Quality validation of PCB-mounted sensors to prevent counterfeit components

Axel Borgdén & Hannes Hallén

2015

Lund University

minut

Master's Thesis

Faculty of Engineering LTH
Department of Biomedical Engineering

Supervisor: Christian Antfolk

Abstract

Counterfeiting of electronic components is a growing problem, leading to lost revenue for companies as well as unreliable products delivered to customers. There is no general way for small hardware companies to deal with the problem. Addressing counterfeiting is usually not a high priority for these companies, though it is still imperative for them to deliver products with guaranteed functionality.

This thesis, written at Minut AB in collaboration with Lund University, concerns the problem of counterfeit components in the tech industry. The approach is to test components during product manufacturing. A test system validating the components' performance is developed to guarantee the quality and functionality of the final product. The results show that the developed test system is capable of finding components with deviating performance.

A malfunctioning counterfeit component can be discovered by the test system, though, counterfeiting is a complex problem difficult to assess at individual component level. Introduction of test and production statistics can point to failing component batches. These statistics opens for further investigation in the search for counterfeit components.

Acknowledgements

This thesis was carried out at Minut AB in Malmö, Sweden in collaboration with Lund University. Without the support from the team at Minut; Nils Mattisson, Marcus Ljungblad, Fredrik Ahlberg and Martin Lööf, as well as the supervisor Christian Antfolk at Lund University, this thesis would not be possible and we would like to thank you for your support and motivation.

Contents

Abstract i

Acknowledgements iii

Abbreviations ix

1 Introduction 1
1.1 Contributions . . . 2
1.2 Disposition . . . 2

2 Background 3
2.1 Counterfeit . . . 3
2.2 Point . . . 4
2.3 Requirements . . . 6

3 Theory 9
3.1 Production . . . 9
3.2 Existing Test Methods . . . 10
3.2.1 In-Circuit Testing . . . 10
3.2.2 AOI . . . 10
3.2.3 AXI . . . 11
3.2.4 Vectoral Imaging . . . 11
3.2.5 JTAG & Boundary-Scan . . . 12
3.2.6 Serial Wire Debug . . . 12
3.2.7 BIST . . . 13
3.2.8 Functional Testing . . . 13
3.3 Sensor Tests . . . 13
3.3.1 Microphone . . . 13
3.3.2 Speaker . . . 13
3.3.3 Golden Sensors . . . 14
3.4 Power Consumption . . . 14
3.4.1 Current Shunt . . . 14
3.4.2 Current Shunt With Amplifier . . . 14
3.4.3 Current Mirror . . . 15
3.4.4 Current Probe . . . 15
3.5 Data communication . . . 15
3.5.1 I^2C . . . 15
3.5.2 GPIO . . . 16

4 Method 17
4.1 Background 17
4.2 High Level Design 18
4.3 Current Supply And Measurement 19
4.3.1 Current Measurement 19
4.3.2 Current Supply 21
4.4 Test Firmware 23
4.5 Sensor Specific Tests And Validation 27
4.5.1 Golden Sensor 27
4.5.2 Microphone And Speaker 30
4.5.3 Particle Sensor 32
4.5.4 Validation Based On Statistics 33
4.6 Test Fixture 34

5 Results 37
5.1 Golden Sensor 37
5.1.1 Temperature 37
5.1.2 Humidity 38
5.1.3 Pressure 40
5.2 Testing Of Points 42
5.2.1 Particle Sensor 46
5.2.2 Microphone And Speaker 47
5.2.3 Current 48

6 Discussion 51
6.1 Test Results 51
6.1.1 Golden Sensors 51
6.1.2 Testing Of Points 52
6.1.2.1 Particle Sensor 53
6.1.2.2 Microphone And Speaker 54
6.1.2.3 Current 54
6.2 Test Method Decisions 54
6.2.1 Microphone And Speaker 54
6.2.2 JTAG & SWD 55
6.3 Test System And Counterfeit 55
6.4 Error Factors 56
6.5 Further Development 57
6.5.1 Additional Components And Techniques 57
6.5.2 Image Analysis 57

7 Conclusions 59

References 63

List of Figures 66

List of Tables 67

A Current Supply And Measurement Schematic 69

B Does hardware counterfeit really exist? 71

C Problems and solutions **73**
C.1 Timing measurements 73
C.2 Working with software and hardware 73

D Test Firmware Code **75**
D.1 Sensor Example Code 75
D.2 ADC Code 76
D.3 I^2C Code 78

Abbreviations

ADC	Analog to Digital Converter.
AOI	Automated Optical Inspection.
BIST	Built-In Selt-Test.
DMA	Direct Memory Access.
DUT	Device Under Test.
FFT	Fast Fourier Transform.
FW	FirmWare.
GPIO	General Purpose Input Output.
GS	Golden Sensor.
IC	Integrated Circuit.
ICT	In-Circuit Test.
IR	Infra Red.
LED	Light Emitting Diode.
LGA	Land Grid Array.
MCU	MicroController Unit.
MEMS	MicroElectroMechanical Systems.
PCB	Printed Circuit Board.
PCBA	Printed Circuit Board Assembly.
PPTC	Polymeric Positive Temperature Coefficient.
PRS	Peripheral Reflex System.
PSU	Power Supply Unit.
PWM	Pulse Width Modulation.
RAM	Random Access Memory.
SUT	Sensor Under Test.
SWD	Serial Wire Debug.

1. Introduction

Counterfeit products have troubled manufacturers for a long time. The problem is common in the clothing and accessories industry. Something less obvious and more unknown is counterfeiting of electronic components. In the semiconductor industry, an estimated $3 billion worth of components is counterfeited worldwide [1]. Just like with counterfeit clothes it hurts brands and companies. Counterfeit components may have poor performance [2] or behave as expected in the beginning but change behavior later on [3]. Both cases can cause malfunctioning products. From a system-integrator standpoint it is imperative to deploy reliable quality-assurance systems. Integrated Circuit (IC) manufacturers also have an interest in solving the problem to increase revenue. Individual hardware companies are interested in designing and applying anti-counterfeiting techniques in production to reduce the risk of a malfunctioning customer product. The number of hardware oriented start-ups is growing and hardware manufacturing is becoming more approachable [4]. The manufacturing industry is also increasingly interested in smaller tech-companies [5].

Big corporations and semiconductor manufacturers are working on various standards and authentication techniques to prevent counterfeiting of components [1][6]. It is important that individual companies and factories follow these initiatives and implement anti-counterfeit systems in the manufacturing process. Without counteraction, the problem will grow and result in damage to company value [7] and decrease of reliability, as seen in military systems [2].

This thesis develops anti-counterfeit techniques for an assembly production line. It is done by extending existing techniques in hardware testing with the purpose to authenticate and validate the quality of the components used in production. The result is a test system focused on distinguishing counterfeit components by component performance validation.

1.1 Contributions

The thesis resulted in a test system designed to authenticate every component in Point which is a WiFi connected house monitoring device. This is done through validation of test data against component characteristics with the purpose of finding erroneous or counterfeit components. The development of the test system contributes to the field by:

- Helping the electronics industry to prevent counterfeiting of electrical components. Testing products and identifying counterfeit components prevents compromised products to reach consumers.
- Presenting methods enabling small hardware companies to contribute in fighting counterfeit goods by establishing practices suitable for small companies. The developed test system can validate components' functionality and confirm that tested parts adhere to manufacturers specifications.
- Expanding the existing production test-suite to assure the quality of every shipped product. Testing all parts of the product to deliver a product that works as expected without counterfeit components.

1.2 Disposition

In the above introduction the problem and purpose is addressed. Chapter 2 will further explain the counterfeiting and piracy problem. The chapter will also introduce the product, the company where this thesis is written, and present the pre-study with system requirements. Chapter 3 lists and explains current test methods and common production errors. Chapter 4 explains how the test station is designed, implemented and describes the validation process. Chapter 5 presents the collected test data. In Chapter 6 the performance of the test-system is assessed and reconnected with the counterfeit problem. In Chapter 7 the conclusions are presented.

2. Background

This chapter will explain the situation further. It contains information about the company,

2.1 Counterfeit

Counterfeit electronic components is a problem [1][8]. With complex supply chains companies are getting more exposed to counterfeit components. The appearance of these components can be divided in two categories, the first category being misrepresented parts. This category includes activities such as relabeling legitimate components at higher grades or illegitimately replicating components and deceptively selling them as originals. The second category is old parts sold as new which involves selling previously used, recycled parts or defective components scrapped by the original manufacturer [6]. The diverse ways counterfeit components appears makes it a complex problem to solve. To counteract the counterfeit problem the industry needs to approach it at different levels.

Work is being done [1][9][10] in setting up standards for testing and authenticating techniques to prevent counterfeiting. Big companies and IC manufacturers try to address the problem, but to completely eliminate it something has to be done in every part of the industry [1]. For a small company designing and producing customer electronics it is not obvious how to manage counterfeit. Good relations with the factory manufacturing and assembling its products is one step. Careful sourcing of components with accepted suppliers is another. It is difficult and requires time and money to dig deep down in a complex supply chain.

Another measure against counterfeiting is to investigate the components used in the manufacturing process. By testing components and assembled products in production, a measurement of quality can be established for every component and unit. If every component can be identified and its quality determined, testing could be a way to prevent counterfeiting. Recycled components may be aged, showing reduced functionality [3], and illegitimately replicated or relabeled components may perform worse than specified [6]. If performance deviations and issues appear in the test results there is a chance the tested component may be counterfeit.

2.2 Point

This thesis is performed at the hardware startup Minut. Minut makes Point which is a battery powered, WiFi connected device designed to monitor houses and inform the owner about abnormalities. Point consists of a Printed Circuit Board (PCB) populated with a variety of sensors and two MicroController Units (MCUs). The Printed Circuit Board Assembly (PCBA) is covered by a protecting and aesthetically appealing plastic shell which also houses the batteries. Point is mounted in the ceiling or on a wall in the room using built-in magnets and a metal mounting plate. Figure 2.1 shows Point as it will be mounted in the ceiling and Figure 2.2 shows an exploded view of the product.

Figure 2.1: Point.

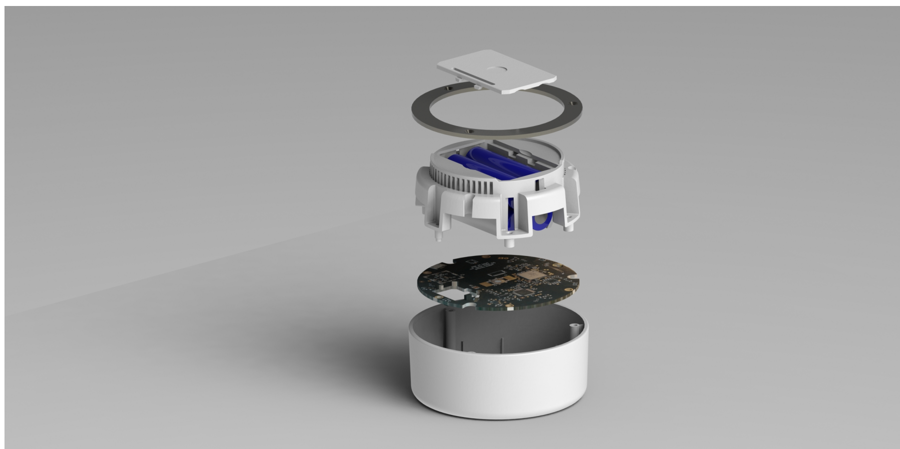


Figure 2.2: Point in exploded view.

Point uses an extensive array of sensors;

- Sensor for relative humidity
- Sensor for temperature
- Sensor for absolute pressure
- Particle sensor for measure air quality
- Microphone
- Hall effect sensor
- Speaker
- RGB Light Emitting Diodes (LEDs)
- Low-power data acquisition MCU
- High performance MCU with built-in WiFi

The overall electrical architecture is a combination of the ambient sensors, a low-power acquisition MCU and a high performance MCU with WiFi. During operation, the low-power MCU is responsible for data acquisition from the different sensors. All acquired data is analyzed in this low-power MCU and if a specified event is recognized this MCU wakes the WiFi MCU which will send the significant information to a back-end. The back-end will then notify the user about the event that occurred through a mobile app. Figure 2.3 shows an overview of the mentioned communication.

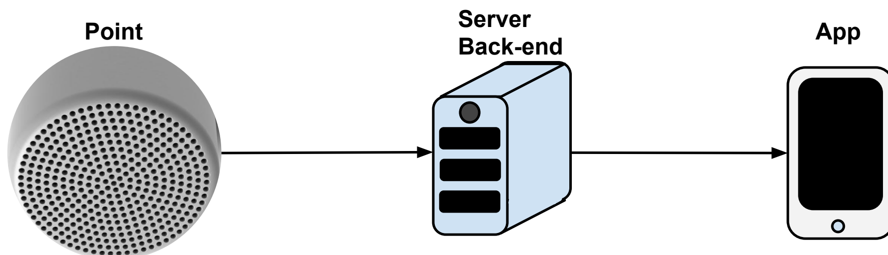


Figure 2.3: Overview of the communication between Point, the back-end and the app.

The low-power acquisition MCU takes care of all direct communication with the sensors and other peripheral ICs. Relative humidity, temperature, pressure and particle sensors are used to monitor the current conditions in the surrounding environment. The microphone is used to recognize specific sound events such as fire alarms or a window that breaks. Point, which is mounted on a metal plate using magnets, uses a Hall effect sensor to determine if the device is mounted or not, making it available to inform the user if it has been removed. The speaker and RGB LEDs is used for signaling events to people nearby.

The manufacturing of Point will be outsourced to a factory in mainland China. During the start-up of the manufacturing process, Minut will have staff on site in the factory to assist factory workers. When the manufacturing is tuned and works, operations will be controlled from Sweden. Due to the risk of counterfeit, Minut wish to be able to keep track of the production without the need of having personnel on site.

2.3 Requirements

Minut wants to assist in the work against counterfeiting. The company also needs testing during the manufacturing of Point. This thesis is about developing an anti-counterfeit test system that keeps track of the production of Point and validates the functionality of every produced unit. There are some additional requirements to the system as well. It needs to be supervised from Sweden.

The thesis will address the counterfeit problem by performing testing during the production of Point. A specific test station will be developed and implemented. The focus for the test station is to test the individual sensors listed in section 2.2, which are the most crucial parts for Point to perform as designed. The test station will be a complete test system controlled by a computer. The station will be placed in the middle of the manufacturing process, after the PCB is populated with components, but before the mechanical assembly process where the plastic parts and PCBA are fitted together. At this stage the electronic parts are supposed to be fully functional but consists only of a bare PCBA. This makes it easy to access all components and it is also the earliest stage where the electronic functionality of the unit can be tested. The test computer will load and control specific test FirmWare (FW) image onto the MCUs on each PCBA. The test process in general is described in Figure 2.4.

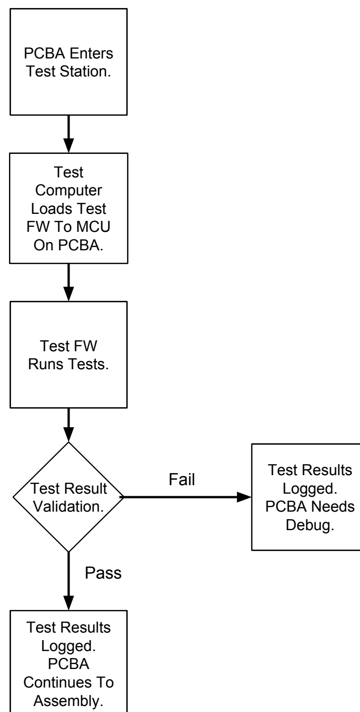


Figure 2.4: Basic flow of the test station.

Manufacturing electronics comes with more problems than just counterfeit components. Testing in electronics production is today an explored yet growing field. A multitude of tests at different stages during manufacturing have been developed. The test station implemented in the thesis will extend existing test methods in production to find malfunctioning components.

The test station also needs to power the PCBA during the tests. Since Point is a battery

powered device, power consumption is critical and needs to be considered. Hence the power consumption will be measured during the tests. High power consumption can indicate shorts or components out of specifications.

The computer controlling the test station will be connected to Minut's servers. For every PCBA entering the test station, details will be logged and stored on the servers. The log will contain detailed information about test results, PCBA ID, batch number, production date and component batches used in the produced unit. Logging all test results and pushing them to the company's servers makes them accessible for staff in Sweden. Production and test history is also stored enabling statistical evaluation.

3. Theory

The theory contains information about numerous existing test methods and basic introduction to the manufacturing process. It also explains data communication methods used in the project.

3.1 Production

Delivering a product to market involves many steps. Raw material is needed to manufacture the basic components which the final product will consist of. These basic components may then be parts building up other components in multiple steps before the final product can be assembled. The last step is for the product to be delivered to the customer, either directly from assembly or through distributors. Figure 3.1 shows an overview of a possible supply chain.

Due to weaknesses in inventory management, record keeping, inspection and testing protocols for example [2]. The multitude of steps in production with many suppliers makes it difficult for companies to trace counterfeit components all the way back to the supplier.

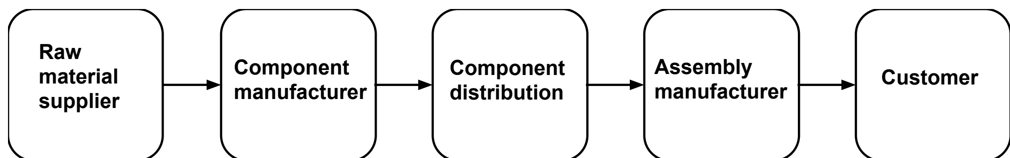


Figure 3.1: Supply chain.

Point's major pieces are a PCBA and a plastic cover holding it all together. Before the PCBA is completed it undergoes a couple of steps as shown in Figure 3.2.

The first step towards a working device is to manufacture a PCB without any components.

After passing tests, the plain PCB is reflowed in which the PCB is inhabited by the desired components. Using soldering paste (a sticky mixture of flux and powdered solder), the components are attached to their contact pads. When all components are attached to the PCB, it is subjected to heat, melting the soldering paste and permanently connecting the joints. After the reflow is completed, the PCB is now called PCBA. With the components attached, the PCBA is again tested to verify that everything works as expected. This is the part of manufacturing where most of the tests are implemented in this thesis.

After testing PCBA, it is now time for final assembly. Before the product can leave the manufacturing line, it is tested fully assembled a final time, verifying normal usage functionality. If the product manage to go through all production steps without failing any tests, it is considered fully functional and free of counterfeit components.

It is nearly impossible to know all different types of errors a product can suffer from. Known errors can be tested thoroughly to guarantee that specific entry points are working. A product ready for shipping may not be guaranteed to have no counterfeit components.

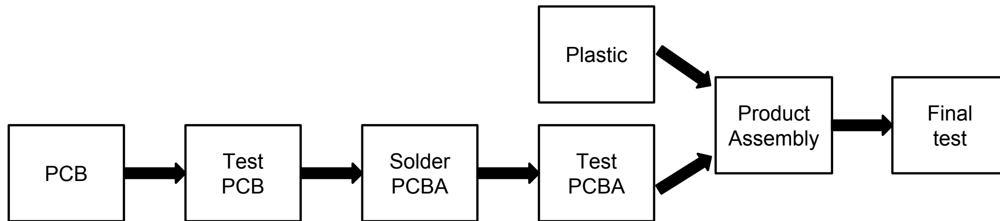


Figure 3.2: Production overview.

3.2 Existing Test Methods

There are various methods for testing hardware. Thanks to being based on different techniques they have unique test coverage of different types of errors. Many of the techniques overlap. Figure 4.1 gives an idea of some of the popular and common test methods and their coverage. Existing methods ranges from basic component tests for measuring component values and polarity, to the more advanced ones like JTAG and automated x-ray inspection.

3.2.1 In-Circuit Testing

In-Circuit Testing (ICT) is a common method during manufacturing, examining problems which arise from the manufacturing of PCBAs. ICT equipment is usually constructed by two major parts, a fixture and a software. The fixture uses a bed of nails which connects the accessible nodes on the Device Under Test (DUT) to the software. The software then runs a test-suite checking the device for open and short circuits. ICT is able to measure the performance of passive components regardless of other components connected to them. By testing the placement and values of these components ICT easily finds basic problems. ICT also has the ability to test operational amplifiers but when it comes to advanced ICs it is uneconomical [11].

A drawback of ICT is that it may not be possible to connect to all nodes on the PCB (e.g. because of shielded components). Another disadvantage is that electrical components are getting smaller and smaller leading to ICT having a harder time connecting to the smallest components.

3.2.2 AOI

Automated Optical Inspection (AOI) is a key technique used during production to find different kinds of errors on the PCBA. Earlier, optical inspection was performed manually but since board size has become smaller and the density of components on a board has increased it is no longer an effective test method. When the optical inspection was performed manually, it was realized that it was not particularly effective as inspectors grew tired at their stations and poor and incorrect constructions were easily missed.

Nowadays, optical inspection uses one or more cameras in a test station taking pictures of the DUT and allowing software to analyze the pictures taken. AOI can be implemented in different stages of the production process including pre- and post-reflow. When implemented in the pre-reflow stage, AOI will inspect the PCB and check for visible faults. An example of this kind of fault is shown in Figure 3.3.

Automated optical inspection implemented at post-reflow inspects a PCBA and compares it to a golden sample with all components placed correctly. If the software of the AOI test station discovers that anything is wrong, the PCBA will fail the test. Automated optical inspection

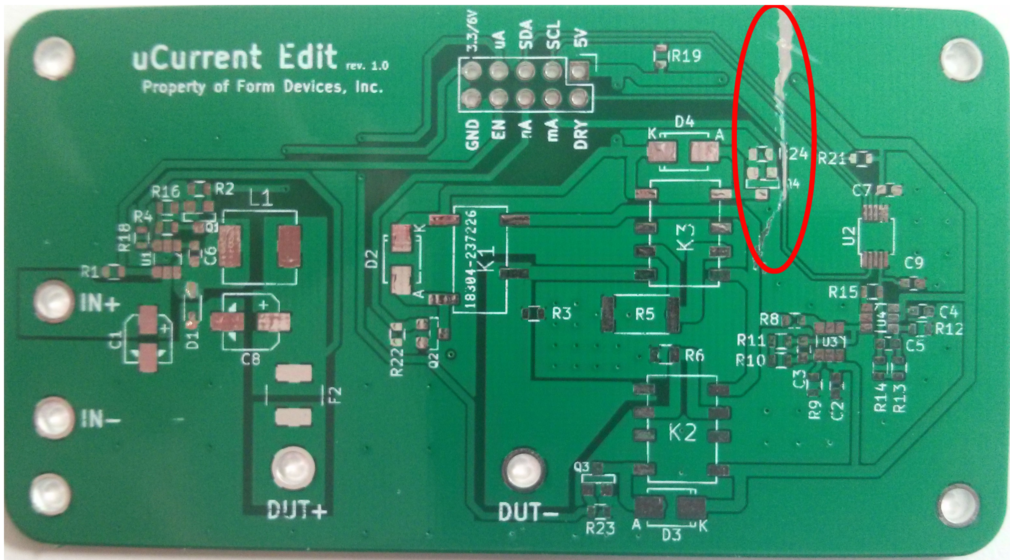


Figure 3.3: PCB with manufacturing faults.

is capable of finding manufacturing faults such as open circuits, short circuits, misaligned or misplaced components and missing components [12]. This results in fast and accurate testing for manufacturing faults. Thanks to being in an early stage of manufacturing, malfunctioning boards found during AOI saves money and time since it will not go further in the manufacturing chain.

AOI has a couple of disadvantages since it can not guarantee the functionality of ICs, values on passive components and soldering of Land Grid Array (LGA) components where the pins are directly underneath the component. It needs to be complemented by other test methods to find errors of these kinds.

3.2.3 AXI

Automated X-ray Inspection (AXI) is an improvement of AOI. Unlike automated optical inspection, AXI uses x-rays instead of a normal camera which allows it to detect soldering errors of LGA mounted components. Compared to AOI, automated x-ray inspection is a more expensive method but gives the opportunity to find soldering errors on all kinds of surface mounted components existing on the market. A balance has to be found between the price for AXI test stations and the need for inspecting LGA mounted components on the PCBA [13].

3.2.4 Vectoral Imaging

Vectoral Imaging is a new part of the optical tests trying to make sure the correct components is in the right places. Since the components are getting smaller and the density of the boards is getting higher the well known optical test methods have a harder time guaranteeing the correct test result. A new technology called Vectoral Imaging might be the right direction to handle the smaller components. Instead of using pixel based technology and absolute grayscale pixel values, the vectoral imaging is a pattern location search technology based on geometric feature extraction. Thanks to geometric features, vectoral imaging can manage components which changes in size and color which may exist due to manufacturing variations. Vectoral imaging

also eliminates background features on the PCB that may cause failures when using grayscale correlation techniques.

Vectoral imaging has another advantage as well, being able to use synthetic models of components removing the need of a golden sample PCBA before testing with vectoral imaging [14].

3.2.5 JTAG & Boundary-Scan

Boundary-scan, or JTAG **boundary-scan**, is a method for testing complex PCBs after assembly. It is defined by the IEEE 1149.1 standard, developed by the Joint Test Action Group, JTAG [15]. A JTAG enabled device (e.g. specific ICs) contains built in dedicated test logic that conforms to the JTAG standard. Depending on the version of JTAG, it uses four or five pins to handle the communication [16].

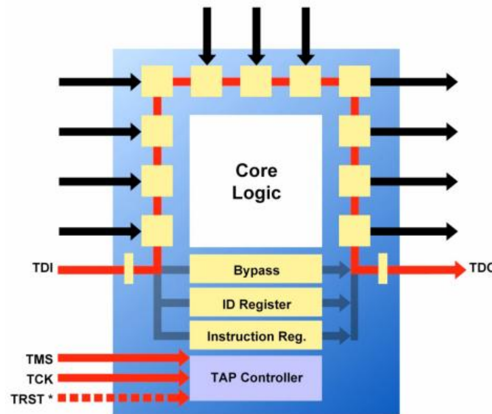


Figure 3.4: The logic of a JTAG enabled device. Reproduced from [17].

Boundary-scan is used to test the connection between devices. It can check that every component is correctly inserted and soldered to the PCB [17]. The method uses no physical test probes like ICT. Instead, every boundary-scan compatible device are connected in a chain. Figure 3.4 shows the internal boundary-scan logic in a device. Every separate device is chained through the TDI and TDO pins. The connection net on the PCB is considered to be boundary-scan testable if it can be driven and sensed by boundary-scan devices on the board. This is not only true between two boundary-scan compatible devices since not all devices needs to be equipped with boundary-scan to get good test coverage. There can be clusters of non-boundary-scan parts that will be testable despite the lack of direct boundary-scan access.

JTAG can also be used to program flash memory devices [18, p. 16, 18] as well as be used as a debug interface thanks to its connection to the ICs on the DUT [16].

3.2.6 Serial Wire Debug

JTAG was not invented for debugging, therefore other methods have been developed as alternatives to JTAG. One of these methods are Serial Wire Debug (SWD).

SWD is a low pin-count alternative to JTAG. It offers debug using only two pins, one clock pin and one single bidirectional data pin. SWD provides the normal JTAG debug and test functionality along with real-time access to the system memory [19]. JTAG was first intended as a component and board test interface. It is not ideal for debugging, which is a reason SWD was developed. SWD offers higher transfer rates and reduces the pin count compared to JTAG [16].

3.2.7 BIST

Built-In Self-Test (BIST) is a test method where the product runs a test on itself. The test is part of the firmware of the product and is usually run during start-up to examine that the device behaves as supposed. BIST does not only have to be run during start up but can also be a part of the operational functionality if the device will not be turned of by itself [20].

This testing method makes it possible to get direct feedback of devices with error if they fail the tests, provided the device being connected to a database containing test results. An advantage of BIST is also that it is possible to test product in the field making sure they still work after being used some time and after possible firmware updates. A disadvantage with BIST is that if some part of the device responsible for normal usage is not working, it is difficult to get feedback as the test itself may not be running.

3.2.8 Functional Testing

All the above test methods takes care of testing different parts of the product during assembly. To be sure that the complete product is working correctly and not only every part by themselves, functional testing is necessary. Functional testing of a product is a last test before it is being packed and shipped. It can be difficult to debug eventual errors and requires disassembling of the unit to fix or replace the defective part, which is why functional testing has to be complemented by other testing methods.

Functional testing is performed on the final, supposedly working product which has passed all previous tests during manufacturing. Performing functional test includes stepping through a test-suite executing and testing different functionality, checking that the overall functionality behaves as expected [21].

3.3 Sensor Tests

Existing test methods can judge whether or not a component is properly attached onto the PCB. JTAG and SWD can also test the specified functionality of an IC. To be able to do a more thorough test of every single sensor and verify its functionality, quality and measurement deviation, a more qualified and specific test system needs to be developed.

3.3.1 Microphone

Test of microphones are executed by measuring the frequency response of the microphone under test where the frequency response indicates if the microphone handles the sound source correctly or if it introduces coloration. One common method to measure frequency response is to place the microphone in an anechoic chamber and stimulate it with a frequency sweep produced by a speaker. The frequency response is recorded and compared to the response of a microphone with known technical data tested under same conditions [22]. If the comparison between the microphones are within the margin of error according to the specifications, the microphone is considered working as expected.

3.3.2 Speaker

Common end-of-line test stations for testing speakers consists of the device under test fixed in a clearly specified position in a rigid test box which provides some shielding against ambient noise. Inside the test box is a calibrated and specified microphone which records the sound produced by the speaker. Test software records the input from the in-box microphone and the

output to the speaker. It then compares the result according to specifications of the speaker, resulting in a passed or failed tests. The system may also comprise another microphone outside the test-box, recording ambient noise, giving the software possibility to reduce the impact of ambient noise in the test result [23].

3.3.3 Golden Sensors

A golden sensor is a sensor known to be working and calibrated to deliver the right measurements according to specifications. It works as a reference in a test case when determining the quality of the measurements acquired from the Sensor Under Test (SUT). The sensor under test acquires measurement values which are compared to the golden samples acquired from the golden sensor during the same time span. Measurements from a golden sensor is preferable to a static golden sample since the ambient environment varies. Using a static golden value may lead to failing tests when the ambient environment changes. Static golden samples may also pass malfunctioning DUTs if the ambient environment is not the same as the static golden samples and the DUT has some offset resulting in measurements matching to the golden sample.

3.4 Power Consumption

In a battery powered device like Point, low and controlled power consumption is an important aspect. During the design process of the hardware and software, power consumption is one of the main constraints and therefore it is important that every shipped device follows the designed power consumption parameters. In testing, power consumption may be a powerful tool as well.

If the DUT draws a higher current than expected during the circumstances there is a high probability that something is wrong, e.g. short circuit or erroneous hardware. There exist many different methods of measuring current. Some are simpler and some more advanced but better for specific cases [24].

3.4.1 Current Shunt

One of the simplest ways to measure current is to use a shunt resistor with known resistance in parallel with the load and measure the voltage drop over the resistor. Using Ohm's law the current flowing through the resistor can be computed. This method is very simple but not without drawbacks.

When the current increases, the voltage drop over the shunt resistor (called burden voltage) increases as well. In low voltage current measurements the burden voltage is undesirable due to the big impact it may have on the system. If the power supply of the system has an output voltage close to the operational voltage of the system a too high burden voltage may prevent the system from being able to function as intended. A possible solution is to use a smaller shunt resistor to minimize the interference with the system. Using a smaller resistances gives a lower voltage drop and therefore requires higher precision in the measurement [25][26].

3.4.2 Current Shunt With Amplifier

Another way to use the current shunt method is in combination with an amplifier. This method can improve the measurement precision. The amplifier may be used to reject common mode voltage as well as scaling the shunt resistor voltage. Depending on how the output voltage is measured in the end, different output ranges may be obtained. The output may be measured with an Analog to Digital Converter (ADC) which requires a specific input range or with a standard volt meter which may have a limited resolution or range. Regardless on how the output voltage

is measured, the method puts some demands on the amplifier used to get a high precision measurement. How the amplifier is designed and which type of amplifier to use depends on the specific characteristics of the system. If it is a high side measurement with high common mode, a differential amplifier may be used and if the measurement needs high precision and very specific range, an amplifier with low noise and offset is more important [25][27].

3.4.3 Current Mirror

This method may not be used on its own to measure current, but be part of a current measurement system in combination with current shunt. The idea is to duplicate the current consumed by the load and measure the duplicate with a current shunt. The advantage with this method is that the current shunt is absent in the direct supply line, thus there will not be a burden voltage from the current shunt in the direct supply line. The mirror may for example be a Wilson current mirror based on bipolar transistors. To reduce the total current drawn from the power supply, the gain of the mirror can be scaled to be less than one. The scaling is then taken into account for the final result [24].

3.4.4 Current Probe

Current probe is an indirect method to measure current in a conductor. The probe exploits the magnetic field around the conductor to measure the current flowing through it. This method is useful because it eliminates the need of cutting supply wires and put measurement instruments in series with the load. One of the drawbacks is that it can not be used to sense current flowing through PCB routes.

The current probe is not commonly used for current measurements of embedded systems, instead it is common when measuring high voltage and current systems because of its safer way to interfere with the system [24].

3.5 Data communication

3.5.1 I^{2}C

Inter-Integrated Circuit or I^{2}C is a bidirectional 2-wire bus for efficient inter-IC control developed by Philips Semiconductors (now NXP Semiconductors). I^{2}C uses two lines to carry the data on the bus, serial data (SDA) and serial clock (SCL). Both lines can be either HIGH or LOW and the different combinations of these states decides what is being transmitted on the bus. For the data to be transferred to the right IC on the system, each type of device has its own unique address. If there are two or more devices on the same system with identical address it is possible to change the address on some ICs. To get around the problem of multiple ICs not being able to change their addresses, there are I^{2}C multiplexers enabling using multiple ICs with the same address.

Each device using I^{2}C can operate either as a receiver or a transmitter, depending on the device. For example, a micro-controller can operate as both a transmitter and a receiver, not at same time, but an LCD-driver may only operate as a receiver. In addition to receivers and transmitters, devices connected to an I^{2}C bus can be configured as master or slave. The master controls the data flow and the slave responds to instructions transmitted from the master. A typical master is a micro-controller which runs the program and uses I^{2}C to communicate with peripheral slaves. Multi-master systems are also possible with I^{2}C, but the masters might end up competing. As soon as one of the masters has pulled SCL low, the other masters will behave as slaves until SCL is free [28].

Data transactions over the bus always starts with a START condition generated by the master. A transition from HIGH to LOW on the SDA while the SCL is HIGH defines the START condition. When the START condition has been sent, the bus is considered to be busy and has to be released by the master by sending a STOP condition. The STOP condition is defined by a LOW to HIGH transition on the SDA while the SCL is HIGH. After a STOP condition has been sent over the bus, it is considered free after a certain time. An overview of the I^2C protocol is shown in figure 3.5.

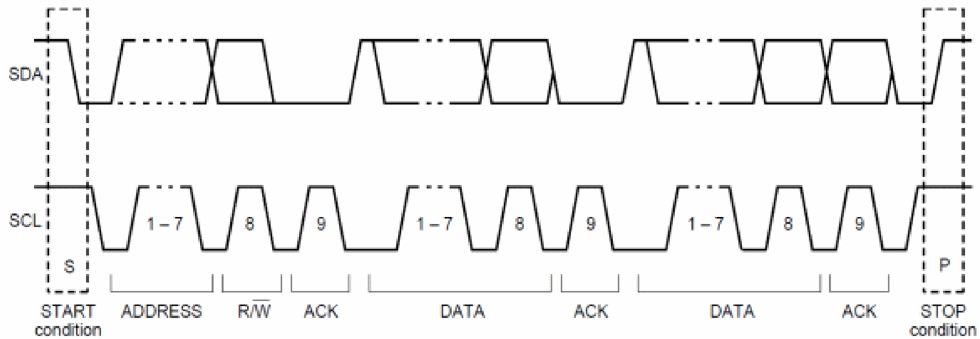


Figure 3.5: Overview of the I^2C communication protocol [29].

3.5.2 GPIO

General Purpose Input/Output or GPIO is a generic software-controlled pin on an IC whose behavior can be decided at run time. It is used for communicating between for example an MCU and an LED. The pin can be set either HIGH or LOW in the MCU.

GPIO is often used for interrupt handling in embedded systems or to control specific behavior in the IC [30].

4. Method

The method chapter explains how the test station is implemented. It starts with presenting a high level design of the system, and then explains in detail how the different parts are designed.

4.1 Background

The main focus will be on testing the array of sensors on Point. Other standard components also need to be tested on the production line. Current methods will be considered to cover the standard production failures. Since all these methods covers different failures a combination of them will be used. Which combination is a matter of what the factory is capable of performing, the additional price for the tests and the test coverage. An overview of different test methods can be found in section 3.2 and shown in figure 4.1.

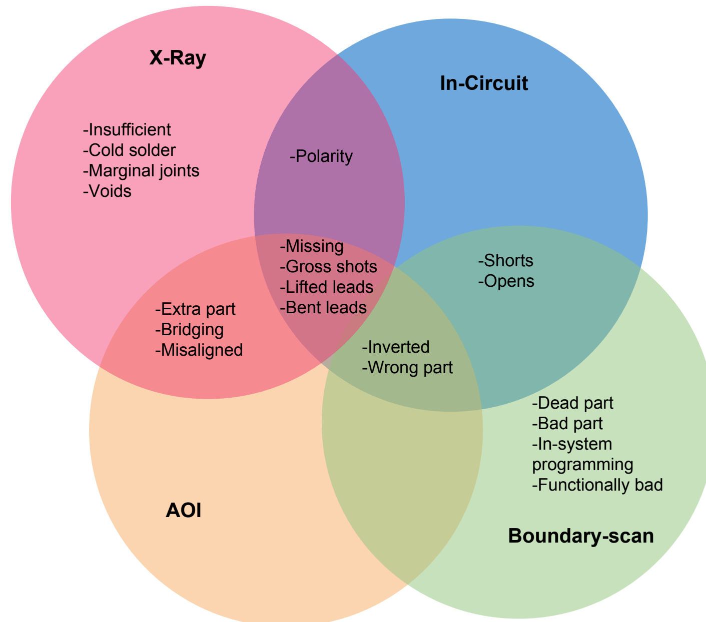


Figure 4.1: Test cover overview of some existing test methods [18].

In addition to existing test methods, the test station will cover specific components. The sensors needs custom test cases, the power consumption measurement generates specific demands and all tests will be designed with the counterfeit problem in mind. A first thought is to test the different sensors and components used on Point the same way they are tested during

their design process and manufacturing. The microphones parameters are tested thoroughly to make sure it works exactly as designed. Testing all sensors thoroughly will consume a lot of time during production. The environment in the factory may also limit the possibilities of doing thorough tests this way. The test system in this project is not aimed at verifying the design of a specific sensor, that is for the manufacturer of the sensor to test. Hence the design and the quality of the sensors used in the production is assumed to follow the specification from the manufacturers. The main focus for the sensor tests will instead be to verify that all sensors works according to their specification after assembly. The tests should also verify that the sensor placed on a unit is authentic and of correct type corresponding to the design. By testing the components this way the station will try distinguish counterfeit and erroneous components and prevent them to be shipped. This decision about how thorough to test results in spending less time researching information, and instead focus on how to test all sensors on the PCBA and in the assembled product.

4.2 High Level Design

To accomplish the desired sensor tests, a test station is designed. The station can be divided into different parts. The parts being a test fixture, a computer controlling the system, a unit for current measurements, a PCBA with golden sensors and two Power Supply Units (PSUs). The test fixture will be the main component inhabiting all other parts mentioned above, enabling easy mounting in the factory and and a simple test process of each DUT. A power management unit used to supply the DUT with current is designed, this unit will also manage the current measurements. It is connected and controlled via the test computer. A separate PCB is designed to house the golden sensor used in the validation process. This golden sensor PCBA is also connected and controlled by the test system computer. The two power supplies is used to power the test computer with 5 V and the current management unit powering the DUT with 12 V. An overview over the test system is shown in Figure 4.2

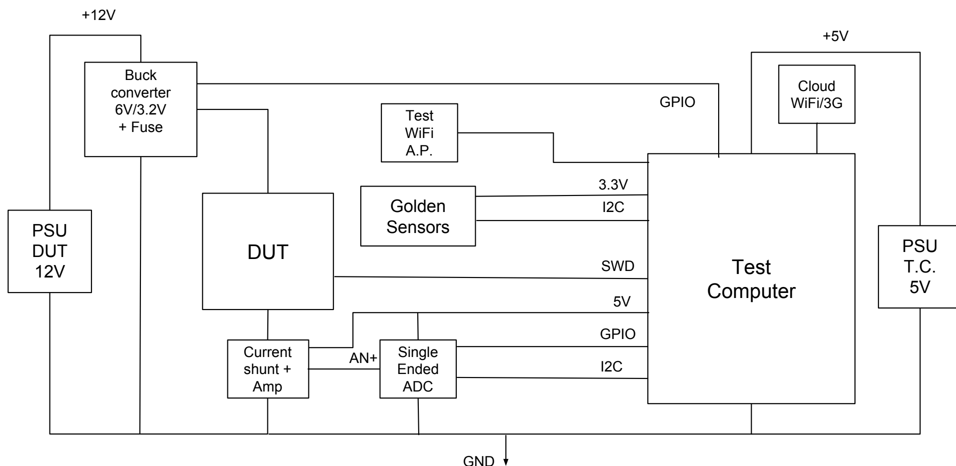


Figure 4.2: A block diagram over the test system.

The overall flow of the test system consists of some key processes. The first process executed when a new PCBA is placed in the test fixture is to supply it with power. The test computer powers the DUT and starts measuring the current usage. This process runs in parallel

during the complete test. When the DUT is powered, the test computer loads the test FW image to the low-power MCU on the DUT via Serial Wire Debug (SWD). When the test FW is completely loaded, it starts executing. While the DUT runs the test FW, the test computer collects test values from the golden sensor PCBA. The test computer then reads the test data from the DUT. As shown in Figure 4.2 the test computer communicates with the current management unit (the buck converter and current sens ADC block) over I^2C and General Purpose Input Output (GPIO). The golden sensor PCBA also uses the I^2C bus for communication.

4.3 Current Supply And Measurement

In order to supply the DUT with current while measuring the current usage a custom circuit was designed. The schematic is shown in figure 4.3. The design can be divided into two main parts, the left upper corner is for the power supply and the right upper corner is for the current measurements. An expanded figure is shown in Appendix A. Figure 4.2 shows the block diagram of the the complete test system. The current measurement system is placed on the low side, between the DUT and ground. The buck converter is powered with at separate 12 V power supply. The test computer and current data acquisition system is powered from a 5 V supply.

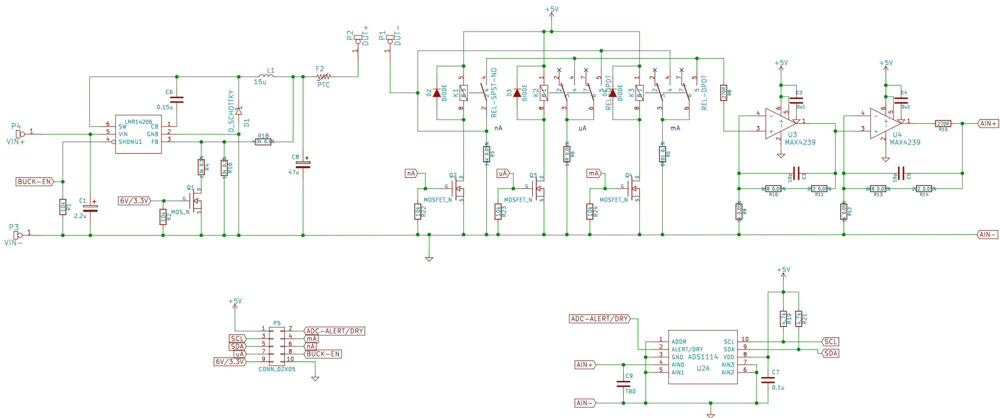


Figure 4.3: The full schematic of the power supply and current measurement PCB.

4.3.1 Current Measurement

Current measurements are performed using the shunt resistor method combined with an amplifier from Section 3.4.2. The amplifying step consists of two low drift, low noise amplifiers with a close to rail-to-rail output. The shunt resistance can be varied to reach good accuracy at different current ranges. Points power consumption varies from low currents of a couple of μA in low-power sleep mode, up to hundreds mA when the WiFi chip is transmitting. This means the input range of the current measurement system needs to be wide. To handle this, three separate shunt resistors is used together with three relays controlling which shunt is used. The shunt resistor values are optimized for the mA, μA and nA ranges. The value of the shunt resistor and gain of the amplifier is designed so that the output voltage is always directly translated to the mV range [25].

In figure 4.4 the design of the current measurement system is shown. The left half of the figure displays the shunt resistors R3, R5 and R6. They are controlled by three mechanical

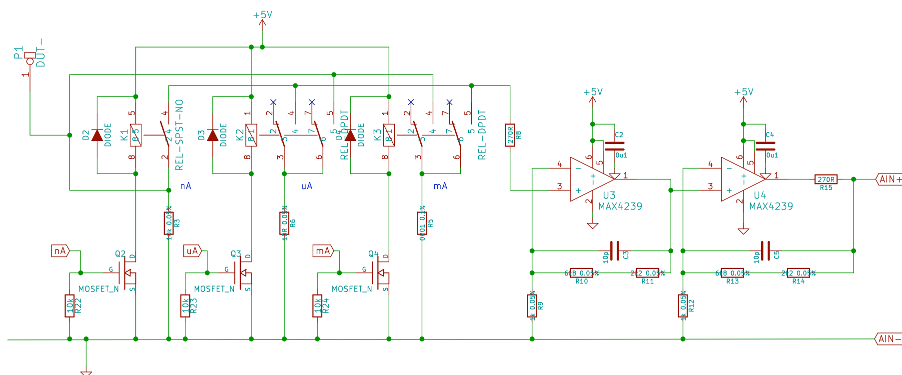


Figure 4.4: The shunt resistors; R3, R5 and R6 controlled by three relays to the left. Two amplifying steps to the right.

relays able to select which shunt resistor to use and which signal to pass to the amplifier. Each relay is controlled by an N-channel MOSFET switch. These switches can be controlled by normal 3.3 V signals from the test computer. Resistor R3 is always connected to ensure that the system is not left open circuit. Since the value of R3 is 10 kΩ, and the values of R6 and R5 is 10 Ω and 10 mΩ respectively, the error contribution of R3 is 0.1 % or less. The error formula below, equation 4.1, is for R6 and R3 in parallel.

$$i_{R6} = \frac{R3}{R6 + R3} \cdot i_{tot} = 0.999 \cdot i_{tot} \tag{4.1}$$

Where i_{tot} is the current to be measured. i_{tot} is assumed to be constant, but with R3 and R6 i parallel the total shunt resistance will be less than with just R6 resulting in a higher i_{tot} and an error less than 0.1 %.

The right half of figure 4.4 shows the amplifying steps consisting of the two low drift and low noise amplifiers. Both have negative feedback loops and a set gain of 10. The output of the amplifiers is the current flowing through the shunt resistor translated to voltage. The amplifying step has a total gain of 100. The combination of gain of 100 and a shunt resistance of 10 mΩ for the mA range gives a one to one ration between mA input and mV output. The three shunt resistors then scales in steps of times 1000. Hence, the 10 Ω shunt in the μA range gives a 1:1000 ratio, and the 10 kΩ shunt for the nA range gives a 1:1000000 ratio. In summary this gives that one mA translates to one mV, one μA to one mV and one nA to one mV. The amplifiers are fed with only +5 V resulting in a output range between 0-5000 mV. To sample the output voltage from the amplifiers an ADC is used. The ADC is a Texas Instruments ADS1114 Delta Sigma ADC controlled through I^2C. It has a resolution of 16 bits and an internal voltage reference of 4096 mV. The output from the amplifier is connected to the ADC in single ended mode which results in a final resolution of 15 bit and an input range between 0 V and +4096 mV. The three ranges of the system will therefore be able to measure currents between 0-4096 mA/μA/nA. The resolution will be

$$lsb = \frac{4096}{2^{15}} = 0.125mV \tag{4.2}$$

To get as accurate readings as possible the input range used needs to be optimized in relation to the current usage at the specific test. These optimizations is preliminary based on theory. In terms of current usage the testing can be divided into three parts. One is during deep sleep

mode, one when the acquisition MCU and sensors are active, and the last one is when the WiFi chip is active. Their current usage is shown in Table 4.1, Table 4.2 and Table 4.3.

Table 4.1: Low-power mode.

Device	Consumption
MCU	1 μA
Total	1 μA

Table 4.2: Mid power mode with sensors activated.

Device	Consumption
MCU	2.75 mA
Temperature and Humidity	2 μA
Pressure	25 μA
Microphone	150 μA
Particle sensor	9 μA
Total	2936 μA

Table 4.3: High power mode with WiFi activated.

Device	Consumption
MCU	2.75 mA
WiFi	400 mA
Total	402.75 mA

In Table 4.1, 4.2 and 4.3 only the main components are taken into account. Table 4.1 shows the consumption in deep sleep or low-power mode. The total amount fits in the nA range of the measurement system. Table 4.2 shows consumption during normal sensor activity. This mode is close to 3000 μA, but still below 4096 μA which is the limit of the μA range. This makes it possible to use the μA range when measuring current usage during normal sensor activity. During the tests not all sensors have to be active at the same time, which could give lower momentary current usage and even more space to measure in the μA range. At last Table 4.3 shows the current usage when the MCU and WiFi is fully active. This has to be measured in the mA range.

4.3.2 Current Supply

The power supply for the DUT needs to comply with certain requirements. The supply voltage needs to be variable to enable tests of the supply voltage boundary cases. Point is designed to run of a supply voltage between 3.2 V to 6 V. The power supply also needs to be able to deliver high enough currents. Table 4.3 shows that the current usage can peak at over 400 mA. In case of a full short the current supply also need to be current limiting.

Based on the requirements a buck converter is used to control the power supply of the DUT. The buck converter has a feedback network of resistors that regulates the output voltage. By varying the ratio between these resistors the output voltage can be changed. During normal testing, the DUT should be powered with 6 V. To test the lower boundary, 3.2 V should also be available as an output from the buck converter. This means the buck should be able to output

two different voltage levels, 6 V and 3.2 V. The buck converter used is a LMR14206 from Texas Instruments.

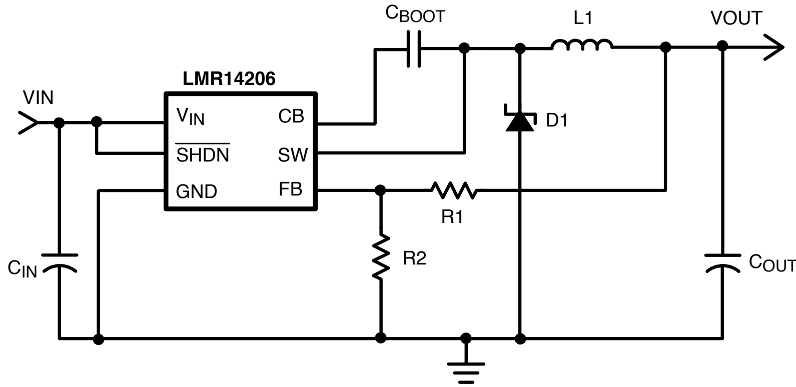


Figure 4.5: Texas instruments reference design of the LMR14206 buck converter [31].

Figure 4.5 shows the Texas Instruments reference design for the buck converter. FB is the Feedback Pin, where the feedback network is connected. R1 and R2 is the feedback network. The relation between the values of R1 and R2 and the output voltage is described by

$$V_{OUT} = 0.765 \cdot (1 + (R1/R2)) \tag{4.3}$$

provided by [31].

To be able to vary the ratio between R1 and R2 an extra resistance will be put in parallel with R2, and in series with a N-channel MOSFET switch connected to ground. The switch will select if the extra resistor should be active in the feedback network or not.

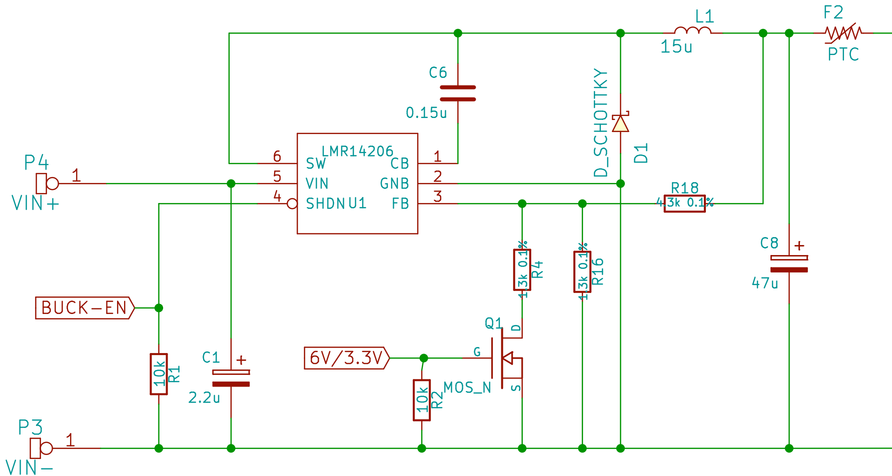


Figure 4.6: The scheme of the buck converter.

In Figure 4.6 R18 corresponds to R1 in the reference design and R16 to R2. R4 is the extra resistance added and Q1 is the switch controlling whether R4 should be active or not. With the Q1 switch off, only R18 and R16 will represent the feedback network. This results in an output voltage of

$$V_{OUT} = 0.765V \cdot (1 + \frac{4.3k\Omega}{1.3k\Omega}) = 3,295V \tag{4.4}$$

When the Q1 switch is on, R2 in the reference design will be the total value of R4 and R16 in parallel. The output voltage is now

$$V_{OUT} = 0.765V \cdot (1 + \frac{4.3k\Omega}{650\Omega}) = 5,826V \tag{4.5}$$

The values of R4, R16 and R18 are based on what is available on the market. This results in output voltages close to the desired output voltage levels. The buck converter is capable of delivering a current of 600 mA to cover the high usage presented in Table 4.3. It also has a built-in short circuit protection to protect the system from shorts [31]. A Polymeric Positive Temperature Coefficient (PPTC) is also added, shown in the top right corner of Figure 4.6, for extra safety in case of a full short.

The buck converter has a maximum duty cycle of 81 %. The duty cycle D is defined as

$$D = \frac{V_{OUT}}{V_{IN}} \tag{4.6}$$

provided by [31]

The maximal output from the buck converter is 5.826 V, this gives that the minimal input voltage to the buck has to be

$$V_{IN} = \frac{5.826V}{0.81} = 7.19V \tag{4.7}$$

An input voltage supply of 12 V hence gives a safe duty cycle.

4.4 Test Firmware

The low-power MCU on the DUT is responsible for sensor data acquisition in the test station. To do this a custom test firmware is flashed to the MCU on the DUT from the test computer via SWD. The test firmware is responsible for initialization of the MCU and sensors, trigger measurements and read measurement data. Figure 4.7 shows the most relevant source files of the test firmware and the basic relations between them. The complete firmware consist of more MCU specific source files than showed in the figure, but they are somewhat less relevant for the main purpose of the test firmware. The `sensor.h` file in Figure 4.7 represents the specific sensor files for the temperature, humidity, pressure and particle sensor.

In Figure 4.7 the main blocks of the test firmware are shown. The program starts in the `_start` method in `main.c`. In this method everything is initialized. All necessary internal peripherals of the MCU are initialized with the system clock. The GPIO pins on the MCU are set up in the correct mode, the ADC, Direct Memory Access (DMA) and timer is initialized, the real time clock used by the MCU is calibrated. When all setup is done the `test_main` method in `tests.h` is invoked as showed in Figure 4.8. This method controls the specific sensor tests.

Listing 4.1 shows how the code in `test_main` method is structured. At first, it powers all the sensors and waits 1000 ms for the power to stabilize and the sensors to boot. After the wait all measurements are triggered. At first the ADC is set to sample sound while the speaker plays a specific sine tone. This is done in the `mic_test` method. When the sound data acquisition is complete the specific pressure, humidity, temperature and particle sensor test methods gets invoked as shown in Listing 4.1. In each sensor test, initialization is done and test measurements gets triggered. While the sensor measurements is processed the MCU waits for data ready interrupts from the sensors. When a sensor interrupt is set, the interrupt

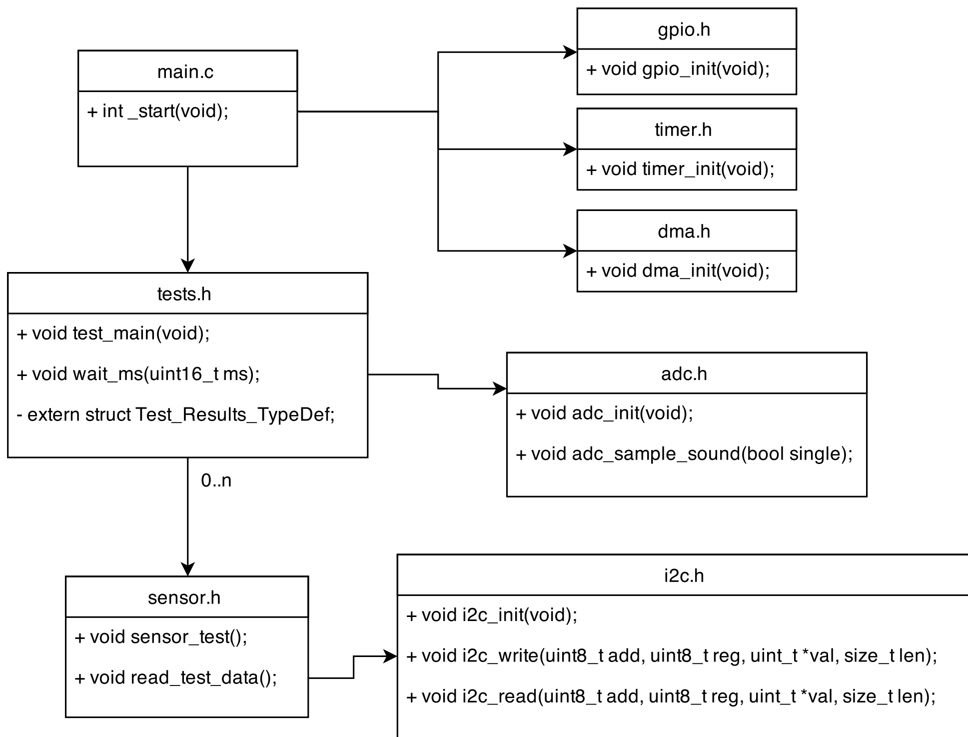


Figure 4.7: A block diagram over the most relevant files in the test firmware.

handler method sets the corresponding `sensortype_ready` boolean to true. When all five sensors have pulled their specific interrupts, the data is read from the output registers of all the sensors and saved in the `Test_Results_TypeDef` struct. After all test data is collected the `test_main` method returns to the `_start` method. As a last step, the test FW shuts down all sensors and internal peripherals to enter deep sleep mode. While in deep sleep, only the real time clock is active. After two seconds it triggers an interrupt to wake the MCU from deep sleep. When the MCU wakes the test is complete. All the test results are stored in the `Test_Results_TypeDef` struct showed in Listing 4.2. This struct is fetched over SWD by the test computer which validates the test results.

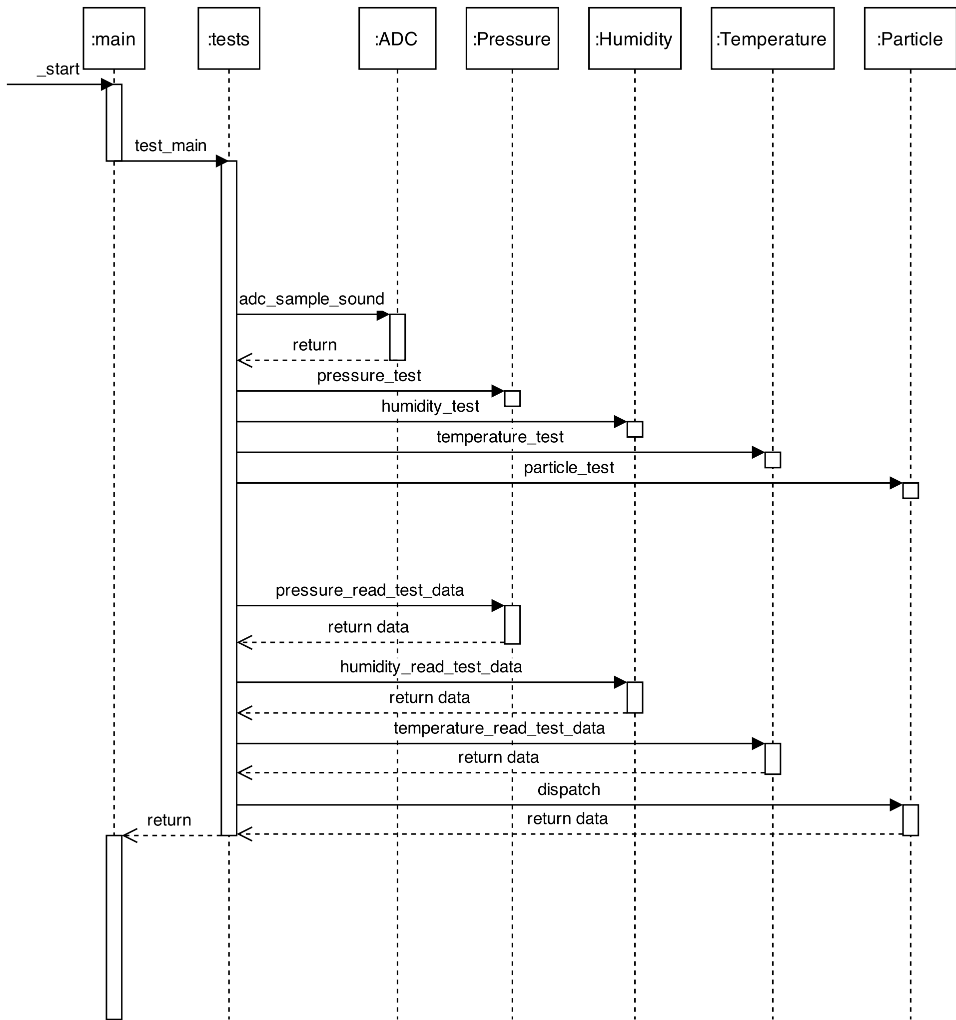


Figure 4.8: A flow chart of the test firmware.

```
struct Test_Results_TypeDef test_results;
static volatile bool temperature_ready, humidity_ready, pressure_ready, particle_ready;

void test_main(void)
{
    /* Set the gpio pins controlling sensor power low => sensor power on. */
    GPIO_PinOutClear(GPIO_PORT_SENSOR_PWRON, GPIO_PIN_SENSOR_PWRON);
    GPIO_PinOutClear(GPIO_PORT_BAR_PWRON, GPIO_PIN_BAR_PWRON);
    GPIO_PinOutClear(GPIO_PORT_SOUND_PWRON, GPIO_PIN_SOUND_PWRON);

    /* Wait 1000 ms */
    wait_ms(1000);

    /* Run test for microphone */
    mic_test();

    /* Run tests for pressure sensor */
    pressure_test();

    /* Run tests for humidity sensor */
    humidity_test();

    /* Run tests for temperature sensor */
    humidity_test();

    /* Run tests for particle sensor */
    particle_test();

    while(!temperature_ready || !humidity_ready || !pressure_ready || !pressure_ready);

    pressure_read_test_data();
    humidity_read_test_data();
    temperature_read_test_data();
    particle_read_test_data();
}
```

Listing 4.1: The `test_main` method.

```
typedef struct Test_Results_TypeDef
{
    bool temperature_correct_id;
    bool humidity_correct_id;
    bool pressure_correct_id;
    bool particle_correct_id;
    bool particle_measurement_error;
    float temperature;
    float humidity;
    float pressure;
    uint32_t sound_e;
    uint32_t sound_zcr;
    float sound_freq;
    uint16_t particle_ambient;
    uint16_t particle_led1;
    uint16_t particle_led2;
    uint16_t particle_led3;
};
```

Listing 4.2: The struct containing the test results.

4.5 Sensor Specific Tests And Validation

Due to the mixture of different sensors and components on Point, all tests needs to be tailored for specific purposes. Every sensor-specific test case should still aim for good test coverage. Step one is to confirm that the component under test is properly connected to the PCB. This will partly confirm the soldering process. Some solder failures may not show up until the PCBA has completely cooled off or passed some temperature cycles. When contact with the sensor is confirmed, the identification is checked to verify that the correct components are used. If the sensor is of correct type the test for validating functionality is set up and performed. Every sensor will give specific data which is read by the test system. The test data needs to be validated in order to complete the test. This is the definite step of the process where the actual decision is made. A variety of different sensors and tests requires a quite complex validation process. Every test is singularly validated but the total test process also needs to be considered. In the beginning the validation process will be based on theory. As new production batches are completed the test results from those can be used to influence the validation process for upcoming batches.

4.5.1 Golden Sensor

All sensors on Point tested with the golden sensor model are digital with built in logic. The logic is controlled from read- and/or write-able registers which can be accessed over the I^2C bus. By setting these registers the behavior of the sensor can be controlled. The registers also contains measurement results and specific sensor information. A sensor test in general is composed of initialization and configuration of registers to trigger a measurement and reading the result when the measurement is complete. The results are then validated with the help of golden sensor values. An example of sensor specific test code is shown in Appendix D.1. The first step for a specific sensor test is to read and verify the ID from the sensor under test. The ID returned from the sensor should match the ID specified in that sensor's data sheet. This introductory check gives various useful information; it tells if the sensor is properly connected to the PCB, if the sensor boots and works correctly, if the I^2C bus between the sensor and MCU works and if the sensor used has the correct ID. Following the ID check is the initialization process of the sensor's configuration registers. The registers are set to generate the desired measurement. This step is completed with a final write to the sensor triggering the measurement. All sensors can be set up to generate an interrupt when an acquisition is completed and a new result can be read from the output registers. Since I^2C is a master-slave bus and does not have support for interrupt handling, the interrupts are triggered on a separate communication line between the sensor and MCU. When an interrupt is generated from the sensor the MCU reads the measurement output registers. The final process is to validate the test results. For the golden sensor method this means determining if the sensor's test value is sufficiently close to the golden sensor value. The ideal way is to compare the sensor test value to the actual, or true, value in the relevant physical quantity. But the true value is not known. Instead the measured value in combination with the possible deviation or margin of error from the true value is used together with the value measured with the golden sensor on the test station. If both the golden sensor and the sensor under test has functions describing their measured values possible deviation from the true value, these functions can be used to determine if a test value should pass the test or not. Each function depends on the specified measurement parameters for the SUT and Golden Sensor (GS). These parameters can vary between different sensors. Figure 4.9 shows an example of parameters for a humidity sensor.

In the case of the sensor from Figure 4.9 the function would depend on the accuracy, repeatability/noise, temperature drift and hysteresis. This is not always the case. For some sensors

Table 4. Humidity Sensor

1.9 ≤ V_{DD} ≤ 3.6 V; T_A = 30 °C; default conversion time unless otherwise noted.

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
Operating Range[1]		Non-condensing	0	—	100	%RH
Accuracy[2, 3]		0 – 80% RH	—	±4	±5	%RH
		80 – 100% RH	See Figure 2.			
Repeatability/Noise		12-bit resolution	—	0.025	—	%RH RMS
		11-bit resolution	—	0.05	—	
		10-bit resolution	—	0.1	—	
		8-bit resolution	—	0.2	—	
Response Time[4]	$\tau_{63\%}$	1 m/s airflow, with cover	—	18	—	S
		1 m/s airflow, without cover	—	17	—	
Drift vs. Temperature			—	0.05	—	%RH/°C
Hysteresis			—	±1	—	%RH
Long Term Stability[3]			—	≤ 0.25	—	%RH/yr

Notes:

1. Recommended humidity operating range is 20% to 80% RH (non-condensing) over –10 °C to 60 °C. Prolonged operation beyond these ranges may result in a shift of sensor reading, with slow recovery time.
2. Excludes hysteresis, long term drift, and certain other factors and is applicable to non-condensing environments only. See section “4.1. Relative Humidity Sensor Accuracy” for more details.
3. Drift due to aging effects at typical room conditions of 30 °C and 30% to 50% RH. May be impacted by dust, vaporized solvents or other contaminants, e.g., out-gassing tapes, adhesives, packaging materials, etc. See section “4.7. Long Term Drift/Aging” .
4. Response time to a step change in RH. Time for the RH output to change by 63% of the total RH change.

Figure 4.9: Parameters for the Silicon Labs Si7006-A20 humidity sensor [32].

only a few of these parameters are specified. For the general case, a measurement deviation function for a sensor can be described as a function of a vector containing all the specified parameters and the true value z of the measured physical quantity for that sensor

$$f_{dev} = f(X, z), \quad X = [x_1, x_2 ... x_n] \tag{4.8}$$

where X contains all the n specified parameters. When f_{dev} is given for the golden sensor and the SUT the following step is to determine whether the sensor values with their specific margin of error should pass. Both the golden sensor and the SUT have an f_{dev} describing their possible deviation relative the true value. Since the true value is not known, the decision needs to be based on the deviation intervals. If both the interval for the GS and the SUT could contain the same true value, the SUT should pass. Figure 4.10 is a simple example of vaildation of a temperature sensor. f_{dev} for both the GS and the SUT is assumed to have linear upper and lower boundaries.

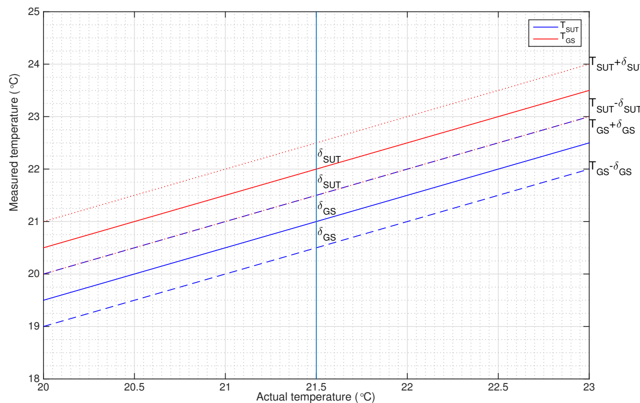


Figure 4.10: Possible test value deviation in a temperature test.

Let the maximal error deviation from the true value at a specific point be denoted δ. In Figure 4.10 the maximal deviation from the true value for the GS and SUT at a point is denoted $\pm\delta_{GS}$ and $\pm\delta_{SUT}$. The true temperature value T_{true} relative to the SUT value is supposed to be in the interval

$$T_{SUT} - \delta_{SUT} \leq T_{true} \geq T_{SUT} + \delta_{SUT} \tag{4.9}$$

And corresponding for the GS interval. For the test to pass, both the GS and SUT intervals needs to possibly cover the shared true value. Hence, the size of the intersection set of the two intervals needs to be greater than zero.

$$|T_{SUT} \pm \delta_{SUT} \cap T_{GS} \pm \delta_{GS}| > 0 \tag{4.10}$$

Figure 4.10 shows a special case where the intersection only contains the boundary values $T_{SUT} - \delta_{SUT}$ and $T_{GS} + \delta_{GS}$. This will be one of the boundary cases where the test should pass. Above statements applies to the general case as well. If the size of the intersection between the deviation intervals is grater than zero, the test should pass. Or maybe simpler, if the distance between the GS value and the SUT value is less than or equal to the sum of the maximal deviation of the two sensors

$$|Val_{SUT} - Val_{GS}| \leq \delta_{GS} + \delta_{SUT} \tag{4.11}$$

where Val_{SUT} and Val_{GS} are the measured test values from the SUT and GS. In the case of this test system, the golden sensors will be of the same type as the sensors under test. This means that f_{dev} will be the same for both the GS and SUT. The three components tested with the golden sensor method are the temperature- , pressure- and humidity sensor. For the three sensors only the accuracy a is specified, which gives that f_{dev} for them is just the accuracy as a function of the true value in the measured physical quantity of the sensor. The accuracy is specified in different intervals with two levels of accuracy depending on the true value in the measured physical quantity. One lower accuracy interval covering the full sensor output range, and one higher accuracy interval covering a smaller range. The high accuracy range is a subset of the low accuracy range. f_{dev} can be described as shown in Equation 4.12 below.

$$f_{dev}(a, z) = \begin{cases} \pm a_1 & \text{if } z_{lowMin} \leq z_{highMin} \leq z \leq z_{highMax} \leq z_{lowMax} \\ \pm a_2 & \text{if } z_{lowMin} \leq z \leq z_{lowMax} \end{cases} \tag{4.12}$$

f_{dev} of the pressure sensor only depends on the ambient temperature according to Equation 4.13 shown below.

$$f_{dev}(T) = \begin{cases} \pm 0.2hPa & \text{if } 20 \leq T \leq 60°C \\ \pm 1hPa & \text{if } 0 \leq T \leq 80°C \end{cases} \tag{4.13}$$

The temperature sensor's f_{dev} only depends on the ambient temperature as well, shown in Equation 4.14 below.

$$f_{dev}(T) = \begin{cases} \pm 0.5°C & \text{if } 15 \leq T \leq 40°C \\ \pm 1°C & \text{if } 0 \leq T \leq 60°C \end{cases} \tag{4.14}$$

For the humidity senor, f_{dev} only depends on the relative humidity according to Equation 4.15.

$$f_{dev}(H) = \begin{cases} \pm 4.5\%rH & \text{if } 20 \leq H \leq 80\%rH \\ \pm 6\%rH & \text{if } 0 \leq H \leq 100\%rH \end{cases} \tag{4.15}$$

4.5.2 Microphone And Speaker

The microphone and speaker will be tested through a custom test. In Point the speaker is not soldered directly to the PCB but instead mounted with springs in the final product assembly. Even if the PCBA does not contain the speaker, it will contain the amplifier driving it. The amplifier needs to be tested. To minimize the dependency of external test parts, the amplifier on the DUT will test the microphone on the DUT with the use of only an external speaker mounted on the test station. This will test the amplifier at the same time. To power the external speaker pogo pins will be placed from the output of the amplifier and with the other end connected to the external speaker. The external speaker will be placed on the test station directly below the microphone on the DUT. During the test sequence, the amplifier and speaker will generate sine signals with known frequency and amplitude. Meanwhile, the microphone listens to the signals and the MCU analyzes the microphone samples to verify that the input have the correct frequency and amplitude. The goal is to test the specified boundary frequencies of the microphone in Point. Due to the surrounding environment and the capability of the speaker the absolute boundary frequencies may not be testable. Therefore the test frequencies used in the test also needs to be based on the capability of the speaker and optimized for the surrounding environment.

Point uses a class-D amplifier to generate the speaker signal. The MCU generates a Pulse Width Modulation (PWM) signal as input to the switching amplifier which in turn feeds the speaker with its output. The design of the amplifier eliminates the need of an output filter if an inductive transducer is used. For simplicity that will be used in Point and also in the test station. The sound sensing unit on Point is a MicroElectroMechanical Systems (MEMS) microphone in combination with an amplifier. To complete the acquisition of sound data the internal ADC of the MCU is used. It has a resolution of 12-bits and a configurable sample clock. The block diagram of the microphone is shown in Figure 4.11

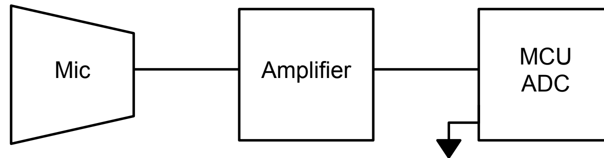


Figure 4.11: The block diagram of the microphone. The amplifier adds half VDD as bias to the amplified signal. The input to the ADC is single ended.

The internal setup of the acquisition in the MCU is a combination of the ADC, DMA, a timer and the Peripheral Reflex System (PRS). Figure 4.12 shows how the internal blocks interacts during the data acquisition of sound.

The timer is responsible for triggering every new ADC sample. It is set up to generate an interrupt every time it overflows. To achieve the correct sampling frequency, F_s, the timer needs to overflow with the same frequency. The clock frequency of the timer is f_{timer}, and the value v_{of} where the timer should overflow to achieve the sampling frequency F_s is

$$v_{of} = \frac{f_{timer}}{F_s} \tag{4.16}$$

When the timer overflows it generates an interrupt, the PRS will notice this and trigger the ADC to begin sampling a new value. When a new sample is available from the ADC, the DMA

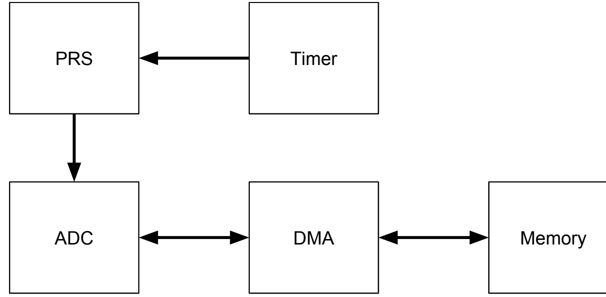


Figure 4.12: The internal MCU setup for the data acquisition.

is notified and moves the new sample to an empty slot in the Random Access Memory (RAM). The DMA is set to move a specified number of ADC samples. It is the number of DMA cycles that determines the length of the sound data acquisition. When the specified number of samples is stored in the RAM, the data acquisition is complete. The DMA will then stop and call a specified callback function. The timer clock is halted in the callback function. This stops the timer from generating new overflow interrupts to trigger new ADC samples, and the sampling process is completed. In the end of the callback function the DMA setup is cleared and all the sound samples are available in the RAM [33]. The detail code for the ADC and sound data acquisition is listed in Appendix D.2.

During the test sequence one frequency is tested at a time. The internal amplifier of the DUT is used to feed a sine signal to the external speaker. The acquisition MCU is responsible for generating the PWM signal to the amplifier. While the speaker plays the tone, the internal microphone is used to sample the sound. Once the sound acquisition process for one of the sinus tones is completed the MCU on the DUT will analyze the sound samples to determine the frequency f and the average power of the sampled signal.

The frequency is computed by $f = \frac{1}{T}$. To find the period time the sample array is first searched for the number of zero crossings n_{zc} of the signal during the sampling time. When the number of zero crossings is found T is computed with the use of the sampling rate F_s and the number of samples N by

$$T = \frac{\frac{1}{F_s} \cdot N}{n_{zc}} \tag{4.17}$$

The average power P of the sampled signal is computed by Equation 4.18 is used [34]. N is the number of samples, and X is the array containing the sound samples.

$$P = \frac{1}{N} \sum_{m=0}^{N-1} |X[m]|^2 \tag{4.18}$$

The sound analysis method requires that the size of the sample array is long enough to fit at least one period of the sampled signal. It also requires the sample frequency to be at least twice the sound frequency due to the Nyquist Theorem. An alternative to doing the analysis locally on the DUT is to push all sound samples to the test computer and do the analysis from there. This makes it possible to use more advanced techniques and software due to a more powerful processor. For instance Fast Fourier Transform (FFT) tools in Python or MATLAB can be used to analyze the frequency content. FFT analysis is too demanding to work on the MCU on Point.

```
static inline uint32_t stazcr(int16_t *src, size_t len)
{
        uint32_t sum = 0;
        int32_t s, last = 1;

        while (len--) {
                s = *src++ >= 0 ? 1 : -1;
                sum += abs(s - last);
                last = s;
        }

        return sum / 2;
}
```

Listing 4.3: Algorithm to count the number of zero crossings in a array of sound samples.

```
static inline uint32_t ste(int16_t *src, size_t len)
{
        uint32_t sum = 0;
        size_t n = len;

        while (len--) {
                sum += *src * *src;
                src++;
        }

        return sum / n;
}
```

Listing 4.4: Algorithm to compute the average power from an array of sound samples.

Performance demands and transfer time needs to be considered if all microphone sample data is transferred to the test computer.

4.5.3 Particle Sensor

The particle sensor on Point is optic and consist of Infra Red (IR) LEDs and a photo diode. The tests for this sensor aims to verify that the LEDs are working and transmits the correct intensity of light, and that the photo diode is capable of registering both the ambient light and light from all of the IR LEDs. The test is executed in four separate measurements sampling the output value from the photo diode. At first the output from the photo diode with only ambient light as the source is sampled. The three following measurements are done with one LED active at a time for each measurement. This way of testing the particle sensor requires no external components on the test station.

The particle sensor has some built in logic. The test is designed like the golden sensor tests in Section 4.5.1. The same steps are performed; check ID, initialize the sensor and trigger a measurement, and finally read and validate the measurement results. To verify that an LED is working the light intensity captured by the photo diode in an LED measurement should be greater than the measured ambient light intensity. Let y_{amb} be the value registered in the am-

bient measurement. Let y_{led1}, y_{led2} and y_{led3} be the values registered when the three different LEDs are active. For all LEDs to be considered working y_{led1}, y_{led2} and y_{led3} needs to be greater than $y_{amb} + \alpha_{ledX}$,

$$y_{ledX} \geq y_{amb} + \alpha_{ledX} \tag{4.19}$$

where α_{ledX} is the minimal acceptable difference in light intensity between the ambient light and an LED for a test to pass. α_{ledX} needs to be greater than the maximal possible variation of intensity in the ambient light during the test

$$\alpha_{ledX} > \beta \tag{4.20}$$

where β is the maximal variation of light intensity for the ambient light during the test. The first level of the test is to determine if the LEDs are working at all. The next level is to determine if they are transmitting the correct intensity of light based on theory and specifications. This will need a model of the optic sensor with LEDs and the photo diode to start with. For a LED the transmitted intensity at a specific supply voltage and the distribution angle needs to be considered. For the photo diode the ability to register light and the bias voltage is relevant. In a complete model, the distance and angle between the LED and the photo diode needs to be taken into account.

4.5.4 Validation Based On Statistics

Before mass production of Point is started, some test production runs will be completed. This is with the purpose of testing and optimizing the production line in the factory. The early units is also used for beta testing of the product. The first test production run will be 25 units. Statistic parameters can be computed for these 25 units. Those parameters can then be used to predict the statistical parameters for the rest of the production. Later on during full production, every tested unit can contribute to the statistical model. The model can then be used in the validation process of test result.

The golden sensor tests for humidity, temperature and pressure is one case. All three sensors have a defined function, f_{dev} Section 4.5.1, telling how they deviate from the true value of their measured physical quantity. Assume that the true value for a physical quantity is constant during a time t, and that the golden sensor measuring that physical quantity runs N measurements during that time t. Then the output value from the sensor can be seen as the random variable X. The expected value $E[X]$ for X is the mean value μ of the N measurements. The variance is $Var(X) = \sigma^2 = E[X_2] - (E[X])^2$ and the standard deviation σ. Based on the theoretical f_{dev} specified in Section 4.5.1 the only thing varying when the true value changes should be the expected value for the measurements, while the standard deviation and variance should stay constant during the two intervals of f_{dev}. This means that the probability distribution function for the golden sensor should be constant except from varying center at the expected value.

For a test, the decision lies in whether to pass it or not. Since the true value for a physical quantity is unknown, this decision has to be based on the golden sensor value. The output of the test is thus based on two values, the SUT value and the GS value. If the difference between the SUT value and the GS value is assumed to be a random variable Y, then the statistical parameters of Y can be used to validate the test in the future. $E[Y]$ should optimally be 0, the SUT and GS should show the same temperature. But if $E[Y]$ differs from 0, there is an expected offset between the GS and SUT. If the GS and SUT is actually the same type of sensor, this could be interpreted as the GS values having a trueness offset explained in Figure 4.13.

If the trueness for a GS value is lower than expected this can result in failing SUT that actually should have passed. Hence, the offset $E[Y]$ should be taken into account when validating the test result for a sensor.

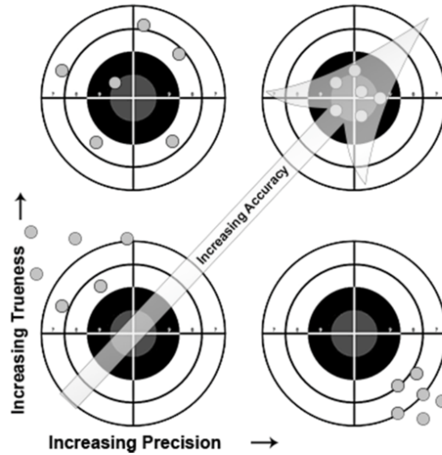


Figure 4.13: The relation between precision, trueness and accuracy. Reproduced from [35].

Statistics could be used for validation of the microphone and speaker test as well. The frequency computed from the microphone test sample data is a random variable Z. The expected value $E[Z]$ should be the frequency of the sine tone generated with the speaker. Here, the variance is $Var(Z) = \sigma^2$ and standard deviation σ can be used to determine if the sampled and computed frequency is correct or too off.

For the particle sensor test, all measurements has statistical parameters. The expected value of the measured intensity in the four measurement are quite complex to compute. Statistics from earlier measurements could be used for validation instead. The variance in ambient light is an important factor in the validation process. If every ambient light measurement in the factory contributes to the statistical model, this can be used to predict the maximal variation of ambient light intensity in the factory.

4.6 Test Fixture

To be able to run tests such as ICT and custom test-cases on the PCBAs a test fixture is usually designed. The test fixture consists of a robust metal or plastic body with a box where the external hardware is located. On top of the box is a lever making it possible to press the PCB down towards pogo pins situated on the box. The pogo pins are the essential part of the test fixture acting conductors between the test points on the DUT and the test computer. In the bottom end the pogo pins are connected to a PCB linked with the computer. Test fixtures usually have some buttons, LEDs or a simple user interface for starting and controlling the test-program and for feedback to the test operator presenting status if the device passed or failed the test. Test fixtures are often constructed to be as easily operated as possible and without language complication making it operable by any person.

Using a test fixture prevents the need for open debug ports in the consumer version of Point which is preferred of the company. The test fixture is constructed with the design files of the PCB and a physical version of the PCB as a reference. The initial version will have a button to start the test and LEDs as pass or fail feedback. It will also contain a small screen for debug output.

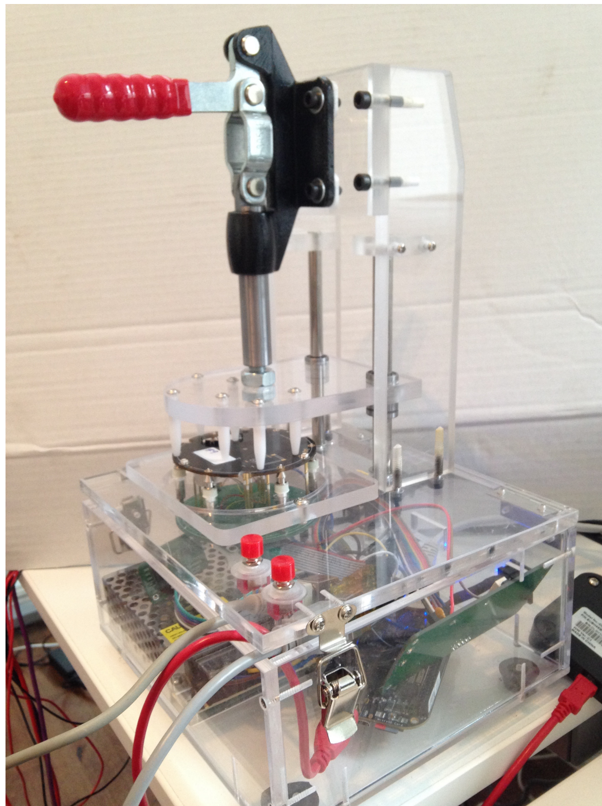


Figure 4.14: The test fixture filled with hardware.

5. Results

This chapter presents the tests results of 25 beta Points. It also shows measurements on how the golden sensors performs.

5.1 Golden Sensor

All measurements of the golden sensors below are acquired during 15 seconds. The ambient environment is attempted to be kept constant for all measurements.

5.1.1 Temperature

Figure 5.1 shows 1000 measurement values from the temperature sensor. As seen in the figure, the temperature values have some distinct levels due the resolution of the sensor. The red dotted line represents the expected value based on the values, the expected value is $\mu_t = 24.55°\text{C}$ and the standard deviation is $\sigma_t = 0.0294°\text{C}$.

The temperature values in form of a histogram is shown in Figure 5.2. A probability plot of the same values is plotted in Figure 5.3. The probability plot clearly shows the distinct levels in output from the sensor.

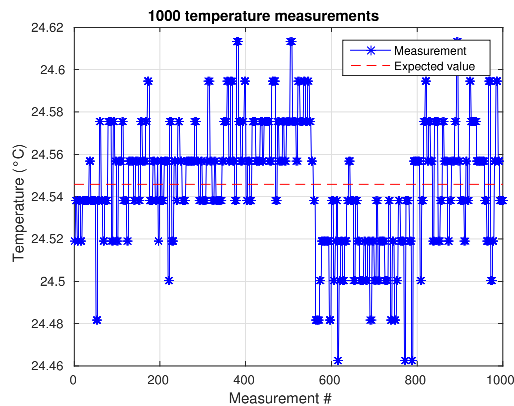


Figure 5.1: 1000 measurements acquired with golden temperature sensor.

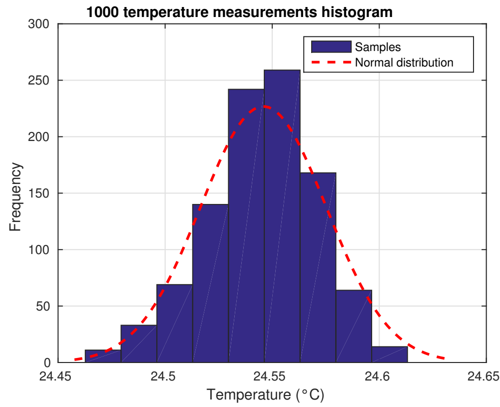


Figure 5.2: Histogram based on the temperature measurements.

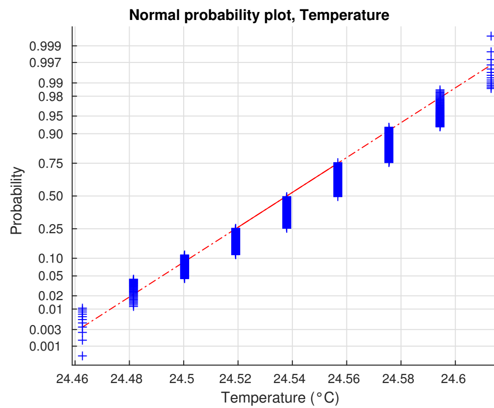


Figure 5.3: Probability plot of the temperature values.

5.1.2 Humidity

The 1000 measurement values acquired from the humidity sensor are shown in Figure 5.4. The dotted red line is the expected value based on the values. The expected value for the humidity is $\mu_h = 43.27\%\text{rH}$ and the standard deviation is $\sigma_h = 0.2975\%\text{rH}$.

Figure 5.5 shows a histogram of the humidity values and Figure 5.6 is a probability plot of the values.

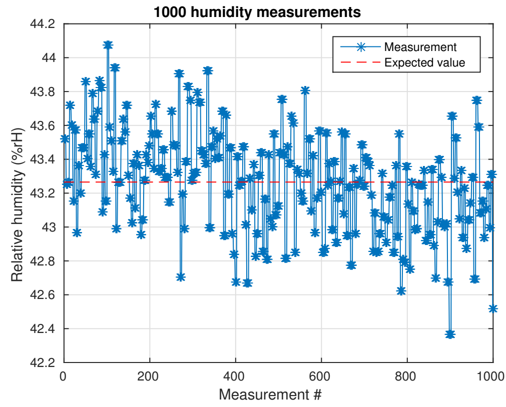


Figure 5.4: 1000 measurements acquired with golden humidity sensor.

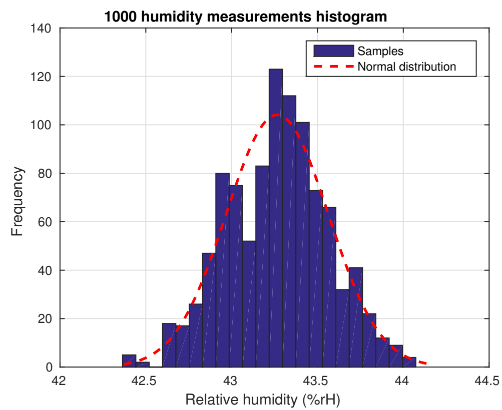


Figure 5.5: Histogram based on the humidity measurements.

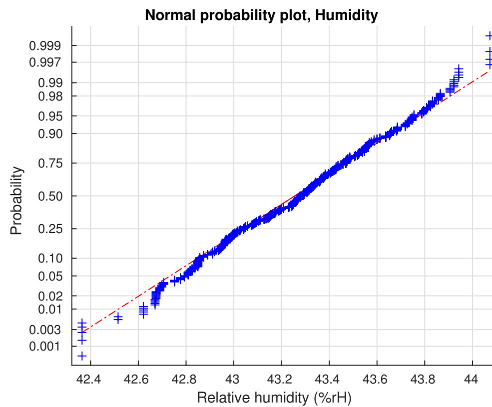


Figure 5.6: Probability plot of the humidity values.

5.1.3 Pressure

The measurement values of the absolute pressure are plotted in Figure 5.7. As with the above value figures, the red dotted line is the expected pressure value based on the values of the measurements. The expected value for the absolute pressure is $\mu_p = 1010.3$hPa and the standard deviation is $\sigma_p = 0.0538$hPa.

Figure 5.8 shows the histogram based on the pressure values and Figure 5.9 shows the probability plot based of the same values.

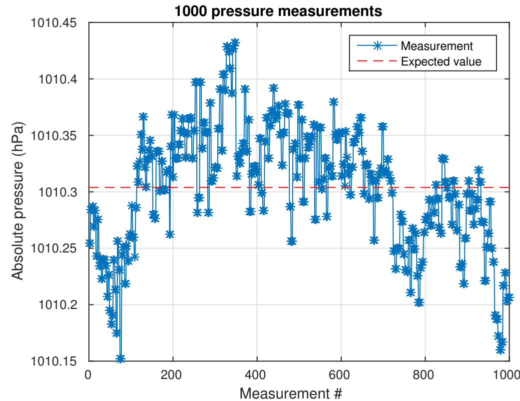


Figure 5.7: 1000 measurements acquired with golden pressure sensor.

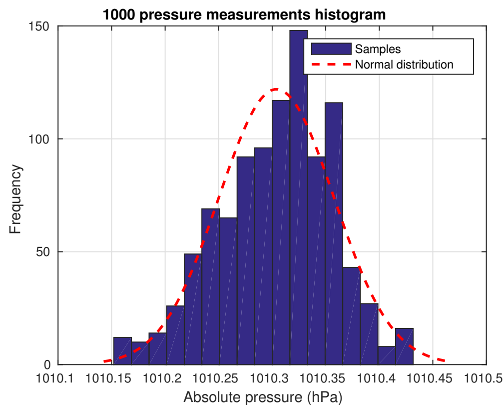


Figure 5.8: Histogram based on the pressure measurements.

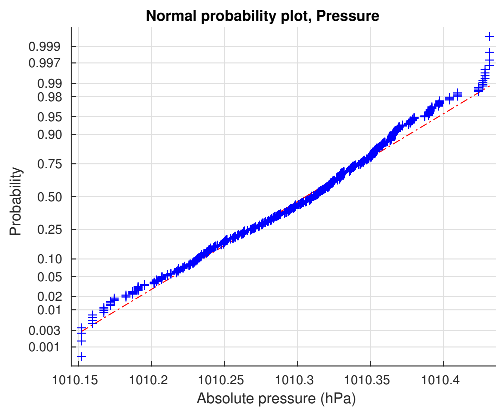


Figure 5.9: Probability plot of the pressure values.

5.2 Testing Of Points

Two test runs are performed on the 25 existing beta Points. The first test run results in plots with values from the sensors together with values from the corresponding golden sensor, histogram plots of the difference between sensor on DUT and the golden sensor, a plot of the test data from the particle sensor and figures presenting the microphone results of the 25 devices. The second test run only contains the histograms of the difference between sensors on DUT and the golden sensors.

The two test runs are performed with different golden sensors, in the attempt to show how differently the perform.

Figure 5.10, 5.11 and 5.12 shows plots of the values from the golden sensors and the SUTs for the temperature, humidity and absolute pressure sensor. The plots also shows the sensors margin of error. Dots represents the acquired values and the dotted lines are the margin of error for the sensors based on those values.

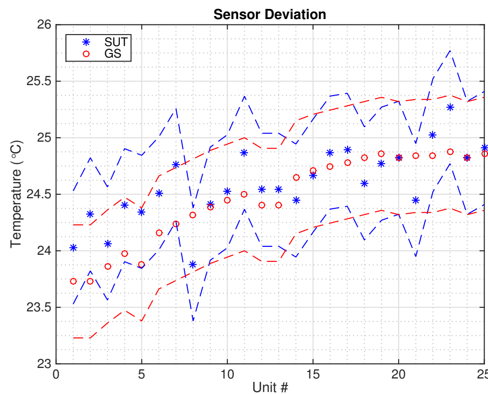


Figure 5.10: Temperature measurement output of DUT and GS and their margin of error.

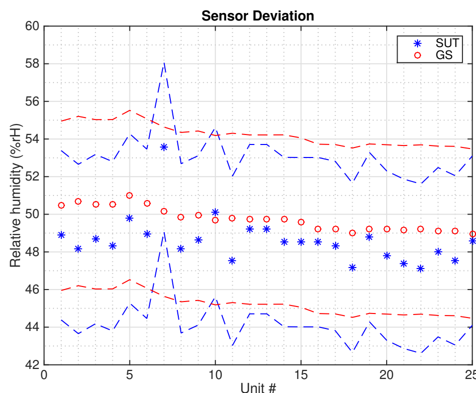


Figure 5.11: Relative humidity measurement output of DUT and GS and their margin of error.

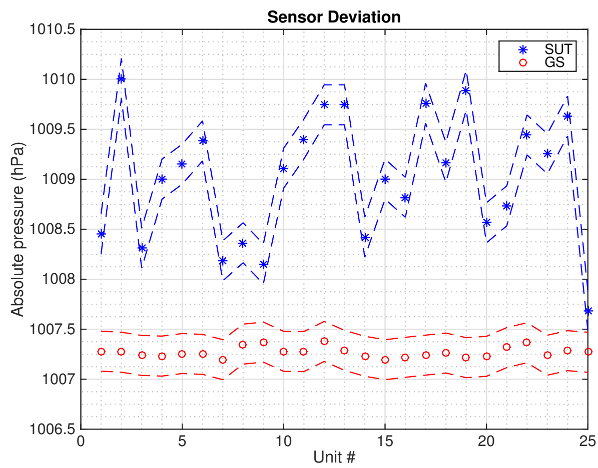
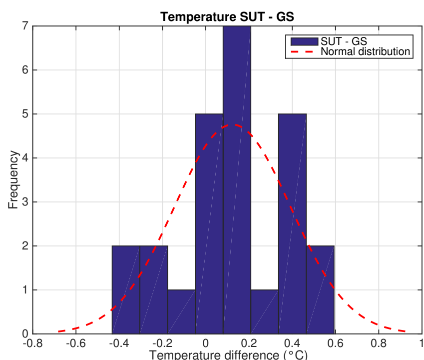
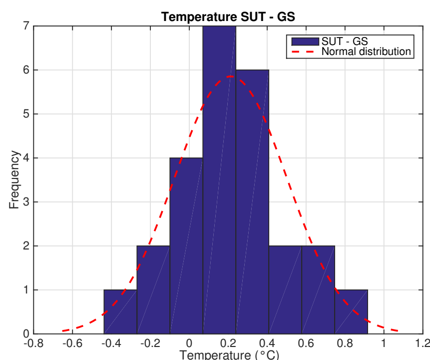


Figure 5.12: Absolute pressure measurement output of DUT and GS and their margin of error.

The histogram in Figure 5.13, 5.14 and 5.15 are based on the difference between the values acquired from the DUT and the GS values for temperature, humidity and pressure sensor respectively in the two different test runs. The histogram should optimally be centered at 0°C.

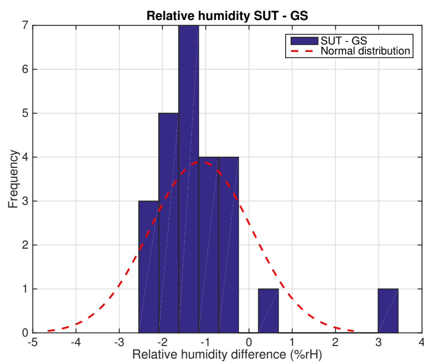


(a) Test run 1.

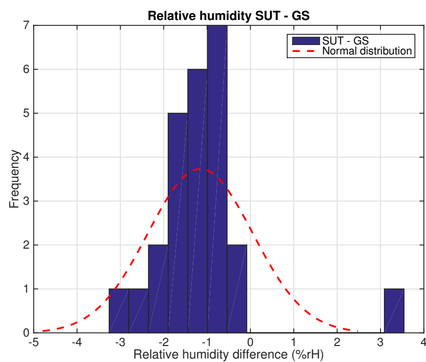


(b) Test run 2.

Figure 5.13: Difference in temperature measurement output between DUT and GS.

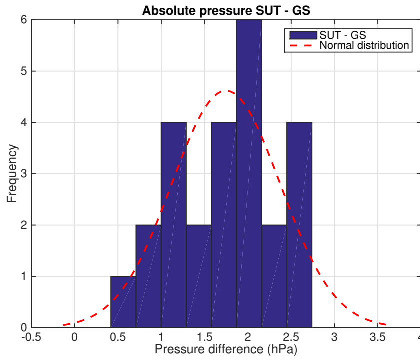


(a) Test run 1.

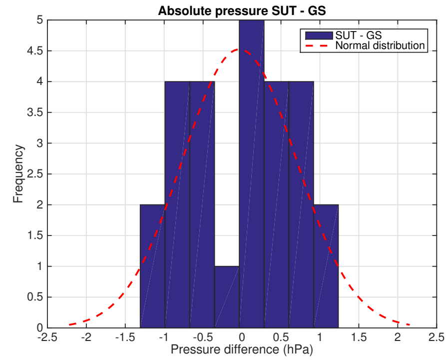


(b) Test run 2.

Figure 5.14: Difference in humidity measurement output between DUT and GS.



(a) Test run 1.



(b) Test run 2.

Figure 5.15: Difference in pressure measurement output between DUT and GS.

The expected value, μ, and standard deviation, σ, from the above histograms are shown in Table 5.1, 5.2 and 5.3

Table 5.1: Difference between SUT and GS for temperature.

	Test run 1	Test run 2
μ_t	0.1236°C	0.2124°C
σ_t	0.2687°C	0.2884°C

Table 5.2: Difference between SUT and GS for humidity.

	Test run 1	Test run 2
μ_h	-1.1168%rH	-1.1567%rH
σ_h	1.1793%rH	1.2119%rH

Table 5.3: Difference between SUT and GS for absolute pressure.

	Test run 1	Test run 2
μ_p	1.7466hPa	-0.0337hPa
σ_p	0.6266hPa	0.7292hPa

5.2.1 Particle Sensor

Measurement data acquired from the particle sensor is shown in Figure 5.16. The red line in the figure represents the values measured with no LEDs active and only the ambient light as source. The other three lines represents the output when the LED is switched on one at a time. The figure shows that for unit 11, the ambient light measurement and LED 1 intensity is correct, but that the measurements with LED 2 and 3 gives very low output.

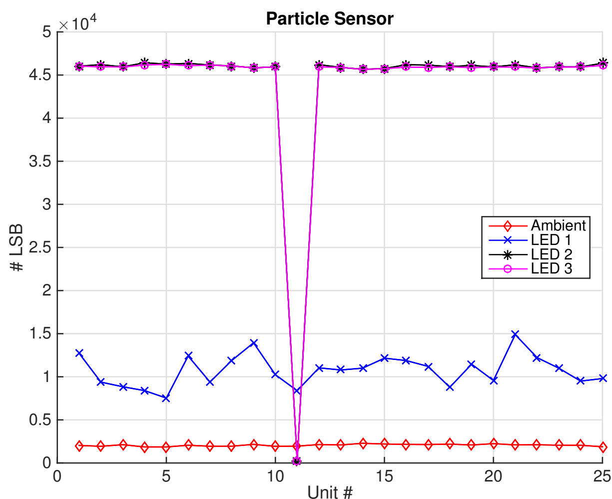


Figure 5.16: Output data from the particle sensor test, test run 1. Unit number 11 shows deviant behavior.

5.2.2 Microphone And Speaker

The results from the 25 microphone tests are shown in Figure 5.17 and Figure 5.18. Figure 5.17 shows the number of zero crossings in the microphone samples from the 25 devices. The sinus tone played by the speaker had a frequency of 875 Hz. The number of zero crossings are represented as integers. This results in quite big distinct steps in measured frequency. 13 zero crossings gives a frequency of 812,5 Hz, 14 gives 875 Hz and 15 gives 937.5 Hz. Figure 5.18 shows the average power of the acquired sound data.

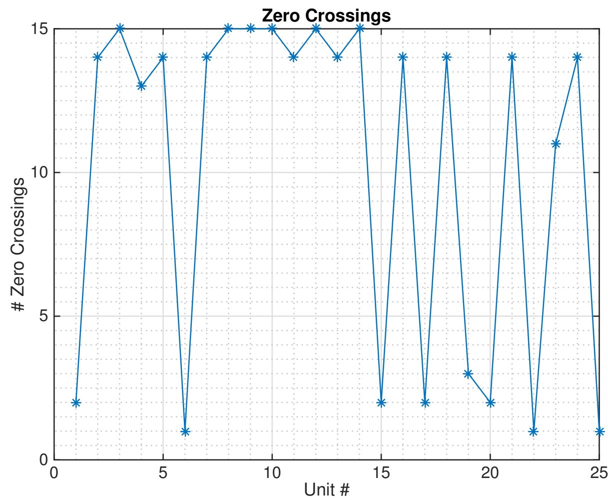


Figure 5.17: The number of zero crossings in the mic samples for all 25 units in the test run.

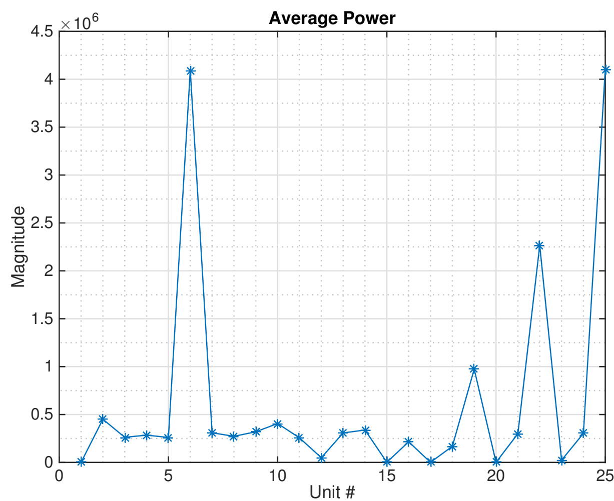


Figure 5.18: The average power of the sound samples for all 25 units in the test run.

5.2.3 Current

The current usage during test run is measured and plotted in Figure 5.19, Figure 5.20 and Figure 5.21. In Figure 5.19 the current usage is normal relative the theory. It takes about one second for the DUT to initialize and set up the MCU. At one $t_1 = 1s$ all sensors are powered up, resulting in a higher current usage. During the wait after the sensors have been powered on the usage is constant. The peak at $t_2 = 2s$ is when all the sensors acquisitions data. Between t_2 and $t_3 = 3s$ the system waits for the sensor data acquisition to be completed, and at the end of that interval the data is fetched from the sensors. During the interval t_3 to $t_4 = 5s$ the MCU enters the deep sleep mode. This results in a current usage as low as 5 μA. After t_5 the test is completed.

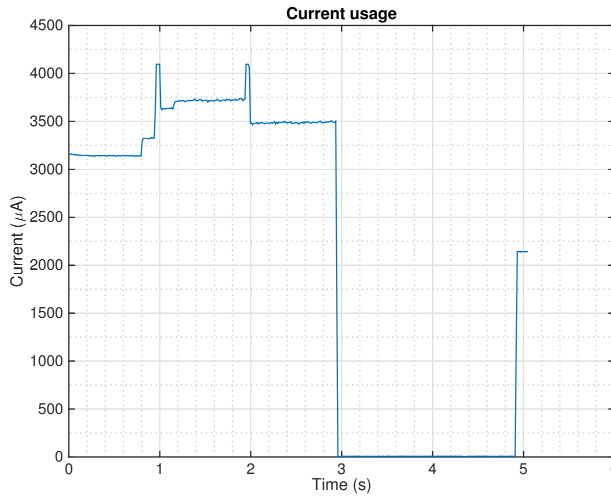


Figure 5.19: Current usage in μA during test of DUT with expected current usage.

Figure 5.20 shows the same steps as Figure 5.19 but from a different DUT and measured in the mA range. The consumption is normal during the MCU initialization, but when the sensors are powered on, the system uses twice as much current as expected. The current usage during the deep sleep is also quite high, although all the sensors are supposed to be powered off.

In Figure 5.21 the current usage is normal in the beginning of the test. But When the sensor measurements are triggered at $t = 2s$ the usage increases significantly. It reaches out of range for the μA measurement range and peaks at 30 mA until the MCU shuts down the sensors and goes to deep sleep.

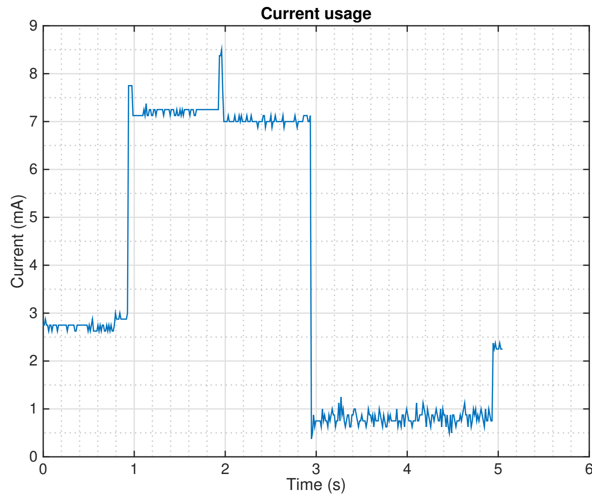
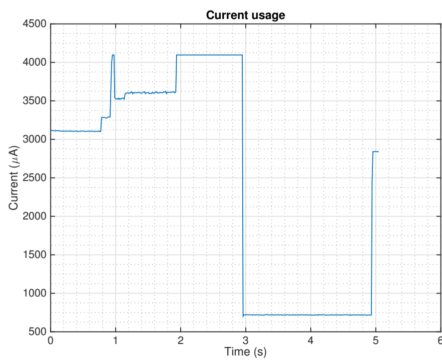
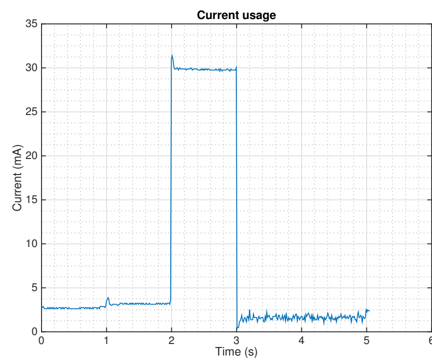


Figure 5.20: Current usage in mA during test of DUT with deviant current usage.



(a) Current usage in μA.



(b) Current usage in mA.

Figure 5.21: Current usage for a DUT with error.

6. Discussion

This chapter consists of discussions based on the acquired test results from the 25 beta units. It discusses how the developed test system performs and connects the results to the counterfeiting problem. At last it presents possible improvements and areas to further develop.

6.1 Test Results

This section presents a discussion on how the acquired test results may be used to find erroneous components.

6.1.1 Golden Sensors

The test results of the golden sensors, presented in section 5.1, are based on 1000 measurements for each of the golden sensors. The tests were performed to get an estimation of the sensors' statistical parameters. The measurements were conducted during $t = 15$ seconds and the environment is assumed to be constant. As seen in Figure 5.1, 5.4 and 5.7 it is difficult to tell if the ambient environment actually is constant during the tests since the values varies. The varying values depends on the measurement error of the sensors, however changes in the environment during measurements affects the results as well. For the temperature sensor in Figure 5.1 it looks like the ambient temperature changes a little. Though, since the maximal difference in measured values is only 0.16 °C it may be the sensor's measurement error. The relative humidity sensor's result in Figure 5.4 shows that the relative humidity decreases 0.5 %rH during the $t = 15$ seconds. This could be due to a non constant ambient environment. According to the specifications of the sensor it is still inside the accuracy margin. The absolute pressure values in Figure 5.7 also varies but stays within its margin of error.

Figure 5.1 and 5.3 shows the distinct resolution of the temperature sensor. Due to the distinct resolution, the probability plot in Figure 5.3 does not fit a normal distribution perfectly, but the values still follows a normal distribution curve. Figure 5.2 also shows that the output values of the temperature sensor is close to a normal distribution. Figure 5.5 and 5.6 indicates that the humidity measurement values approximately follows a normal distribution. The same applies to the pressure sensor in Figure 5.8 and 5.9. This also conform to the central limit theorem in statistics.

The results indicates that the GSs behave as they are supposed to, but there could be a bias relative to the true values. The exact biases are impossible to find without knowing the true values. Using calibrated high accuracy sensors, the supposed bias for the golden sensors may be estimated.

6.1.2 Testing Of Points

The figures in Section 5.2 presents results from test runs of the 25 beta units. The section begins with results from the sensors tested against golden sensors. In total two test runs are performed. Most of the data is from the first test run but for a comparison purpose, data from a second test run is used in some figures.

Figure 5.10, 5.11 and 5.12 shows the test output of the temperature, humidity and pressure sensors along with their corresponding GS values. The temperature measurements are plotted in Figure 5.10. This figure shows an increase in temperature during the testing and that both the SUT and GS follows that increase. This points to the golden sensor model is working as expected. The temperature measurements also show that all units have overlapping f_{dev} with the golden sensor. Hence all temperature sensors will pass the test.

Figure 5.11 shows a plot of the relative humidity measurements. Unit number 7 stands out compared to the other SUTs values, though it is still within the specified margin of error and is considered working as expected. In the relative humidity plot, almost all GS values are greater than the SUT values. It is not by much, but still clear. This may point to a minor offset in the GS output. Since the offset is small, all SUTs passes. If the offset of the GS is unknown it can potentially be a problem in the validation process. The offset error is more noticeable in Figure 5.12 presenting the absolute pressure measurements. The figure shows that all SUT values are far above the GS values and that their margin of error do not overlap. This results in failing all the pressure SUTs if the GS is considered accurate. Since all GS values are far below the SUT values it is more likely an offset error in the GS. If there is a potential offset in the GS values, SUTs may still be correct even without an overlapping margin of error. For the golden sensor model to work without offset corrections, the GS has to be calibrated with a known correct instrument. Otherwise, the possible offset of the golden sensor has to be taken into account in the validation process.

In an approach to calibrate and find the offset of a golden sensor the difference between SUT and GS values is used. Figure 5.13, 5.14 and 5.15 shows histograms of the difference SUT-GS. The figures are based on two different test runs with the same 25 units used in both runs. The only thing varied is the golden sensors used which is done to determine that the golden sensor model works for different GSs. Optimally, the histograms should be centered around 0. This means that the expected offset between the SUT and GS is 0. For the temperature measurements in Figure 5.13 the center is close to 0, this is also shown in Table 5.1.

The difference in relative humidity measured from the SUTs and GS shown in Figure 5.14 shows that for both test run 1 and test run 2 there is one value which is off. This value is still accepted since it is within the margin of error even though it deviates much from the rest of the values. Due to the small population the results are based on, it is difficult to tell if this is a normal behavior. The humidity differences do not appear to fit the normal distribution, this is because the 25 measurements are too few and that the sensor has a wide margin of error. In both Figure 5.14a and Figure 5.14b, all values are within the margin of error and the sensors are considered working as supposed.

Figure 5.15 shows the two different results of the absolute pressure sensor tests. The distribution of the values are compact and looks fine, but in Figure 5.15a all values has an offset around 1.5 hPa while the values in Figure 5.15b does not have this offset. The only thing that differ in the two test runs is the GS. The conclusion of this is that the absolute pressure GS used in test run 1 has an output offset and is not working as expected. The estimated expected offset is shown in Table 5.3. If the expected value of the offset is added to all the golden sensor values for test run 1, Figure 5.15 will look much better and more sensors will pass the test. This is shown in Figure 6.1.

During the measurements the temperature is around 24 °C. For the pressure sensor used,

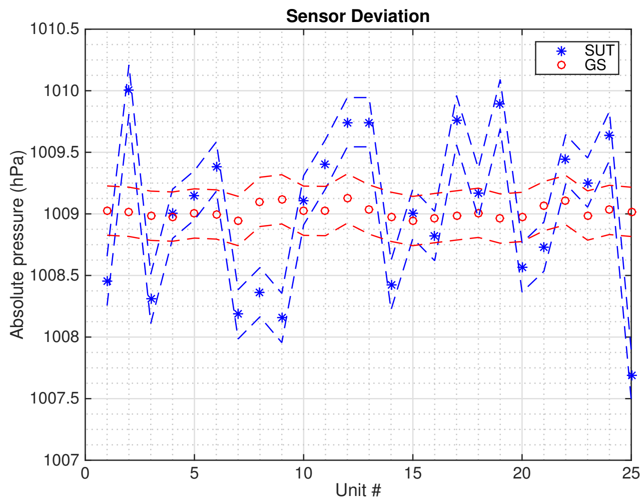


Figure 6.1: The SUT and GS values from the absolute pressure test. The expected value of the difference SUT-GS is added to all the GS values

the lower boundary for the high accuracy interval of f_{dev} is 20 °C, stated in Equation 4.13, which is close to 24 °C. The theory presents a constant f_{dev} during the high and low accuracy intervals. In practice, the step between the high and low accuracy intervals is not just one big step. According to Equation 4.13, the high accuracy interval should hold down to 20 °C, which would mean that the accuracy decreases gradually while decreasing below 20 °C. As shown in Figure 5.15 the margin of error is narrow. Even if the GS offset is compensated, many of the pressure sensors will not pass. According to the specifications of the sensor the accuracy is only theoretically estimated. This could be a reason to the high fail rate.

Overall, a higher test population is required for a stable validation model. Optimally these tests should be done in the factory to optimize it for the prevailing environment there. Looking at the two test runs in Figure 5.13, 5.14 and 5.15 the difference between the histograms is mainly explained with the use of different golden senors. The low number of measurement data can also affect the result. To be able to guarantee the test results, the golden sensor has to be tested and calibrated more thoroughly and statistical data has to be acquired for further improvement.

6.1.2.1 Particle Sensor

The result from the particle sensor of the tested devices, plotted in Figure 5.16, shows that the values are approximately the same for all devices except one. Device number 11 shows indication of two malfunctioning LEDs. A malfunctioning photo diode or digital sensor can be precluded since both the ambient light value and LED 1 shows similar results as the other tested devices. The malfunction can depend on various things. The LEDs could be improperly connected to the PCB or not work at all. It could also be the connection from the component driving the LEDs that is not working as expected. In this case the LEDs does not seem to be working at all which makes the error easy to spot. Even if they would work with reduced intensity it is easy to spot the error in Figure 5.16. The results from LED 1 varies the most. It is still a clear gap in measured intensity between the ambient light and LED 1. Here, statistics of the test history can be used to improve the validation model.

6.1.2.2 Microphone And Speaker

The microphone test shows that quite a few of the microphones fail. This partly depends on a manufacturing error in the soldering-paste stencil. Four of the microphones gives high average power results and six of them shows low power. Almost all of the microphones with abnormal average power levels shows too low number of zero crossings. For the units where the average power is high, but the zero crossing count low, the offset or gain could be off. A too high offset could exceed the ADC reference or just result in very high values. An incorrect gain could push the complete signal above or below the 0 level. In this case it was due to a manufacturing error, but errors like this could also point to an errant microphone, microphone amplifier or speaker amplifier.

6.1.2.3 Current

The current usage plotted in Figure 5.19 shows the anticipated usage for each Point. The current usage conforms to the theory in Table 4.1 and Table 4.2. When the acquisition MCU and all sensors are active, $t \approx 1s - t \approx 3s$ in Figure 5.19, the usage is a bit higher than the theoretical usage. This may be because accompanying components in the overall system consumes more power than expected or leaking low currents. The firmware is not optimized for the lowest power consumption which can lead to a marginally higher current usage than expected from theory. The peaks at $t_0 = 1s$ and $t_1 = 2s$ could be due to some different reasons. It could be leaks from the sensor decoupling capacitors when the power is switched on. The peak at t_1 is during the sensor measurements. During this time the sensors are active and the LEDs in the particle sensors as well. Hence, the peak could be because of that.

The current usage during the deep sleep mode,$t \approx 3s - t \approx 5s$, is a few μA higher than in Table 4.1. As above, this can be due to the theory which only includes the MCU in the current usage estimation and not the overall components on the DUT.

Figure 5.20 shows a DUT with a malfunction in its current usage. The usage is twice the anticipated usage with all sensors powered on and activated. This can point on a current leakage at some of the sensors which can be due to a short or a bad sensor. The test firmware is designed to switch on the power to all the sensors at the same time. This makes it hard to determine the specific sensor which consumes more power than expected. A better design would be to power the sensors in different steps. This way it becomes easier to determine what sensor is faulty.

Since the tests does not include any tests involving the WiFi chip yet, the current usage during active WiFi communication stated in Table 4.3 can not be measured.

6.2 Test Method Decisions

During the development process design decisions has to be made. This section covers some of these decisions and the different factors are discussed.

6.2.1 Microphone And Speaker

A first problem with testing the microphone is the environment and circumstances in the factory. As described in the section 3.3.1 a specific test environment is required to verify the specification of a microphone. As stated in section 3.3.2 a similar test station as for the microphone is needed to verify the speaker's specifications. Fulfilling those requirements in the production line is difficult since testing is supposed to be performed quickly and it may not even be possible to have such specific construction at a factory. It was decided to skip such a thorough test

and settle with a test verifying the normal functionality of the microphone and speaker instead of verifying that they match their specifications.

To test the microphone, a speaker is needed to generate the test signal. This can be done by having an external speaker and amplifier on the test system able to generate the test tones independent of the DUT. It can also be done by letting the DUT generate the test tone with the built in speaker which ideally requires no external parts. Instead, it requires the internal part to work correctly. The method used in the test system is somewhere in between and utilizes as many internal parts as possible (mainly the speaker amplifier) but it also requires an external speaker.

Self-testing, where internal parts tests each other, results in less external parts on the test station. Though it makes some tests depend on other tests and components with unknown functionality. If the speaker's amplifier is broken both the speaker and microphone tests will fail, even if the microphone works perfectly fine. An external amplifier and speaker on the test station does not have these problems. The external setup can also malfunction, resulting in the test failing every unit. If the test history is monitored, this malfunctioning external test can probably be discovered after a few units. By using only internal parts the error can be concentrated to the individual unit. If every failing unit is to be debugged and fixed a precise error message is preferred. Self-testing depending on internal parts may give a less precise error messages.

6.2.2 JTAG & SWD

Boundary-Scan (JTAG) is a common test method for PCBAs with complex integrated circuits. At a first glance it is a promising method for the case with Point. Boundary-Scan would be good to use but on Point few components support it. Test station uses SWD for flashing and debugging of Point. SWD is compatible with JTAG and the low-power MCU allows external components to use JTAG. The only component except from the low-power MCU supporting JTAG is the WiFi chip. The WiFi chip is not yet included in the test-suite. This resulted in building a custom FW instead of using boundary-scan.

Most of the tests are controlled from the low-power MCU with the test FW. This gives equal control of the hardware pins as JTAG. The firmware can also drive the communication buses like I^2C to allow full control of all connected sensors and other peripherals. JTAG is also designed to be able to drive I^2C and similar but this is nevertheless done by a custom firmware.

6.3 Test System And Counterfeit

The test system is designed to validate the functionality of the most critical components in every produced Point. The design is developed with the counterfeit problem in mind and with the goal to avoid delivering any units containing counterfeit or errant components. The focus has been to verify that critical components works as expected. Test results from the 25 pre-production units shows how the developed test station performs in the various tests. The results indicate that it is possible to distinguish if a tested sensor or component is not working properly and produces incorrect results.

This is a first and important step in finding counterfeit components, but counterfeit electronic components is a multidimensional problem. They come in a variety of types and ways with different behavior [9]. It is not obvious how a counterfeit component performs or behaves during different stages of its lifetime. This makes it hard to address the counterfeit problem on component level.

The developed test station tries to tackle possible failures in different ways. One part of the test is to validate the power consumption of the components. Often, abnormal power consump-

tion points to a failing component. In terms of counterfeit, this covers some different areas. The failing component could be of lower grade than specified, just relabeled. It could be a recycled component performing worse because of aging. It could be from a overproduced batch where scrapped components not passing the validation process are sold off. These are just some potential sources of counterfeit components that a power consumption test could address.

In another part, the system examines how the individual sensor and component performs. Abnormal or poor test results could point to different kinds of counterfeit as stated above.

The digital sensors also gets their internal logic verified. By verifying their ID and internal register setup, counterfeit components marked as one type of component but with incorrect or failing internal functionality can be distinguished.

With these type of tests many possible types of counterfeit can be addressed. In reality it is not that simple. The developed methods can find an individual failing component, but the hard part is to determine if that component is actually counterfeit. Authentic components also has a failure rate. They may be delivered broken or be damaged during the manufacturing and assembly process. As shown in the results, some components may not fulfill their specified accuracy. This makes it hard to determine if a single component is a counterfeit or just malfunctioning. A failing component could be analyzed and tested for counterfeit with more complex methods such as thermal imaging [36]. These type of tests are time consuming and requires a lot of external test equipment. Therefore, failure and counterfeit analysis for every individual failing component is not reasonable.

The big problem is shown when whole batches of components are counterfeit. Hence, it is more reasonable to do a deeper investigation of counterfeit if one type of components keep failing for multiple units. It is therefore important to keep track of the test history. Logging all test results and storing them makes it possible to label produced units and store information such as production date and component batches used. This can potentially help to find counterfeit components that are working in the beginning but changes performance after a while. If a batch has a high fail- and return rate of customer units, the failing components could be matched with the test results. If the units has somewhat abnormal test results these abnormalities can influence the validation process later on. They may also point to bad component batches.

6.4 Error Factors

The models of how to validate a test value is mainly based on theory. In practice there are some more error factors. One of them is due to the reflow step. After the reflow step the PCBA will have a much higher temperature than the ambient environment and the golden sensors. For the golden sensor model to work both the GS and the SUT needs to do the measurements during equal conditions. This means that the PCBA ready for testing needs to cool down before it enters the test station. Since time during production is precious and the test directly follows the reflow, the cool off period needs to be considered. Other tests like Automated Optical Inspection (AOI) or In-Circuit Test (ICT) can be done prior to the specific sensor tests and give the PCBA time to cool down. If the PCBAs does not reach the correct temperature, the test validation model needs to compensate for this. Compensating this can be difficult since the time between reflow and test varies for individual PCBAs. The cool down curve will also vary depending on the ambient environment.

A second, more specific problem is related to the MEMS humidity sensor. After soldering and heating up the component, it needs some time for re-hydration of the sensor element. The accuracy specification of the sensor can only be guaranteed after this re-hydration period. Before that, there is an unknown offset in the sensor's measurements. The re-hydration period can take up to three days depending on the ambient humidity and temperature. The offset error

needs to be added to the validation model if the sensor is tested directly after soldering. Since different sensors has different offset depending on the time since reflow and the fact that not two sensors are exactly the same, it is a complex task. One way to solve this could be to validate the sensor values with lower accuracy and pass the sensor if everything else with it seems to be fine. Statistics of tested working sensors can tell if the measured value has a reasonable offset at the time of testing. Another way could be to acquire multiple measurement values and try to predict how the offset decreases.

6.5 Further Development

There are still components on Point not covered by any tests. This section discusses some of them and how they can extend and complement the test station.

6.5.1 Additional Components And Techniques

The MCU with WiFi and its antennas as well as the Hall effect sensor are examples on components not yet covered by the test system.

Tests on the WiFi chip is planned to be done as part of a functional test in the end of the assembly process. By letting every assembled Point connect to a WiFi, perform some test communication and verify the signal strength. This can also be a test case in the test station. This way the quality and performance of antennas and the antenna switch can be validated.

The Hall effect sensor needs to be tested as well. It is not located on the main PCBA but on a smaller PCBA housing battery connectors. These smaller battery connector PCBAs are mounted to the main PCBA with pin headers in the final assembly. Hence they are not mounted at the test station step of production. This makes it impossible to test the Hall effect sensor in the test station. Instead it can be tested in the final functional test by mounting the assembled Point to a metal plate. The following additional test techniques complements to the developed test station:

- A final functional test can be used to test the RGB LEDs, battery connectors, WiFi and Hall effect sensor on Point.
- Built-In Selt-Test (BIST) can be a way to diagnose how the sensors in Point ages. A BIST could be scheduled for regular runs and the results pushed to the companies servers. This can distinguish potential counterfeit components.
- Using automated optical inspection in the production enables finding early signs of malfunctioning units.

6.5.2 Image Analysis

AOI is a standard method to validate the quality of PCBAs. An alternative to the standard AOI method is to build a custom version on the test station. Hardware-wise a custom optical inspection does not require much more than a camera mounted on the test fixture. The complexity of a custom optical inspection lies in the analysis done in software. Image analysis gives many to possibilities. It can be used to authenticate individual components and verify that they are properly placed and soldered. The image analysis can also be a useful tool to discover counterfeit components. Improper components may vary in color, size or labeling. On the software side open libraries like OpenCV [37] can be used to build a basic platform for a custom AOI.

One variant of image analysis is thermal imaging, which focuses on finding counterfeit components. Thermal imaging is a new part of optical inspection and is still not very common

in testing. The method uses an infrared camera and software to analyze the heat generation in ICs and conclude if they are warmer than they are supposed to. The heat from an IC depends on the current they use and older or used components use more current than newly manufactured ones [36]. A drawback of thermal imaging making it difficult to implement in a common test station is the fact that the IC has to be decapped (e.g. removing the plastic cover to expose the internal parts of the IC). Another problem is that not all ICs are designed with the silicon die facing upwards. Thermal imaging is a way to find counterfeit components, but if used with decapping, such components can not be used in the final product.

The decision of using AOI and/or AXI is also a matter of cost. Most optical inspections on the market today are expensive. AOI or AXI would be a good complement to the designed test system but the price needs to considered.

7. Conclusions

Counterfeit components is a growing problem which endangers the safety of electronics. Big corporations are introducing standards to prevent the problem, but everyone can contribute. For small hardware companies it is not obvious how to contribute given their limited resources. This thesis develop test methods to find counterfeit components. The methods can be applied in the manufacturing process of a product giving an opportunity for everyone to contribute.

The test results indicate that it is possible to distinguish components with performance issues, and since this may be an indication of counterfeit components, the test system has potential of finding such components. The system can assure the quality of individual units, preventing malfunctioning devices to pass tests, whether it is a counterfeit or a poor component. This prevents products containing counterfeit components to reach the customer.

Individual counterfeit component is not the big problem, but rather when they show up in large quantities. To attack this, the test and production history needs to be logged. Statistics from a batch with numerous errors can point to counterfeits. The test system will help find those batches.

Regardless of the underlying cause of the errors, it lies in the company's interest to investigate it. Both to ensure the quality of the company's products and to contribute in the work against the counterfeiting problem. A company affected of bad components needs to communicate the problem to its factory, control the component's sources and contact the IC manufacturer. Analyzing the component's supply chain is important work. Tracing all used components to the actual manufacturer through their suppliers can help guaranteeing the components are not counterfeit.

It can be difficult to motivate a small company with limited resources to address the counterfeiting problem. Though, if the problem keeps increasing, they will have to implement these kind of tests. In hardware production, a variety of test methods exists. Their main goal is to make sure the final product works according to its specification and does not have any manufacturing errors. None of these test methods focuses on finding counterfeit components, but with little additional work the methods can be combined and complemented with custom test methods to help counteract the counterfeiting problem. By adding logging of test data and keeping statistics counterfeit components can be found.

Continuous development in the test field is required to combat counterfeiting of components. It does not require much from companies to help prevent the problem. If every one contributes, the problem can be controlled.

References

[1] N. Kae-Nune and S. Pesseguier, 'Qualification and testing process to implement anti-counterfeiting technologies into ic packages', in *Proceedings of the Conference on Design, Automation and Test in Europe*, EDA Consortium, 2013, pp. 1131–1136 (cit. on pp. 1, 3).

[2] U.S. Department Of Commerce, Bureau Of Industry And Security, Office Of Technology Evaluation. (Jan. 2010). Defense Industrial Base Assessment: Counterfeit Electronics, [Online]. Available: https://www.bis.doc.gov/index.php/forms-documents/doc_view/37-defense-industrial-base-assessment-of-counterfeit-electronics-2010 (visited on 28th May 2015) (cit. on pp. 1, 9).

[3] Dominik Lorenz. (24th Apr. 2012). Aging Analysis of Digital Integrated Circuits, [Online]. Available: https://mediatum.ub.tum.de/doc/1096635/1096635.pdf (visited on 26th May 2015) (cit. on pp. 1, 3).

[4] M.-L. Tan. (12th Jul. 2015). Hardware Is The New Software, [Online]. Available: http://techcrunch.com/2014/07/12/hardware-is-the-new-software/ (visited on 26th May 2015) (cit. on p. 1).

[5] Tom Whitwell. (13th Jun. 2014). Inside Shenzhen: China's Silicon Valley, [Online]. Available: http://www.theguardian.com/cities/2014/jun/13/inside-shenzen-china-silicon-valley-tech-nirvana-pearl-river (visited on 8th Jun. 2015) (cit. on p. 1).

[6] K. Huang, J. M. Carulli and Y. Makris, 'Counterfeit electronics: a rising threat in the semiconductor manufacturing industry', in *Test Conference (ITC), 2013 IEEE International*, IEEE, 2013, pp. 1–4 (cit. on pp. 1, 3).

[7] K. Lewis, 'The fake and the fatal: the consequences of counterfeits', *The Park Place Economist*, vol. 17, no. 1, p. 14, 2009 (cit. on p. 1).

[8] M. Pecht and S. Tiku, 'Bogus: electronic manufacturing and consumers confront a rising tide of counterfeit electronics', in *IEEE Spectrum, vol. 43, no. 5*, IEEE, 2006, pp. 37–46 (cit. on p. 3).

[9] Ujjwal Guin, Mohammad Tehranipoor, Dan DiMase, Mike Megrdichian. (2013). Counterfeit IC Detection and Challenges Ahead, [Online]. Available: http://www.engr.uconn.edu/~tehrani/publications/ACM-SIGDA-2013.pdf (visited on 27th May 2015) (cit. on pp. 3, 55).

[10] IDEA. (2011). IDEA-STD-1010B - Acceptability of Electronic Components Distributed in the Open Market, [Online]. Available: http://www.idofea.org/products/118-idea-std-1010b (visited on 27th May 2015) (cit. on p. 3).

[11] I. Poole. (2015). ICT, In Circuit Test Tutorial, [Online]. Available: http://www.radio-electronics.com/info/t_and_m/ate/ict-in-circuit-test-tutorial.php (visited on 12th May 2015) (cit. on p. 10).

[12] ——, (2015). Automatic optical inspection, AOI systems, [Online]. Available: http://www.radio-electronics.com/info/t_and_m/ate/aoi-automatic-automated-optical-inspection.php (visited on 11th May 2015) (cit. on p. 11).

[13] ——, (2015). Automated X-Ray Inspection AXI for PCB & BGA, [Online]. Available: http://www.radio-electronics.com/info/t_and_m/ate/automated-x-ray-inspection-pcb-bga.php (visited on 12th May 2015) (cit. on p. 11).

[14] M. J. Norris. (24th Sep. 2000). Vectoral imaging... the new direction in automated optical inspection, [Online]. Available: http://amtest.bg/press/AOI/Vectoral%20Imaging.pdf (visited on 12th May 2015) (cit. on p. 12).

[15] JTAG Technologies B.V. (2015). JTAG Boundary-Scan, firmly based on IEEE standards, [Online]. Available: http://www.jtag.com/en/content/standards (visited on 11th May 2015) (cit. on p. 12).

[16] ARM Ltd. (2009). Low Pin-count Debug Interfaces for Multi-device Systems, [Online]. Available: http://web.archive.org/web/20150321140036/http://www.arm.com/files/pdf/Low_Pin-Count_Debug_Interfaces_for_Multi-device_Systems.pdf (visited on 11th May 2015) (cit. on p. 12).

[17] JTAG Technologies B.V. (2015). About boundary-scan, [Online]. Available: http://www.jtag.com/en/content/about-boundary-scan (visited on 11th May 2015) (cit. on p. 12).

[18] ——, (Jun. 2008). When does boundary-scan make sense, [Online]. Available: http://www.jtag.com/en/content/lead-form?download=417 (visited on 11th May 2015) (cit. on pp. 12, 17).

[19] ARM Ltd. (2014). Serial Wire Debug, [Online]. Available: http://www.arm.com/products/serial-wire-debug.php (visited on 11th May 2015) (cit. on p. 12).

[20] NPTEL. (2015). Module 11 :Built in Self test (BIST), [Online]. Available: http://www.nptel.ac.in/courses/106103016/30 (visited on 29th May 2015) (cit. on p. 13).

[21] A. Cort. (1st Jul. 2002). Functional Testing of PCBs, [Online]. Available: http://www.assemblymag.com/articles/83988-functional-testing-of-pcbs (visited on 29th May 2015) (cit. on p. 13).

[22] DPA Microphones. (2011). Microphone specifications and measurement techniques, [Online]. Available: http://www.dpamicrophones.com/en/Mic-University/Tech-Guide/Microphone-specifications-and-measurement-techniques.aspx (visited on 13th May 2015) (cit. on p. 13).

[23] Wolfgang Klippel (2011). End-Of-Line Testing, Assembly Line - Theory and Practice, Prof. Waldemar Grzechca (Ed.), ISBN: 978-953-307-995-0, InTech, DOI: 10.5772/21037. [Online]. Available: http://www.intechopen.com/books/assembly-line-theory-and-practice/end-of-line-testing (cit. on p. 14).

[24] Z. Nakutis, 'Embedded systems power consumption measurement methods overview', *MATAVIMAI*, vol. 2, no. 44, pp. 29–35, 2009 (cit. on pp. 14, 15).

[25] D. L. Jones. (2010). The μCurrent - A professional precision current adapter for Multimeters, [Online]. Available: http://alternatezone.com/electronics/ucurrent/uCurrentArticle.pdf (visited on 13th May 2015) (cit. on pp. 14, 15, 19).

[26] N. Instruments. (11th Nov. 2014). Current Measurements: How-To Guide, [Online]. Available: http://www.ni.com/tutorial/7114/en/ (visited on 21st May 2015) (cit. on p. 14).

[27] P. Semig and T. I. Collin Wells. (2nd Aug. 2012). A Current Sensing Tutorial–Part 1: Fundamentals, [Online]. Available: http://www.eetimes.com/document.asp?doc_id=1279404 (visited on 23rd May 2015) (cit. on p. 15).

[28] NXP Semiconductors. (4th Apr. 2014). I^2C-bus specification and user manual, [Online]. Available: http://www.nxp.com/documents/user_manual/UM10204.pdf (visited on 20th May 2015) (cit. on p. 15).

[29] I2C Info. (2015). I2C Bus Specification, [Online]. Available: http://i2c.info/i2c-bus-specification (visited on 20th May 2015) (cit. on p. 16).

[30] Linux Documentation Project. (2014). GPIO Interfaces, [Online]. Available: https://www.kernel.org/doc/Documentation/gpio/gpio.txt (visited on 14th May 2015) (cit. on p. 16).

[31] Texas Instruments Incorporated. (Apr. 2013). LMR14206 SIMPLE SWITCHER - 42Vin, 0.6A Step-Down Voltage Regulator in SOT-23, [Online]. Available: http://www.ti.com/lit/ds/symlink/lmr14206.pdf (visited on 14th May 2015) (cit. on pp. 22, 23).

[32] Silicon Laboratories. (2014). Si7006-A20 - I2C Humidity And Temperature Sensor, [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/Si7006-A20.pdf (visited on 19th May 2015) (cit. on p. 28).

[33] Silicon Labs. (12th Nov. 2013). Direct memory access, an0013 - application note, [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0013.pdf (visited on 12th May 2015) (cit. on p. 31).

[34] Gary P. Scavone, McGill University. (2014). Discrete-Time Signal Metrics, [Online]. Available: http://www.music.mcgill.ca/~gary/306/week12/metrics.html (visited on 30th May 2015) (cit. on p. 31).

[35] Christian Antfolk. (2014). Signal processing and data representation, [Online]. Available: http://bme.lth.se/fileadmin/biomedicalengineering/Courses/Datorbaserade_maetsystem/EEMN10_Lect10_Signal_Processing_CA.pdf (visited on 8th Jun. 2015) (cit. on p. 34).

[36] K. King, D. Orosz, C. Smedberg. (12th Dec. 2012). Reliability Analysis and Counterfeit Detection in Integrated Circuits Using Thermal Imaging, [Online]. Available: http://ecesd.engr.uconn.edu/ecesd173/files/2012/12/Senior-Design-Final-Report.pdf (visited on 26th May 2015) (cit. on pp. 56, 58).

[37] Itseez. (2015). Open Source Computer Vision Library, [Online]. Available: http://opencv.org (visited on 10th Jun. 2015) (cit. on p. 57).

List of Figures

2.1 Point. 4
2.2 Point in exploded view. 4
2.3 Overview of the communication between Point, the back-end and the app. . . . 5
2.4 Basic flow of the test station. 6

3.1 Supply chain. 9
3.2 Production overview. 10
3.3 PCB with manufacturing faults. 11
3.4 The logic of a JTAG enabled device. Reproduced from [17]. 12
3.5 Overview of the I^2C communication protocol [29]. 16

4.1 Test cover overview of some existing test methods [18]. 17
4.2 A block diagram over the test system. 18
4.3 The full schematic of the power supply and current measurement PCB. 19
4.4 The shunt resistors; R3, R5 and R6 controlled by three relays to the left. Two amplifying steps to the right. 20
4.5 Texas instruments reference design of the LMR14206 buck converter [31]. . . . 22
4.6 The scheme of the buck converter. 22
4.7 A block diagram over the most relevant files in the test firmware. 24
4.8 A flow chart of the test firmware. 25
4.9 Parameters for the Silicon Labs Si7006-A20 humidity sensor [32]. 28
4.10 Possible test value deviation in a temperature test. 28
4.11 The block diagram of the microphone. The amplifier adds half VDD as bias to the amplified signal. The input to the ADC is single ended. 30
4.12 The internal MCU setup for the data acquisition. 31
4.13 The relation between precision, trueness and accuracy. Reproduced from [35]. . 34
4.14 The test fixture filled with hardware. 35

5.1 1000 measurements acquired with golden temperature sensor. 37
5.2 Histogram based on the temperature measurements. 38
5.3 Probability plot of the temperature values. 38
5.4 1000 measurements acquired with golden humidity sensor. 39
5.5 Histogram based on the humidity measurements. 39
5.6 Probability plot of the humidity values. 39
5.7 1000 measurements acquired with golden pressure sensor. 40
5.8 Histogram based on the pressure measurements. 40
5.9 Probability plot of the pressure values. 41
5.10 Temperature measurement output of DUT and GS and their margin of error. . . 42
5.11 Relative humidity measurement output of DUT and GS and their margin of error. 42

5.12 Absolute pressure measurement output of DUT and GS and their margin of error. 43
5.13 Difference in temperature measurement output between DUT and GS. 44
5.14 Difference in humidity measurement output between DUT and GS. 44
5.15 Difference in pressure measurement output between DUT and GS. 45
5.16 Output data from the particle sensor test, test run 1. Unit number 11 shows deviant behavior. 46
5.17 The number of zero crossings in the mic samples for all 25 units in the test run. 47
5.18 The average power of the sound samples for all 25 units in the test run. 47
5.19 Current usage in μA during test of DUT with expected current usage. 48
5.20 Current usage in mA during test of DUT with deviant current usage. 49
5.21 Current usage for a DUT with error. 49

6.1 The SUT and GS values from the absolute pressure test. The expected value of the difference SUT-GS is added to all the GS values 53

A.1 Full schematic of the power supply and current measurement PCB. 69

B.1 Reselling used cellphones. 72
B.2 Local Chinese disassembling cellphones. 72

C.1 The debugging setup with a logic analyzer on the I^2C bus. 74

List of Tables

4.1 Low-power mode. 21
4.2 Mid power mode with sensors activated. 21
4.3 High power mode with WiFi activated. 21

5.1 Difference between SUT and GS for temperature. 45
5.2 Difference between SUT and GS for humidity. 45
5.3 Difference between SUT and GS for absolute pressure. 45

A. Current Supply And Measurement Schematic

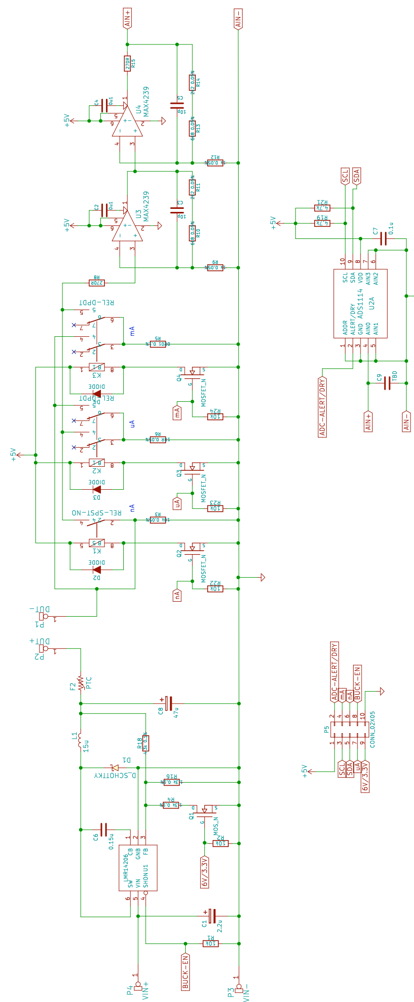


Figure A.1: Full schematic of the power supply and current measurement PCB.

B. Does hardware counterfeit really exist?

As written in section 2.1 counterfeiting is a growing problem in today's electronics. But is this really true?

This master's thesis was mainly performed in Shenzhen, mainland China, close to manufacturers and with easy access to components. The office is located close to multiple multi-storey markets selling electronic components. Each of these markets has hundreds of stands with any kind of components. Some shops are structured and clean, but many of them are a mess. Buying components here is a trade off between fast delivery and the risk of getting unauthentic or incorrect products. The shops do not use computer system to keep track of inventory and sales, and components in the shop gets mixed up with each other. Their products are delivered by people dragging big crates of components on the street. As non locals it is impossible to know where these components comes from and their authenticity.

On another street, every night there is a different kind of market. People coming with big bags of used electronics stuffed on their electrical mopeds and pours out the content on the street. As soon as one of these sellers has opened their bag, a group of buyers are there to scramble on the ground searching for the best products. The products are mostly used and broken cellphones. All buyers hurry to get the specific model of phones they are looking for. Some of them tries to boot the phones with external batteries to ensure that they work. When they have searched through a pile, they move on to the next one and start all over again. Figure B.1 shows some locals searching through a pile of phones.

Where these phones go after being bought on the street we do not know. What we do know is that every morning, on a street just outside our apartment, locals disassembles phones and sorts the parts into different piles as shown in Figure B.2. Presumably these parts will be on the market, blended with new components, later that day.

Figure B.1: Reselling used cellphones.

Figure B.2: Local Chinese disassembling cellphones.

C. Problems and solutions

As with all projects, problems of different importance and difficulty occurred throughout the project time.

C.1 Timing measurements

To control that some of the sensors on the DUT are working, they are being compared with values from the golden sensors. The factory environment are supposedly stable without fast changes in temperature, humidity or pressure. It is still preferred to do measurements on the DUT and GS time synchronized. In the current test station the DUT and GS measurements are synchronized within a few seconds. The MCU on the DUT collect measurement values from the sensors within 3 seconds. The test computer collects data from the GS immediately after the test FW starts executing. This is approximately at the same time, but timing them to do measurements at exactly the same time needs further development. To time the measurements perfectly a synchronization protocol has to be defined. This could be a hardware protocol using the GPIO pins on the DUT and test computer or it could be done in software using breakpoints in the debugging process of the MCU on the DUT.

C.2 Working with software and hardware

Working with hardware and software in parallel, debugging errors are more complicated than when debugging only software or only hardware. If something does not work as expected debugging process are two dimensional. Either the error depends on the recently implemented code, or it depends on the hardware malfunctioning. Locating the error can be a complicated process. Software debugging is an explored field and contains lots of competent software instruments. In hardware tools like oscilloscopes, voltmeters and logic analyzers can be used for debugging. The complicated part of the debugging is to know where to start and what could go wrong in the development platform.

For example; When a sensor test does not work as expected a quite complex debugging process is required. Thanks to a logic analyzer, all I^2C communication can be sampled and analyzed. This makes it easier to understand if the correct data is transferred between ICs on the DUT. An other example is, if the microphone ADC code does not produce the expected results. It could be due to the software code. It could also be a malfunctioning microphone on the development board. A oscilloscope can be used to analyze the electrical signal all the way to the ADC pin on the MCU. If the signal is correct at that point it is probably a software error.

In this project software is implemented to test an already existing hardware platform which is supposed to work correctly. If the hardware is assumed to work, the errors is expected to be in the software. But if the error actually is due to malfunctioning hardware in a platform that is supposed to be working, it is easy to get confused.

Figure C.1: The debugging setup with a logic analyzer on the I^2C bus.

D. Test Firmware Code

D.1 Sensor Example Code

Listing D.1: sensor.c, test code for the pressure sensor. The specific sensor name is replaced with "sensor"

```
#include "board.h"
#include "sensor.h"
#include "i2c.h"
#include "tests.h"

/*
 * Reads pressure and temperature data of a measurement.
 * Stores pressure in hPa and temperature in deg C in
 * test results struct.
 */
void sensor_read_test_data(void)
{
    static uint8_t res[3];

    /*
     * Read an convert sensor data
     */

    /* Read pressure data, 24-bit signed value as 3 8-bit registers */
    i2c_read(SENSOR_ADDR, SENSOR_POUT_XL, res+2, 1);
    i2c_read(SENSOR_ADDR, SENSOR_POUT_L, res+1, 1);
    i2c_read(SENSOR_ADDR, SENSOR_POUT_H, res, 1);

    /* Compute pressure in hPa*/
    int32_t press_data = (res[0] << 24) | (res[1] << 16) | (res[2] << 8);
    test_results.sensor_pressure = (press_data >> 8) / 4096.0;

    /* Read temperature data */
    i2c_read(SENSOR_ADDR, SENSOR_TEMP_L, res+1, 1);
    i2c_read(SENSOR_ADDR, SENSOR_TEMP_H, res, 1);

    /* Compute temperature in deg C */
    int16_t temp_data = (res[0] << 8) | res[1];
    test_results.sensor_temp = 42.5 + (temp_data / 480.0);
}


/*
 * Initializes and configures sensor for a One Shot measurement.
 * Also checks ID of sensor and sets a boolean in test result
 * struct depending if ID is correct.
 */
void sensor_test(void)
{
    static uint8_t res[3];

    /*
     * Read and verify chip id
     */

    i2c_read(SENSOR_ADDR, SENSOR_WHOAMI, res, 1);

    if (res[0] == SENSOR_ID) {
        test_results.sensor_correct_id = true;
    }

    /*
     * Init sensor registers
     */

    /* Reset and reboot chip */
    res[0] = SENSOR_CTRLREG2;
    res[1] = SENSOR_BOOT | SENSOR_SWRESET;
    i2c_write(SENSOR_ADDR, res, 2);

    /* Enable chip and interrupts */
    res[0] = SENSOR_CTRLREG1;
```

```
    res[1] = SENSOR_PD | SENSOR_DIFF_EN;
    i2c_write(SENSOR_ADDR, res, 2);

    /* Set interrupts on data ready */
    res[0] = SENSOR_CTRLREG4;
    res[1] = SENSOR_P1_DRDY;
    i2c_write(SENSOR_ADDR, res, 2);

    /* Enable interrupts on INT_BAR GPIO pin on MCU */
    GPIO_IntClear(1 << GPIO_PIN_INT_BAR);
    GPIO_IntEnable(1 << GPIO_PIN_INT_BAR);

    /* Configre One Shot */
    res[0] = SENSOR_CTRLREG2;
    res[1] = SENSOR_ONE_SHOT;
    i2c_write(SENSOR_ADDR, res, 2);
}
```

Code for the pressure sensor is presented. The sensor.h file includes the register definitions for the sensor.

D.2 ADC Code

Listing D.2: sensor.c, test code for the pressure sensor. The specific sensor name is replaced with "sensor"

```
#include "adc.h"
#include "dma.h"
#include "board.h"
#include "tests.h"
#include "sensor/acoustic.c"

static void microphone_dma_complete(unsigned int channel, bool primary, void *user);

static DMA_CB_TypeDef microphone_cb = {.cbFunc = microphone_dma_complete};
//static task_t *task_cb;
static bool single_shot, stop, stream;

struct adc_mic_buf {
    uint8_t len, op;
    uint16_t data[MIC_SAMPLES];
} __attribute__((packed));

struct adc_mic_buf mic_buf[2];

uint16_t *adc_last_buffer, *single_result;
uint8_t *adc_last_packet;

/*
 * Callback function for the DMA. When a sample buffer is filled
 * this function gets invoked. If the sampling is configured to a ping-pong
 * (multiple) run the function refreshes the DMA to handle new ADC samples, otherwise it stops.
 */
static void microphone_dma_complete(unsigned int channel, bool primary, void *user)
{
    adc_last_buffer = primary ? mic_buf[0].data : mic_buf[1].data;
    adc_last_packet = primary ? &mic_buf[0].len : &mic_buf[1].len;

    if (single_shot || stop) {
        ADC_Reset(ADC0);

        /* Turn off frontend */
        GPIO_PinOutSet(GPIO_PORT_SOUND_PWRON, GPIO_PIN_SOUND_PWRON);

    } else {
        DMA_RefreshPingPong(channel, primary, false, NULL, NULL, MIC_SAMPLES-1, false);
    }

    /* Single fix, needs averaging for ping-pong.
     * Compute e, zcr and f for samples and store in test results
     */
    int16_t *p = (int16_t *)adc_last_buffer;
    uint32_t e, zcr;

    static int timer = 0;

    remove_bias(p, adc_last_buffer, 64);
    test_results.sound_e = ste(p, 64);
    test_results.sound_zcr = stazcr(p, 64);
    test_results.sound_freq = (float) 1/(((1/8000.0)*64.0)/(test_results.sound_zcr/2));
}
```

```
/*
 * Initializes the DMA
 */
static void adc_dma_init(void)
{
    DMA_CfgChannel_TypeDef chnlCfg;
    DMA_CfgDescr_TypeDef descrCfg;

    chnlCfg.highPri = false;
    chnlCfg.enableInt = true;
    chnlCfg.select = DMAREQ_ADC0_SINGLE;
    chnlCfg.cb = (DMA_CB_TypeDef *) &microphone_cb;
    DMA_CfgChannel(DMA_CHANNEL_MICROPHONE, &chnlCfg);

    descrCfg.dstInc = dmaDataInc2;
    descrCfg.srcInc = dmaDataIncNone;
    descrCfg.size = dmaDataSize2;
    descrCfg.arbRate = dmaArbitrate1;
    descrCfg.hprot = 0;
    DMA_CfgDescr(DMA_CHANNEL_MICROPHONE, true, &descrCfg);
    DMA_CfgDescr(DMA_CHANNEL_MICROPHONE, false, &descrCfg);
}

/*
 * Initializes the ADC with the DMA to store data in RAM
 * and Timer0 to trigger new ADC samples
 */
void adc_init(void)
{
    CMU_ClockEnable(cmuClock_ADC0, true);
    CMU_ClockEnable(cmuClock_PRS, true);

    mic_buf[0].len = sizeof(uint8_t) + MIC_SAMPLES * sizeof(uint16_t);
    mic_buf[1].len = sizeof(uint8_t) + MIC_SAMPLES * sizeof(uint16_t);

    adc_dma_init();

    ADC_Init_TypeDef adcCommonInit = {
        .ovsRateSel = adcOvsRateSel16,
        .lpfMode = adcLPFilterDeCap,
        .warmUpMode = adcWarmupNormal,
        .timebase = ADC_TimebaseCalc(0),
        .prescale = ADC_PrescaleCalc(12000000, 0),
        .tailgate = false,
    };

    ADC_Init(ADC0, &adcCommonInit);
    ADC_IntDisable(ADC0, _ADC_IF_MASK);
    ADC_IntClear(ADC0, _ADC_IF_MASK);
    ADC_IntEnable(ADC0, ADC_IF_SINGLEOF);
    NVIC_EnableIRQ(ADC0_IRQn);

    /* Timer0 overflow is our sample clock */
    PRS_SourceSignalSet(0, PRS_CH_CTRL_SOURCESEL_TIMER0, PRS_CH_CTRL_SIGSEL_TIMER0OF, prsEdgeOff);
}

/*
 * Triggers a new ADC sample run.
 *
 * @param bool single - If true, the system runs a single run with one buffer,
 *                      else it does a continuous ping-pong run.
 */
void adc_sample_sound(bool single)
{
    /* Turn on frontend */
    GPIO_PinOutClear(GPIO_PORT_SOUND_PWRON, GPIO_PIN_SOUND_PWRON);

    ADC_InitSingle_TypeDef adcSingleInit = {
        .prsSel = adcPRSSELCh0,
        .acqTime = adcAcqTime1,
        .reference = adcRefVDD,
        .resolution = adcRes12Bit,
        .input = ADC_SOUND_CHANNEL,
        .diff = false,
        .prsEnable = true,
        .leftAdjust = false,
        .rep = false,
    };

    ADC_InitSingle(ADC0, &adcSingleInit);

    single_shot = single;
    stop = false;
    stream = true;

    if (single) {
        DMA_ActivateBasic(DMA_CHANNEL_MICROPHONE, true, false,
                          (void *) mic_buf[0].data,
                          (void *)&(ADC0->SINGLEDATA),
                          MIC_SAMPLES - 1);
    } else {
        DMA_ActivatePingPong(DMA_CHANNEL_MICROPHONE,
                             false,
                             (void *) mic_buf[0].data,
                             (void *) &(ADC0->SINGLEDATA),
                             MIC_SAMPLES - 1,
                             (void *) mic_buf[1].data,
                             (void *) &(ADC0->SINGLEDATA),
```

```
                                MIC_SAMPLES - 1);
    }
}
```

D.3 I²C Code

Listing D.3: sensor.c, test code for the pressure sensor. The specific sensor name is replaced with "sensor"

```
#include "i2c.h"

/*
 * Initialize the i2c.
 */
void i2c_init(void)
{
    CMU_ClockEnable(cmuClock_I2C0, true);
    I2C_Init_TypeDef i2c_init = {
        .enable = true,
        .master = true,
        .refFreq = 0,
        .freq = 400000,
        .clhr = i2cClockHLRAsymetric,
    };
    I2C_Init(I2C0, &i2c_init);
    I2C0->CTRL |= I2C_CTRL_AUTOACK;
    I2C0->ROUTE = I2C_ROUTE_LOCATION_LOC0 | I2C_ROUTE_SDAPEN | I2C_ROUTE_SCLPEN;
}

/*
 * Trasfers the i2c data, returns when transfer complete.
 */
void i2c_xfer(I2C_TransferSeq_TypeDef *xfer)
{
    I2C_TransferReturn_TypeDef ret;

    ret = I2C_TransferInit(I2C0, xfer);
    while (ret == i2cTransferInProgress)
    {
        ret = I2C_Transfer(I2C0);
    }
}

static I2C_TransferSeq_TypeDef xfer;

/*
 * Write data to the i2c bus.
 *
 * @param addr - The slave address to write to.
 *
 * @param reg - Slave register to write.
 *
 * @param val - The data to write.
 *
 * @param len - Length of the val array
 */
void i2c_write(uint8_t addr, uint8_t reg, uint8_t *val, size_t len)
{
    static uint8_t tx_reg;
    tx_reg = reg;

    xfer.addr = addr << 1;
    xfer.flags = I2C_FLAG_WRITE_WRITE;
    xfer.buf[0].data = &tx_reg;
    xfer.buf[0].len = 1;
    xfer.buf[0].data = val;
    xfer.buf[0].len = len;

    i2c_xfer(&xfer);
}

/*
 * Read data to the i2c bus.
 *
 * @param addr - The slave address to read from.
 *
 * @param reg - Slave register to read.
 *
 * @param val - A pointer to where to store the result.
 *
 * @param len - Length of the val array
 */
void i2c_read(uint8_t addr, uint8_t reg, uint8_t *val, size_t len)
{
    static uint8_t rx_reg;
    rx_reg = reg;

    xfer.addr = addr << 1;
    xfer.flags = I2C_FLAG_WRITE_READ;
    xfer.buf[0].data = &rx_reg;
    xfer.buf[0].len = 1;
    xfer.buf[1].data = val;
    xfer.buf[1].len = len;
```

```
    i2c_xfer(&xfer);
}
```