This analysis would help repair team to fix faulty machines in less time. Furthermore, it can help the “Research and Development” (R&D) team in better planning for further development.

References

Agiwal, M., Roy, A., & Saxena, N. (2016). Next generation 5G wireless networks: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, *18*(3), 1617–1655. https://doi.org/10.1109/COMST.2016.2532458

Aiman, M., Bahrin, K., Othman, F., Hayati, N., Azli, N., & Talib, F. (2016). *Jurnal Teknologi Full Paper INDUSTRY 4.0: A REVIEW ON INDUSTRIAL AUTOMATION AND ROBOTIC*. *78*, 2180–3722. www.jurnalteknologi.utm.my

Al-Kashoash, H. A. A., & Kemp, A. H. (2016). Comparison of 6LoWPAN and LPWAN for the Internet of Things. *Australian Journal of Electrical and Electronics Engineering*, *13*(4), 268–274. https://doi.org/10.1080/1448837X.2017.1409920

Al-sarawi, S., Anbar, M., Alieyan, K., & Alzubaidi, M. (2017). *Internet of Things (IoT) Communication Protocols : Review*. 685–690.

Alpaydin, E. (2014). Introduction to Machine Learning Ethem Alpaydin. *Introduction to Machine Learning, Third Edition*.

Ashbacher, C. (2002). Test-Driven Development: By Example, by Kent Beck. In *The Journal of Object Technology* (Vol. 2, Issue 2). https://doi.org/10.5381/jot.2003.2.2.r1

Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E., & Mailloux, L. O. (2017). The Z-Wave routing protocol and its security implications. *Computers and Security*, *68*, 112–129. https://doi.org/10.1016/j.cose.2017.04.004

Baronti, P., Pillai, P., Chook, V. W. C., Chessa, S., Gotta, A., & Hu, Y. F. (2007). Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. *Computer Communications*, *30*(7), 1655–1695. https://doi.org/10.1016/j.comcom.2006.12.020

Briand, L., & Labiche, Y. (2002). A UML-based approach to system testing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2185*, 10–42. https://doi.org/10.1007/s10270-002-8208-5

Cheng, J., Chen, W., Tao, F., & Lin, C. L. (2018). Industrial IoT in 5G environment towards smart manufacturing. *Journal of Industrial Information Integration*, *10*, 10–19.

https://doi.org/10.1016/j.jii.2018.04.001

Ebert, C., & Jones, C. (2009). Embedded software: Facts, figures, and future. *Computer*, *42*(4), 42–52. https://doi.org/10.1109/MC.2009.118

Freeman, H. (2002). Software testing. *IEEE Instrumentation and Measurement Magazine*, *5*(3), 48–50. https://doi.org/10.1109/MIM.2002.1028373

Frotzscher, A., Wetzker, U., Bauer, M., Rentschler, M., Beyer, M., Elspass, S., & Klessig, H. (2014). Requirements and current solutions of wireless communication in industrial automation. *2014 IEEE International Conference on Communications Workshops, ICC 2014*, 67–72. https://doi.org/10.1109/ICCW.2014.6881174

Gharbi, R. B. C., & Mansoori, G. A. (2005). An introduction to artificial intelligence applications in petroleum exploration and production. *Journal of Petroleum Science and Engineering*, *49*(3–4), 93–96. https://doi.org/10.1016/j.petrol.2005.09.001

Gnad, A., Krätzig, M., Rauchhaupt, L., & Trikaliotis, S. (2008). Relevant influences in wireless automation. *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS*, 341–348. https://doi.org/10.1109/WFCS.2008.4638736

Goursaud, C., & Gorce, J. M. (2015). Dedicated networks for IoT: PHY / MAC state of the art and challenges. *EAI Endorsed Transactions on Internet of Things*, *1*(1), 150597. https://doi.org/10.4108/eai.26-10-2015.150597

Grenning, J. W. (2011). *Test-Driven Development for Embedded C* (J. Carter (ed.)). Pragmatic bookshelf. https://doc.lagout.org/programmation/C/Test-Driven Development for Embedded C %5BGrenning 2011-05-05%5D.pdf

Khan, M. A., Cherif, W., Filali, F., & Hamila, R. (2017). Wi-Fi Direct Research - Current Status and Future Perspectives. *Journal of Network and Computer Applications*, *93*(October), 245–258. https://doi.org/10.1016/j.jnca.2017.06.004

Kreibich, O., Neuzil, J., & Smid, R. (2014). Quality-based multiple-sensor fusion in an industrial wireless sensor network for MCM. *IEEE Transactions on Industrial Electronics*, *61*(9), 4903–4911. https://doi.org/10.1109/TIE.2013.2293710

Kushalnagar, N. C., Montenegro, G. (Microsoft C., & Schumacher, C. A. (2007). RFC4919:

IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. In *Request for Comments: 4919*. https://doi.org/10.1017/CBO9781107415324.004

Lee, E. A. (2000). What's ahead for embedded software? *Computer*, *33*(9), 18–26. https://doi.org/10.1109/2.868693

Li, X., Li, D., Wan, J., Vasilakos, A. V., Lai, C. F., & Wang, S. (2017). A review of industrial wireless networks in the context of Industry 4.0. *Wireless Networks*, *23*(1), 23–41. https://doi.org/10.1007/s11276-015-1133-7

Miller, R. W., & Collins, C. T. (2001). Acceptance testing. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *Miller*, *R.*, 112–121. https://doi.org/10.1007/3-540-45672-4_11

Mirka Oy. (2020a). *Mirka-Tools*. Retrieved March,3, 2020, from https://www.mirka.com/uk/uk/UK_TOOLS/

Mirka Oy. (2020b). *Mirka introduction*. Retrieved March,3, 2020, from https://www.mirka.com/uk/

Mirka Oy. (2020c). *myMirka*. Retrieved March,3, 2020, from https://www.mirka.com/mymirka/

Myers, G. J., Sandler, C. S., & Badgett, T. (2011). *The Art of Software Testing* (3rd ed.). John Wiley & Sons.

Naidu, G. A., & Kumar, J. (2019). Wireless Protocols: Wi-Fi SON, Bluetooth, ZigBee, Z-Wave, and Wi-Fi. In *Lecture Notes in Networks and Systems*. https://doi.org/10.1007/978-981-13-3765-9_24

Pabot. (2019). *Pabot Library*. Retrieved February ,20, 2020, from https://github.com/mkorpela/pabot

Paul, A., & Jeff, O. (2008). *Introduction to software testing* (1st ed.). Cambridge University Press.

Rahman, A. B. A. (2015). Comparison of Internet of Things (IoT) Data Link Protocols. In *Comparison of Internet of Things*.

Robot Framework. (2020). *Introduction and Examples.* Retrieved January, 10, 2020,

from https://www.robotframework.org

robotframework-mqttlibrary. (2019). *MQTT-Library*. Retrieved February ,10, 2020, from https://pypi.org/project/robotframework-mqttlibrary/

Salman, T., & Jain, R. (2017). Networking protocols and standards for internet of things. *Internet of Things and Data Analytics Handbook*, 215–238. https://doi.org/10.1002/9781119173601.ch13

Seriallibrary. (2017). *robotframework-seriallibrary*. Retrieved Feburary,18, 2020, from https://pypi.org/project/robotframework-seriallibrary/

Singh, P., Sharma, D., & Agrawal RIT, S. (2011). A Modern Study of Bluetooth Wireless Technology. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, *1*(3), 55–63. https://doi.org/10.5121/ijcseit.2011.1306

Tessaro, L., Raffaldi, C., Rossi, M., & Brunelli, D. (2018). LoRa Performance in Short Range Industrial Applications. *SPEEDAM 2018 - Proceedings: International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, 1089–1094. https://doi.org/10.1109/SPEEDAM.2018.8445392

Tsai, W. T., Yu, L., Zhu, F., & Paul, R. (2005). Rapid embedded system testing using verification patterns. *IEEE Software*. https://doi.org/10.1109/MS.2005.103

Zachariah, T., Klugman, N., Campbell, B., Adkins, J., Jackson, N., & Dutta, P. (2015). The internet of things has a gateway problem. *HotMobile 2015 - 16th International Workshop on Mobile Computing Systems and Applications*, 27–32. https://doi.org/10.1145/2699343.2699344