Embedded functional testing using LabVIEW

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg

LTH Ingenjörshögskolan
Lunds Universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
E-husets tryckeri
Lund 2017

Abstract

The past decade, advancements in semiconductor technology as well as the construction of large data centers around the world have given rise to the possibility to connect anything to the internet. In today's electronics industry, "Internet-of-things"-applications are in a huge demand where companies are being subject to massive competition. Given this reality, resources within a small development company need to be managed carefully and optimally. In this project, LabVIEW is used to develop a rig for automation of the previously time-consuming task of testing. The project has resulted in the benefits of faster test throughput and a more systematic testing procedure, enabling both development and sales staff to make faster and more confident decisions about if the product is ready to go when the customer needs it. The basis for a more comprehensive and statistical test is also laid due to the modular approach employed in the system design and software suite.

Keywords: Internet of things, functional testing, embedded electronics

Sammanfattning

Det senaste decenniet har framsteg inom halvledarteknik samt konstruktionen av stora datacentraler runt om i världen lett till möjligheten att ansluta mer eller mindre vad som helst till internet. I dagens elektronikindustri är "Internet-of-Things"-applikationer i stor efterfrågan där företagen utsätts för massiv konkurrens. Med tanke på denna verklighet behöver resurser inom ett litet utvecklingsföretag hanteras noggrant och optimalt. I detta projekt använder författaren LabVIEW för att utveckla en rigg med målet att automatisera den tidigare tidskrävande uppgiften att testa producerade enheter. Projektet har resulterat i en snabbare test genomströmning och ett mer systematiskt förfarande, vilket gör det möjligt för både utvecklings- och säljpersonal att fatta snabbare och säkrare beslut om att produkten är redo att släppas när kunden behöver det. Grunden för ett mer omfattande och statistiskt test har också lagts i och med det modulära tillvägagångssättet som används i systemdesignen och programvaran.

Nyckelord: Sakernas internet, funktionstestning, inbyggda system

Acknowledgements

The author wishes to acknowledge the opportunities given by JEMAC AB with its affiliates; Stefan Jakobsson, Eva Johansson, Torbjörn Carlqvist, Jens Lorentzson and Ingemar Jaegtnes. Thanks for the learning experience, interesting discussions, and a friendly workplace environment.

Thanks to my examiner, Professor Christian Nyberg for the help in writing this report.

Thanks to fellow students, Martin Erlandsson and Fredrik Johansson for the constructive feedback on the report.

Special thanks to Professor Mats Lilja, who with his passion for teaching and personal engagement has inspired to continue learning about the electromagnetic phenomena.

Thanks to Christoffer Wernersson for providing housing during the time in Kalmar.

Abbreviations

ADC	:	Analog-to-digital converter
ANSI	:	American National Standards Institute
BIST	:	Board In-System Test
CAN	:	Controller area network
CMOS	:	Complementary metal oxide semiconductor
DAC	:	Digital-to-analog converter
ESD	:	Electrostatic discharge
FFT	:	Fast Fourier transform
GNSS	:	Global navigation satellite system
I2C	:	Inter-integrated circuit
IC	:	Integrated circuit
IEEE	:	Institute of Electrical and Electronics Engineers
IMEI	:	International Mobile Equipment Identity
IoT	:	Internet of things
LDO	:	Low-dropout regulator
MCU	:	Microcontroller
NI	:	National Instruments
NMEA	:	National marine electronics association
OtG	:	On-the-go
PCB	:	Printed circuit board
PCI	:	Peripheral component interconnect
PHY	:	Physical layer
PTP	:	Precision time protocol
RTC	:	Real-time clock
RTCA	:	Radio Technical Commission for Aeronautics
RX	:	Receive
SAR	:	Successive approximation
SIM	:	Subscriber identity module
SPI	:	Serial peripheral interface
SSL	:	Secure sockets layer
TCP	:	Transfer control protocol
TTL	:	Transistor-transistor logic
TX	:	Transmit
UART	:	Universal asynchronous receiver transmitter
UDP	:	User datagram protocol

Table of contents

1 Introduction 2
1.1 The client 2
1.2 Background 2
1.3 Objectives and goals 3
1.4 Problem specification 4
1.5 Source criticism 4
1.6 Limitations 5
1.7 Resources 6
2 Technical background 8
2.1 The gateway 8
2.1.1 Overview 8
2.1.2 MCU board 9
2.1.3 INT board 11
2.2 Features 13
2.2.1 RS-232 13
2.2.2 RS-485 13
2.2.3 CAN-bus 13
2.2.4 USB on-the-go 14
2.2.5 Accelerometer 14
2.2.6 Modem 14
2.2.7 GNSS 15
2.2.8 ADC 15
2.2.9 DAC 15
2.2.10 Digital I/O 16
2.2.11 Ethernet 16
2.2.12 Wake-up and sleep modes 16
2.3 Software 17
2.3.1 LabVIEW 17
2.3.2 Teststand 22
2.3.3 BIST 22
2.3.4 J-COM 22
3 Method 24
3.1 Clients requirements 25
3.2 Test system requirements 26
3.3 Deciding on hardware 28
3.3.1 Velleman PS3005D 28
3.3.2 LabJack U12 29
3.3.3 FTDI USB-to-RS232 and USB-to-RS485 adapters 30
3.3.4 Plexgear USB-hub 30
3.3.5 Luxorparts 8-channel relay module 31

3.4 Planning the test sequences 31

4 Implementation 34
4.1 Setting up hardware 34
4.2 Setting up communication 36
4.3 Building Sub-VI's 37
4.3.1 Serial communication VI 37
4.3.2 Analog measurement VI 42
4.4 Building the interface 46
4.5 Setting pass limits 47
4.6 Automating the process 49

5 Verification 52
5.1 Provoking errors 52
5.2 Measurements 52

6 Results 54
6.1 Goals of the project 54
6.2 Reliability 54
6.3 Time-savings 55
6.4 Ease of use 55
6.5 Risk 56
6.6 Omitted features 56

7 Conclusions and possible improvements 58
7.1 Terminal blocks 59
7.2 Battery simulator 59
7.3 Casing and wiring 60
7.4 Usage of extra I/O 60
7.5 ESD protection 60

8 References 62

Appendix A 66

Appendix B 67

Appendix C 68

Appendix D 69

1 Introduction

This chapter contains an introduction of the client, a description of the project background, the objective and goals, the problem specification and project limitations. A technical background is presented in the following chapter, followed by a test requirements specification, and further with an implementation description and verification description of the test system. Finally, results and possible improvements are discussed.

1.1 The client

The work is performed on behalf of JEMAC Sweden AB, which is a company working with the design and development of electronics in the Internet of Things field. They also provide consultant services. Over the past year, the company has developed a new generation of "IoT Gateways" for connecting machines to the Internet. The gateway collects, processes and makes data available for further use in different cloud services. JEMAC caters to the global market for enterprise solutions. End customers are companies interested in using the cloud and in need of the hardware interface to do so. [1]

1.2 Background

Currently, the gateway is in its third incarnation. Designed as a simple prototype with basic functions, the first version meant to serve as a proof-of-concept. After probing the market, JEMAC quickly realized the need for a modular approach due to the different requirements of the customers. Engineers improved the flexibility by developing a single platform that can be tailored both economically and technically and thus made the second and later third gateway designs more suitable to the market demands. In general, large bulky components such as the RJ45-connector is relatively expensive. Together with the PHY-chip, isolation transformers and other minor components associated with for example Ethernet, the cost for the feature becomes significant when producing in large scales. At the same time, the cost per unit of PCB is significantly lowered when ordering in bulk, and by doing so, it can be shared between many customers. Simply not mounting components while using the same PCB’s therefore reduce cost and make the gateway more attractive to customers with tighter budgets, that do not wish to take advantage of various features.

Gateway model 3 is in the final stages of prototyping and is becoming mature enough to be released. Tests are conducted at various stages of development projects and in production to ensure fulfillment of the product specification. One of the tests that the units go through is a so-called functional test; this tests all the product features at a basic level. The company would like to improve the efficiency of this process. Currently, the whole testing process takes a significant amount of time, as it is performed manually. Automating the testing procedure means time-savings and a freeing up of resources to be deployed in other areas to make better use.

1.3 Objectives and goals

To develop and build a test system used for product development and testing of complete products is the purpose of this project. It aims to standardize and streamline testing. The goal is to develop a system with the following objectives:

1. Develop a test requirements specification based on the existing product specification and test instructions for the current manual test and through review and dialogue with the hardware designer.

2. The design of a hardware system based on PC / Windows, with analog and digital interfaces, communication interfaces, and a fixture for the units.

3. Develop the test software's user interface, test sequence, communications and storage of settings and data.

4. As the project finishes, verify that the test system works as planned and document the project.

The task at hand involves both hardware and software understanding, theoretical and practical training, and fulfills a real need for the client. It puts to use knowledge acquired from a large part of the education.

1.4 Problem specification

These initial questions were asked before beginning work on the project. In the planning and design phase, they are considered, and the actions following is made with the goal to resolve them.

1. How should the testing be performed?
2. What are the requirements for each parameter to be considered OK?
3. What parts should constitute the test system?
4. What are the requirements of the test system hardware?
5. How should information be shown in the user interface?
6. How should settings and data be stored?

1.5 Source criticism

The informational sources used in the research for the project has been mainly datasheets and application notes for the components used in the gateway. White papers have been used to provide information about some of the topics, being aware of the marketing angle of these. They still provided much information. In some cases, other informational sources from the internet have been used. These have been compared against each other for consistency among them. As the electronics industry heavily relies on standards, websites that provide information for educational purposes is considered not to have incitement to falsify the content.

References 3, 5, 8, 10 and 16 are white papers. They are in some cases angled to market the author's product or persuade the reader to use a specific technology. Still they provide much information that is considered valid. Many times, white papers are not written for marketing, but instead written by standardization organizations such as IEEE. The white papers used in this report are from well-known manufacturers with a good reputation in the industry that should not be able to afford to sell products that is not performing as intended.

Datasheets has been consulted in the case of 4, 7, 11, 15, 17 and 26. For the same reasons as white papers, they are also considered trustworthy. The creation of datasheets is assumed to be preceded by detailed measurements of the specifications, and experience with using datasheets give confidence of the validity content of these.

1, 12, 13, 21 and 23 are official company webpages and mainly used for the reason of giving the reader ability to read more about the products used in the project.

18, 19, 20, 24 and 25 are references to vendors, given so that the reader can look for current prices and availability of the products used in the project.

References number 6, 14 and 22 come from webpages advertising themselves as independent informational sources with articles written and peer reviewed by experts.

Reference 9 is a handout from a lecturer at Berkley, a prestigious university which is assumed to have qualified lecturers providing valid information.

Another major informational source has been the LabVIEW forum; reference number 2. Since the author had no previous experience with this software and had to learn it quickly due to the limited time available, this forum was frequently visited. Both enthusiasts and professionals write extensively on the forum. Hence the amount of content was very large, but the quality of the provided information was varying. Aware of this, it was required to discard some of the ideas presented there, trying different methods. Overall, the forum proved to be a good source, however.

All the above is considered valid sources and has since been used for the project research.

1.6 Limitations

To eventually develop a general system that can adapt to different variants of gateways is JEMAC's intention. The project will take this into account if possible but is limited to developing a system for the third generation of gateways as the first prototype.

After production of new boards, a verification test is carried out by the hardware designer. Verification testing is detailed, as it is supposed to verify that the circuit design is performing as intended. Any problems are documented to be, if possible, corrected in the following production runs. As a part of this process, the functionality is indirectly tested, but only on one or a few units.

The automated tests that the project intends to implement differ from the verification test. They are to be performed on every shipped unit in the basic form of “pass/fail”, as a final check that everything is working; not rooting the cause of eventual failures.

Only the electronic parts of the gateway will be included in the technical background and not the mechanical aspects of the system.

The project is carried out single-handedly over a limited amount of time and due to these constraints, planned features might be omitted.

1.7 Resources

Resources in the form of hardware and software engineers will support the work. JEMAC will provide all equipment in the form of gateways, computers, measurement systems, and software.

2 Technical background

This chapter aims to describe the technical background of the project, starting with a description of the gateway itself, and continuing with information about the features that should be tested. Lastly, an introduction of the software used in the test system is given.

2.1 The gateway

The gateway's main components are described; an overview and description of the two boards.

2.1.1 Overview

Figure 1: Overview of the gateway assembly.

Figure 1 illustrates the assembled gateway. Top view shows a general-purpose button and three programmable LED indicators. External GNSS and RF SMA antenna connectors are visible in the top right picture. There is also the option to use internal u.FL antennas; in that case, the hole is covered with a plate. Bottom left is the terminal block interface to various devices and sensors. Bottom side right shows the RJ45 Ethernet and micro-USB connectors. The case is made in polyamide using a professional 3D-

printer, but when the gateway reaches mass production, press-formed plastic will be used.

Internally, in its basic configuration, the gateway consists of two PCB's. One board holds the microcontroller and modem and the other board provides the power circuitry and interface to the terminal connections. These boards are referred to as the MCU- and INT-board (interface board) in the following text. On the INT-board, there is also two headers to connect so-called "add-on boards", with the purpose to extend and further customize the design. The add-on boards are not a part of the core gateway design and will not be discussed, but the following text will describe the MCU- and INT-boards in more detail.

2.1.2 MCU board

Figure 2: Top view of the MCU board.

The MCU-board is a smaller PCB connected to the larger INT-board via a mini-PCI connector. The connector can be seen in the bottom of figure 2. Modularity and flexibility are considered important aspects of the gateway design and this split configuration makes it possible to use the core MCU with future products and product variants.

The gateway features 120MHz ARM Cortex-M4 based Kinetis K64 microcontroller from NXP, chosen because it provides a sound basis with many desired functionalities already embedded in the processor itself, reducing the need for external circuitry. The MCU utilizes an external 4MB SPI flash memory for storing firmware.

Mounted on the PCB is also NXP's MMA8652 accelerometer, as well as a PCF85263 real-time clock. More on these in chapter 2.2.

A SARA-U2-201 modem, from U.Blox, used in conjunction with a MAX-M8 GNSS module, provides for cellular connection and positioning needs. Mounted with a full-spec modem, the gateway can have worldwide mobile network coverage and use all different positioning systems. The choice to use different spec modems enhance price dynamics. The modem utilizes a passive antenna, but the GNSS chip requires the use of an active to receive the faint satellite signals.

Two of the K64's six UART's have been dedicated to external interfaces while the other four is being used internally to communicate with the modem and the extension boards. Besides this, I^2C-multiplexers is used to expand the MCU I/O pins to allow for more signal pathways.

For analog interfacing, there is the K64's two 16-bit ADC's and two 12-bit DAC's. They provide conversion between analog signals and the digital domain. High precision is possible, especially with the 16-bit ADC. There is custom circuitry between the signals and the MCU's pins. More on this in chapter 2.1.3 about the INT-board.

The K64 embeds a USB-OtG controller, which enables the gateway to act both as a host and slave when connected to other "smart" devices. It supports USB 3.0 that specifies transfer speeds up to 10Mbit/s, so it makes up a yet another, very fast, communication channel to other devices.

Ethernet capability is supported internally of the K64 with the maximum bandwidth of 100Mbit/s. A physical layer chip interfacing with the network is placed on the INT-board.

2.1.3 INT board

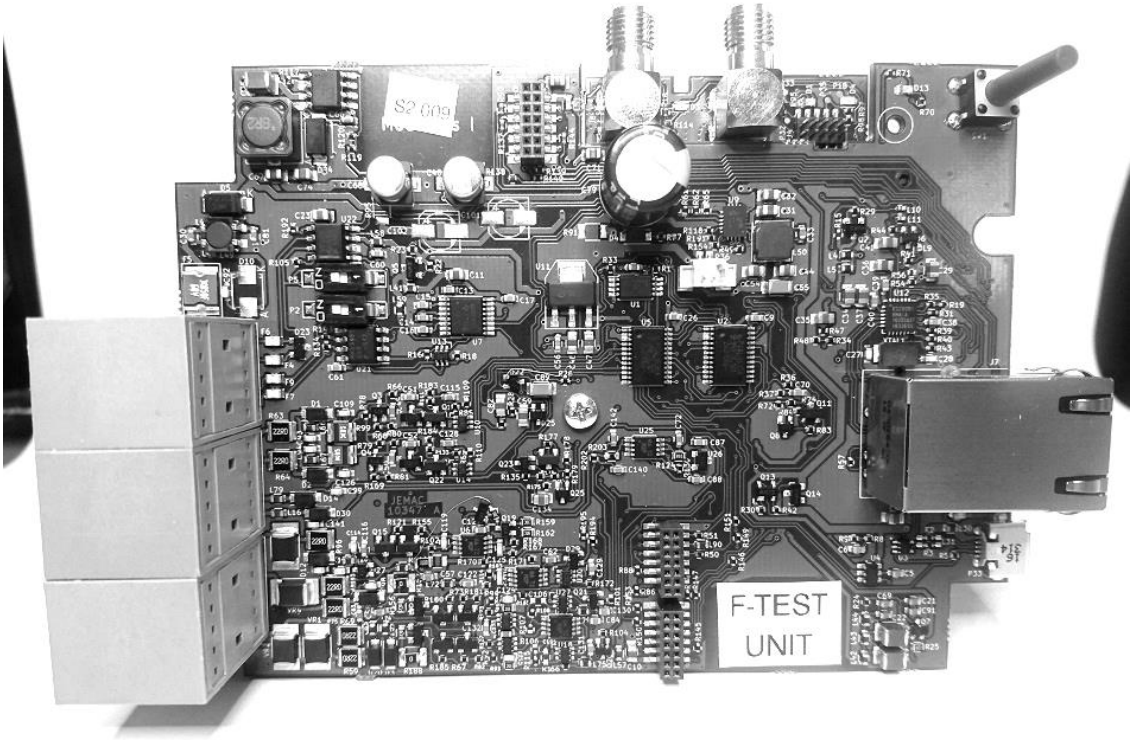


Figure 3: Top view of the INT board.

Figure 3 shows the top view of the interface board. The power supply circuit is placed here. Figure 4 shows a block diagram of the setup. The chain begins with a filtered 24 V nominal input into a DC/DC converter that steps the voltage down to 5 V. The input voltage also provides power to the custom ADC/DAC circuits. USB power input also connects to the 5 V rail via back-to-back diodes. Then this rail connects to a battery charger which besides charging the battery with 400 mA of current, regulates the output to 4.05 V. On battery power, however, the rail follows battery voltage. A lithium-ion battery is used to store the energy, capable of providing 5400 mAh to the gateway. The final conversion step is a precision LDO that outputs the 3.3 V rail. Control signals both directly from the advanced battery charger and from rail-dedicated ADC's allow the MCU to monitor power consumption and charging status.

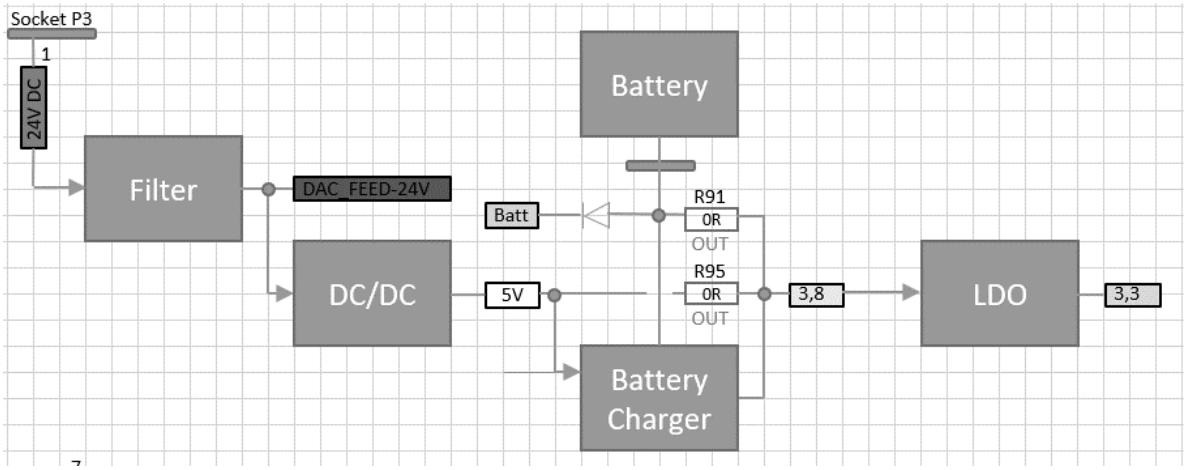


Figure 4: Power supply block diagram.

All the chips to drive the interfaces is placed on the INT-board; the RS-232 and RS-485 line drivers, Ethernet PHY and CAN bus. It also hosts the DAC interface which consists of a circuit with many components, thus placed on the bigger board, as well as the circuit for the ADC and digital I/O.

Several analog switches and I/O-expanders used for the many lines to and from the MCU-board are also present on the board. They are beside internal duties used for switching between input and output of the digital interfaces and switch the ADC's input between general analog signals and specially constructed thermistor inputs. The ADC’s can also be switched by the expanders to read either voltage or current signals.

There are several pin headers on the board as well. They are intended to be used with add-on modules built to the customer's needs. For example, one customer might want to use Wi-Fi as local wireless network, while another one wants to use Bluetooth for the same purpose. Using add-ons, less development and time is needed to provide a solution that just requires a particular module, than it is to redesign the whole INT-board.

2.2 Features

Even though the reader is assumed to be familiar with many of the features that are listed below, for completeness, the text provides some general information about these. Some features are described in more detail with specifications for each component.

2.2.1 RS-232

The gateway is compliant with the RS-232 standard and uses a Maxim MAX232 line driver chip to perform the logic level conversion between the UART pins on the MCU and the terminal block on the INT-board. A three-wire version is used, with only RX, TX and GND signals. The board features two ports where one of them is used for debugging in the development phase. [3][4]

2.2.2 RS-485

RS-485 is a standard that defines use of differential voltages, and thus possesses common mode rejection and a better noise immunity than RS-232. It is widely used in industrial applications. The standard's specification allows for up to 32 devices to be connected to the bus; this is an advantage over RS-232 that broadens the usage possibilities considerably. Maximum cable length is also extended to about 1200 meters, at data rates up to 10Mbit/s. RS-485 is implemented in the gateway using a Maxim MAX487 chip. [5][6][7]

2.2.3 CAN-bus

The bus is commonly associated with the automotive industry but is being very much used in other areas as well. The protocol has a light software stack which makes it favorable for programmers in terms of memory restrictions. CAN use frames that are being distributed across all nodes of the network, using CSMA for collision detection. Error correction is achievable as well, and the bus has high noise immunity. The gateway supports a CAN high-speed ISO standard that specifies transfer rates up to 500kBit/s. [8][9]

2.2.4 USB on-the-go

Originally included in the second USB specification, USB "on-the-go" provides a way for devices to act both as host and slave. Communication rates in the standard is specified as 480Mbit/s. The K64 MCU supports an even later version (USB 3.0) which further improves data rates and power delivery. [10]

2.2.5 Accelerometer

An MMA8653 accelerometer provides the functionality for detecting orientation and changes in motion of the gateway. Manufactured by NXP, the tiny DFN footprint chip uses capacitive sensing technology to do this. Resolution of the device is 10 bits; data is accessible via I2C together with lots of embedded functions. It also supports several power-saving modes with reduced resolution and update frequency. Digital registers are used to make settings, and the accelerometer works pretty much standalone; no data processing needs to be done by the MCU. Two individual interrupts can be programmed to inform the MCU of changes in acceleration and position. [11]

2.2.6 Modem

Two devices from U.Blox are used; one of these is the SARA-U2-201. It is currently being advertised as "the world's smallest" modem. Multiple versions of the same modem share the same footprint, making regional adaptions of the gateway easy. The modem supports IPv4, IPv6, embedded TCP and UDP, SSL and the use of eSIM. It operates at an extended temperature range of -40 to +85ºC. It also supports direct utilization of U.Blox GNSS modules that can be used as a hybrid positioning scheme together with the modems own CellLocate technology. [12]

2.2.7 GNSS

The other U.Blox module used is the MAX-M8 GNSS. The module can simultaneously use three of the four available GNSS systems; GPS, Galileo, GLONASS, and BeiDou. When started cold, the module is specified to lock on to satellites within 28 seconds. Reacquisition time is claimed to be 1 second on lost signal. It can output NMEA or RTCM strings for further processing. [13]

2.2.8 ADC

The gateway utilizes the K64's built in ADC's, which is of a successive approximation type; they use a binary search method, comparing the input voltage against a decreasing reference for each bit of the converter. Therefore, an N-bit converter requires N comparisons, making increased precision achieved at the price of speed. As can be seen in figure 5, SAR technology covers a mid-range of this sample rate/resolution relationship. The K64's ADC has 16 bits which give a resolution of approximately 38 uV per bit on a 2,5 V reference. [14]

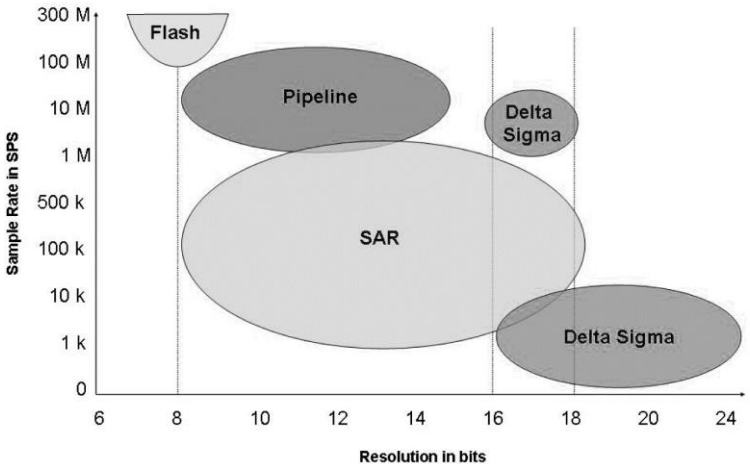


Figure 5: Types of ADC's [13]

2.2.9 DAC

There are many types of DAC techniques, and unfortunately, it has proven hard to provide information about which type the K64 uses. The datasheet and NXP application notes give no information about this. It is possible that the DAC can use different types or modes, but the underlying hardware cannot be determined. It is known, however, that there is both 6-bit and 10-bit DAC's available to the user. [15]

2.2.10 Digital I/O

The digital I/O portion of the gateway is a custom circuit that can be switched between input and output by a signal from the MCU to an analog switch IC, that manipulates the signal paths on the PCB. A maximum of 24 V signals is supported; 3.4 V being the threshold for a low signal and 4 V for a high. Voltages between 3.4 and 4 are undefined. The circuit uses an op-amp to compare the voltage-divided input against a 0,7 V reference for signal detection and use an open-collector output for outgoing signals.

2.2.11 Ethernet

The K64's embedded controller supports 10/100Mbit Ethernet with the IEEE 1588 standard which allows for external clock synchronization (PTP). It makes it possible to achieve microsecond time precision that is needed at high transfer speeds; doing this with minimum computational performance. The PTP master clock can be a based on a GNSS receiver that passively utilizes the atomic clocks used for timing within the positioning system. [15][16]

2.2.12 Wake-up and sleep modes

Implemented using a fairly complex custom circuit is the different sleep modes of the gateways, built to provide the ability to lower the current consumption of the gateway to increase battery life. The circuit does not rely on the MCU's built-in sleep modes. Instead, it is built so that the MCU can disconnect the power to itself. When a wake-up event occurs, power to the MCU is restored. Also, the modem, analog circuitry, and flash memory can be disabled in software before sleep. That also saves power. When in sleep mode, the unit consumes below 250uA, which is equivalent to more than a year's on-time using the provided 5600mAh battery. Button press, RTC, digital input and DC supply connection can trigger a wake-up event.

2.3 Software

The software that was used in the project is described here.

2.3.1 LabVIEW

LabVIEW version 16.0 almost entirely makes up the software part of the project. It is a graphical programming tool developed by National Instruments, and instead of writing code, one uses building blocks to program. LabVIEW shows many similarities to the MATLAB plugin Simulink. NI claims that drag and drop features improves the workflow and makes it easier to go from thought to result. The software is very advanced, but this project utilizes only the most basic functionalities. NI also provide hardware interfaces to support the software and has an extensive catalog with prices spanning across a range of a few thousand to hundreds of thousand SEK. Precision ADC's, frequency analyzers, PID controllers and lots of other modules is available. A lot of third-party manufacturers include support for LabVIEW. There is a large community forum online where much information was acquired when learning about the software. [2]

Like most project based software, when starting LabVIEW one is introduced to a project manager where all access and organization of project files happens. Fundamental to LabVIEW is something that NI calls "virtual instruments" or "VI's". These are software equivalents of traditional hardware instruments, designed to take advantage of a modern computers processing power and connectivity for measurements and analysis. When they are created, LabView generates a linked file pair consisting of a "front panel"-file and a "block diagram"-file. Working with the VI's one creates functions, with inputs and outputs, in the block diagram; that then generates input options and indicators in the front panel. The block diagram is where most of the work is done, however. Figure 6 shows the two files; the reader is encouraged to study the relationship between them.

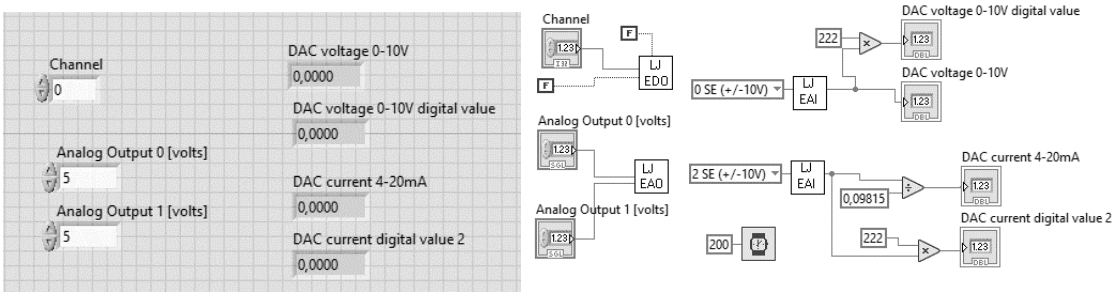


Figure 6: The front panel, block diagram pair

The following text will present functions that were used in the project, and many screen-dumps is shown to illustrate them. These dumps are somewhat edited to save space, and to improve the readability of the report.

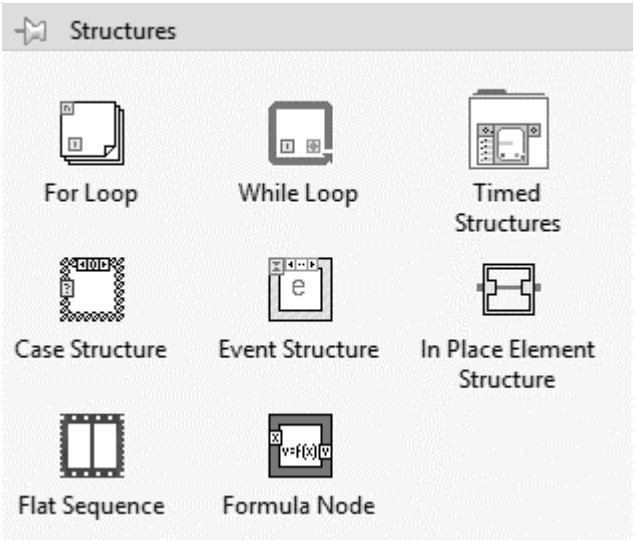


Figure 7: The structures palette

"Palettes" give access to functional blocks. One example of such can be seen in figure 7. The "structures" palette is fundamental as it provides basic programming structures. It contains the for-loop and while-loop, which is essential functions in any programming language. Figure 8 and figure 9 shows them in the graphical LabVIEW form when used in a block diagram.

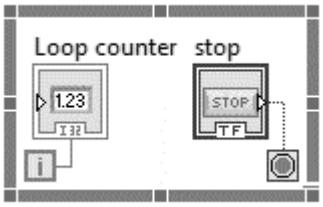


Figure 8: The basic WHILE-loop

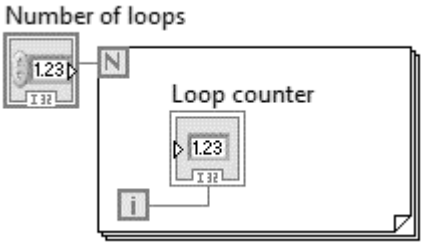


Figure 9: The basic FOR-loop

The figures also illustrate the "wires" that is used to connect blocks together. The stop block in the while example is a Boolean input. It will show up in the front panel as a switch that the user can interact with. Both an integer input and output is displayed in the for example. They will appear as an input "ticker" and an indicator field on the front panel. LabVIEW uses colors to mark the block type, although not seen in this print.

Inside of the structures, other blocks can be placed. Booleans, integers, doubles, strings, other functional blocks, and even other block diagrams. It is at the core of LabVIEW, the possibility to create functions within functions.

For example, saving the for loops block diagram (and its corresponding front panel) will create a "Virtual Instrument", VI. This loop-VI can then be placed in another VI's block diagram, connecting the two and making the loop-VI's "Number of loops" input and "Loop counter" output available to the second VI. That is, the inputs and outputs of a "sub-VI" can be controlled and read from a higher-level VI. The stacking can be done indefinitely and create very complex constructions.

The connection between the two is made by routing controls and indicators (inputs and outputs) to "wire connection points". The act of routing is performed in the front panel by clicking on the colored fields and then choose to associate them with the desired object. Figure 10 tries to illustrate this. Objects that has been routed will then shows up as "wire connections" when the VI is used inside of another VI. Figure 11 shows the block that becomes of the newly routed sub-VI. The same objects that were routed becomes available to connect whatever that is desired; in the example buttons and indicators.

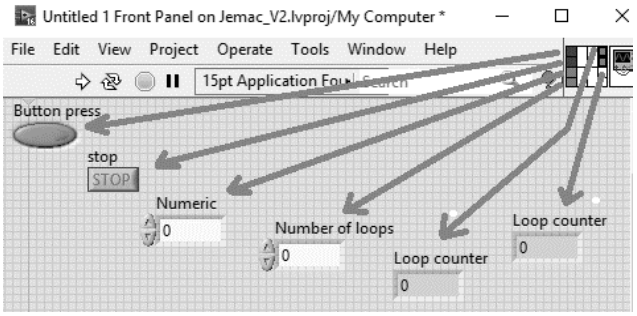


Figure 10: Routing in the front panel

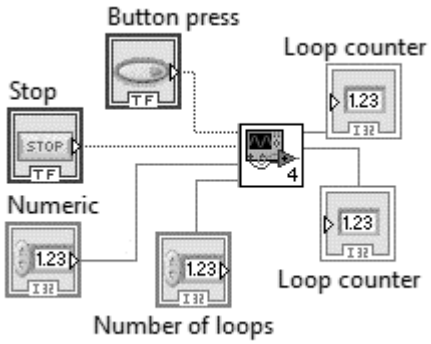


Figure 11: The routing in the block diagram

Returning to structures, other essentials are the "case" and the "event" blocks. The case block is equivalent to an if-statement. It can use both Boolean and numeric input as shown in figure 12. Multiple blocks of any kind can be placed in the frames to create more complicated structures. The event block, on the other hand, is somewhat equivalent to interrupts. The LabVIEW framework looks for events in the background. Running code will be interrupted if such an event happens, and the code inside of the event block will be executed instead. Figure 13 illustrates the event frame.

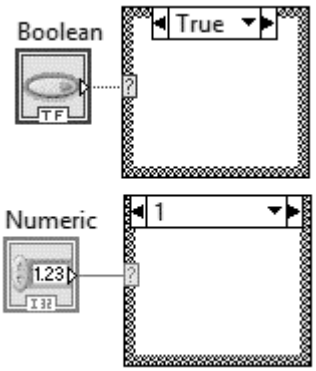


Figure 12: The "case" frames

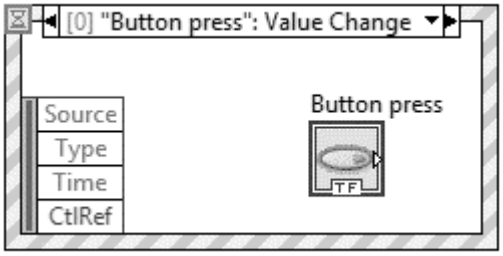


Figure 13: The "event" frame

Code typically runs from beginning to end in a sequential fashion. In LabVIEW, on the other hand, code can be executed in parallel. Therefore, the blocks (i.e. code) must sometimes be sequenced. A "flat sequence" is used to ensure that code executes the right order. Figure 14 illustrates the principle.

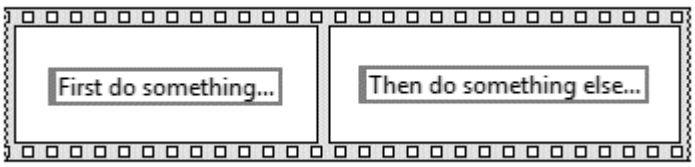


Figure 14: A flat sequence

A comprehensive suite of mathematical tools is provided too. The project uses the most basic of them, but one can easily apply FFT, Z-transform and other advanced math to the system. The “comparison” and “numerical” palettes, where functions like the described can be found, are shown in Appendix A.

Of course, the use of Boolean logic is fundamental to programming and computing. The "Boolean" palette in figure 17 gives access to such functions.

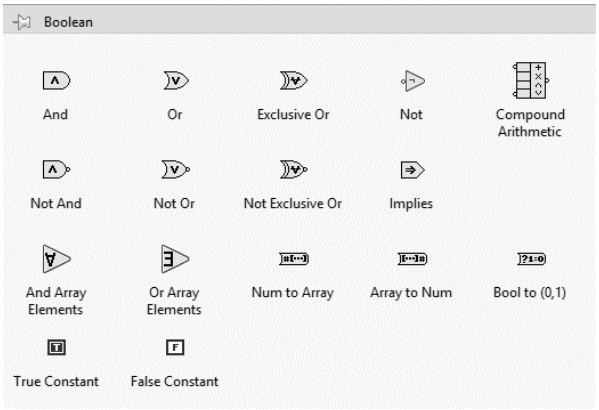


Figure 17: The Boolean palette

The functions for serial communication is also built-in to LabVIEW. It allows for the basic usage of COM-ports. The “serial” palette is used to do this and shown in figure 18 below.

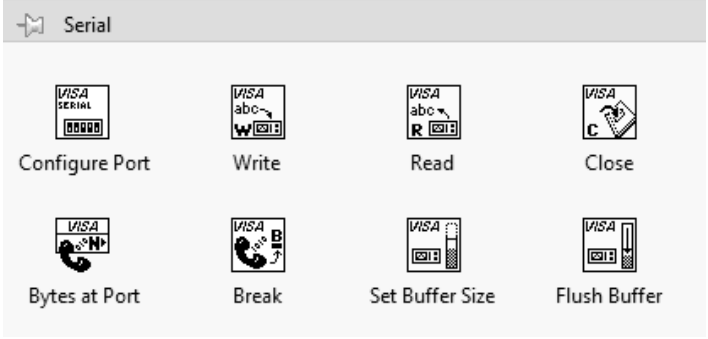


Figure 18: The serial palette

The end of this chapter concludes the description of the core functionalities in LabVIEW. In chapter 4.3 these will be combined, and some other features will show.

2.3.2 Teststand

Another product by NI is their test suite Teststand. Using this software in conjunction with LabVIEW makes it easy to create a sequenced testing scheme based on VI's. Teststand can automatically generate reports in HTML format, pictures and messages can be presented to the test technician as a request for manual interaction, and it provides a graphical interface for the whole functional test environment. More will be discussed about Teststand in chapter 4.6, "Automating the process".

2.3.3 BIST

In the current manual test, something called "Board in System Test", BIST, is used. It is a simple program that when uploaded to the MCU executes a check of internal components, such as memory. It also gives access to all the external functionality of the gateway. The interface for BIST is a serial terminal. A program called CoolTerm was used in the manual testing, but since it is not used in the project, it will not be discussed. Figure 19 on the next page shows the BIST initialization dump. The test rig will later utilize BIST and the gateways debug port to send and receive commands automatically.

2.3.4 J-COM

J-COM is a program developed by Jens Lorentzson, an employee at JEMAC, as a terminal for interaction with the gateway BIST. Programs like the above-mentioned CoolTerm, TeraTerm or the terminal included in Windows is usable, but these lack the ability to save frequently used commands. It was considered an annoying problem when debugging the gateway and was resolved by writing a simple C++ program. J-COM was handy to use in the project for testing out sequences manually and to explore BIST parameters on the gateway. Figure 20 on the next page shows J-COM's self-explaining interface.

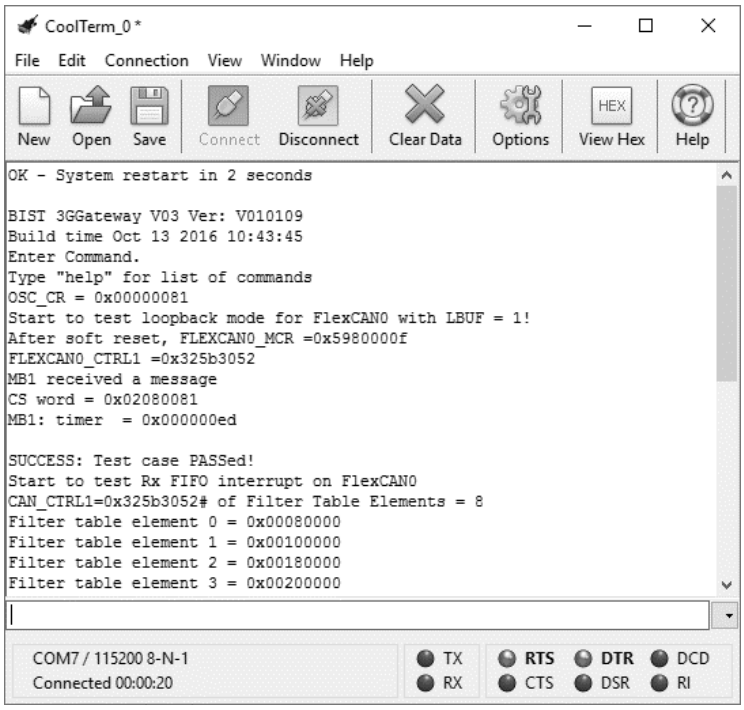


Figure 19: BIST initialization dump

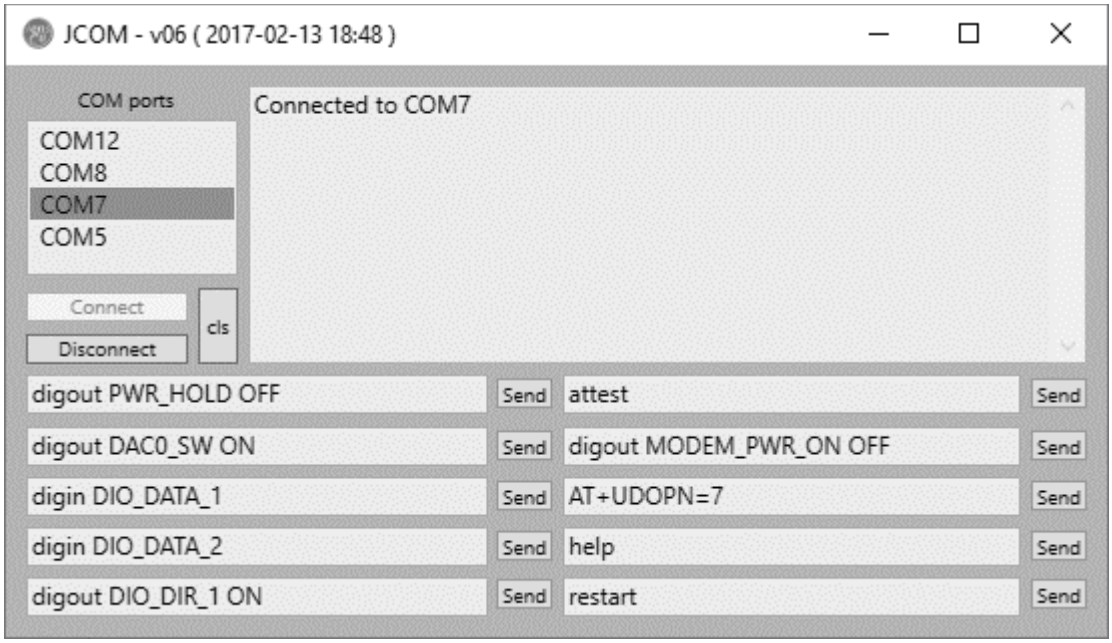


Figure 20: J-COM user interface

3 Method

The project plan uses a traditional waterfall approach. It was chosen over other models like Kanban and Scrum mainly due to that the project was to be completed by one person only. Designed to be used in teams, more complex models seem not to add more clarification of the project structure for a single individual, but make the workload heavier because of the time that must be spent on tracking progress and constant follow-ups. The defined goals lend themselves to a pretty straightforward approach; therefore, an iterative process was not considered feasible, and the idea of this was discarded.

Together with JEMAC, a Gantt-table was created to provide an initial timeframe for the completion of the project. Figure 21 shows the planned workflow.

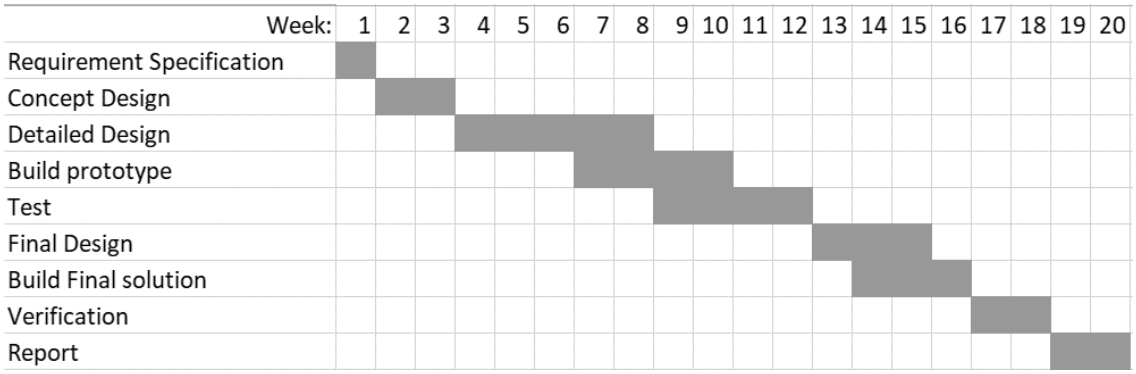


Figure 21: A preliminary timetable for the project.

Tasks were completed one by one as the project evolved. Some deviations from the original plan occurred due to unforeseen adjustments and other external factors such as waiting for parts. When delayed by events like that, the time was used to prepare for other tasks or to conduct research for, or write on, the report. The original timeframe was continuously revisited, to make sure that there was sufficient progress made despite the deviations.

3.1 Clients requirements

When the project began requirements of the system were amongst the first things to be discussed.

JEMAC wants to make a system which can be scaled and easily adapted to later models of the gateway. Initial ideas were to use an off-the-shelf microprocessor board, like the ATmega2560, as the central hardware unit. This board has plenty of I/O, DAC and ADC, and with the option to expand. It was thought to be supported by other non-costly devices as the hardware platform for the project.

Discussions concluded that the upside of this kind of boards is that they are very cost-effective when produced in the large quantities that they are, but they are not considered reliable enough in an application of professional character like the project is supposed to be. Components might be of inferior quality, and there is no traceability or manufacturer warranty provided. JEMAC distinctively opted for calibrated hardware that can be easily sourced and replaced in case of failure and because of these demands, the idea was discarded.

Along with the proposition of using the ATmega, there was also an idea of writing the test system in a language like Java or C++. The author had previous experiences from school and when working on private projects using these programming languages and thought that it might have been advantageous for the programming efficiency, not having to learn new coding methods. Considering this, JEMAC instead opted to use National Instruments LabVIEW as the software choice. Arguments for this were that is widely used in industry, has an excellent help forum online, and goes in line with JEMAC's desires of a scalable system.

The procedure of the current manual test uses serial communication with the gateway to send test commands and receive responses in a terminal on the computer. The automatic test system will use this model as a starting point. Naturally, discussions about the idea to use multiple serial interfaces followed, for trying to control all the hardware components to be used in the test rig. Almost full automation would be possible. The ability to control everything via a PC minimizes the need for human interaction with the test

rig; this is something that can prevent errors and speed up test times. LabVIEW has support for multiple serial terminals, so a solution like this is not prevented by software limitations. At last, the decision was made to use this configuration.

It would have been convenient to choose NI's customizable racks to unify the hardware and software requirements, but the price for these was not in the range of the project budget and they were a bit too advanced for the intended purposes. After searching for alternatives, the final choice of hardware was made; the chosen devices are discussed in chapter 3.4.

To summarize, when having had discussions with the client, these requirements have been made clear:

- A scalable and flexible system approach.
- LabVIEW as the system software.
- Preferably calibrated and manufacturer supported hardware.
- The use of devices that supports serial communication.
- Start off from current test software and BIST
- Budget is to be held reasonably small.

3.2 Test system requirements

The functionalities that are being advertised in the product specification were decided to be the basis for what is to be tested in the system. Discussions led to the soon followed specification of what is the criteria for each feature to pass the functional test. The features included are listed below, also technically described in chapter 2.2. Note that USB- and CAN-buses have not been included; this is due to that the current version of the BIST tests CAN while booting, and for that it does not handle the USB hardware yet. BIST is developed by another company, and efforts were made to get hold of a new version that included the desired new functions, but it proved unsuccessful. The addition of the 3.3 V and battery rail is made because of that that testing functionality was already included in the BIST, and might as well be included in the tests. Because of the failure to produce a new BIST, the test specification is limited to the gateway features listed below.

1. The following analog values must be within limits set by analysis of the production batch.

    - Battery rail
    - 3.3 V rail
    - Analog voltage inputs
    - Analog current inputs
    - Analog output
    - PTC inputs

2. Serial communication interfaces must be able to send and receive data at the fixed baud rate determined by BIST.

    - RS-232
    - RS-485

3. The modem needs to be able to recognize the SIM-card, connect to a predetermined carrier and report back signal strength.

4. The GNSS chip needs to be able to report NMEA strings on request.

5. For the Ethernet functionality to pass, the gateway needs to be recognized by the computer used when testing.

6. The gateway needs to be able to communicate with the accelerometer chip.

7. On request, the digital inputs should report back the corresponding logical state when the incoming signal is above or below the threshold.

8. The gateway must be able to produce a digital signal recognizable to external equipment.

9. The gateway must be able to go into sleep mode and be woken up using the following triggers:

- Reception of a digital high on port A.
- A button press from the user.
- The connection of external DC supply.
- On a signal from the RTC.

3.3 Deciding on hardware

With the decision about the choice of software made, a search for suitable hardware equipment followed. Sources were the catalogs of suppliers like Digi-Key, Farnell, Mouser and ELFA, as well as local distributors like Kjell&Co and Clas Ohlsson for generic devices.

3.3.1 Velleman PS3005D

To supply the rig with power, Velleman PS3005D was considered and later chosen. This power supply is shown in figure 22. JEMAC already has two of these units and is using them in development. They have been proven robust and trustworthy. PS3005D come with the desired feature of remote connection via a USB. The output voltage and current limit are controllable, and the actual current drawn from the supply can be read with three digits precision from the computer. Current limiting serves as protection for eventually short-circuited boards and other failures. Maximum power is specified at 30VDC and 5A. Voltage ripple is 2 mV, and current ripple is 3 mA. It meets the requirements of the test system. The unit was bought at Kjell&Co which offers this product for 1300 SEK at the time of this report. [17][18]

Some alternative options were also considered. LED power supplies is a cheap way of obtaining power. They come in various sized and, can be used to drive practically anything. For example, the price of a 720 W 24 V supply is 350 SEK on eBay. The price of a 24 W 24 V is 47 SEK [19].

These supplies have the disadvantage of being no-name, no warranty products, but are widely available and easy to replace though. They do not support serial communication. If this alternative was to be used, the supply should be controlled with a digital output pin via a relay and measurements of voltage and current would have to be made at a greater effort. The supplies meet none of the requirements but provide an example of the research made to provide product comparison.

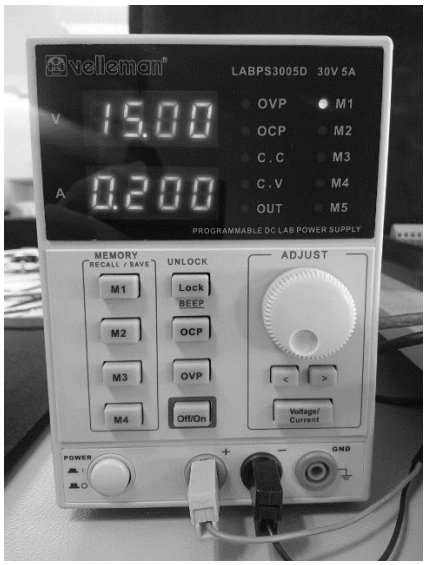


Figure 22: Velleman PS3005D.

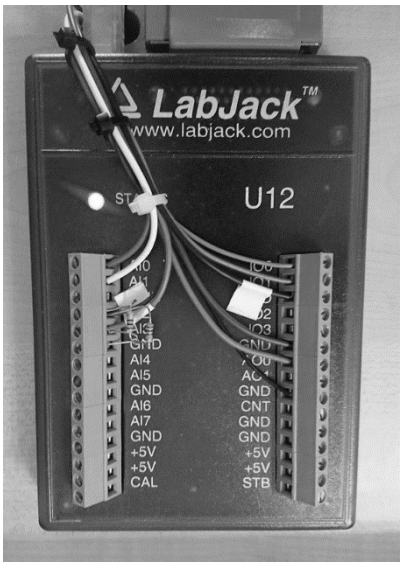


Figure 23: LabJack U12.

3.3.2 LabJack U12

An I/O system is needed to provide the gateway and relay switches with digital signals and analog voltages. The selected unit must also be able to measure the DAC voltage and current. To meet the requirements, it must feature a serial interface. ELFA turned out to have a good assortment of these kinds of measurement laboratory units. Manufacturers of this type of devices were shown to be Meilhaus, Velleman, and Advantech. Surely other manufacturers exist as well. A comparison of the available products proved that the different models of LabJack units were best suited. LabJack provides customers with a LabVIEW plugin for instant use in the software environment; this is a significant advantage over the others. Their prices are up to about 8000 SEK for the most competent versions. Simpler versions as the U12, appropriately sized for the project, is also available. [20]

Model U12 is the original product from LabJack, released in 2001. The company still supports the model and enjoys an excellent record of reliability. There are two analog outputs provided with a range of 0-5 V with 10-bit resolution. 20 digital I/O's is available to control switches and read signals. The analog inputs can be configured as eight ground-referenced or four differential, with ±10 V range and the resolution of 12-bit. [21]

To meet the requirement for the test system to be expandable, an excess of all types of I/O’s than what is needed for the project is desirable. The model U12 met this demand and the price criteria and was chosen for usage. Bought directly from the manufacturer the price is €130. However, the unit was purchased from ELFA.

3.3.3 FTDI USB-to-RS232 and USB-to-RS485 adapters

In the requirements, it was specified to use serial communication for all the devices. Nowadays, only old computers have a serial port. USB has become the new standard, but there is a lot of USB-to-serial adapters available. One prominent manufacturer of these is FTDI. Not much research had to be done in this case as it stood clear that most of the generic converters available on the market use FTDI chips. Two adapters were purchased directly from the company because the research made discovered cases where counterfeit FTDI chips were found. One converter was bought for RS-232 and one for RS-485. Information about the differences of these standards can be found in chapter 2.2. [22][23]

3.3.4 Plexgear USB-hub

The computer intended to run the test software does not have enough USB ports for the number of devices to be used. A generic hub to expand the ports was found on Kjell&Co for about 250 SEK. With this device, a deviation from the requirement of a calibrated device was accepted. USB hubs were considered reliable, and they are also easy to replace. [24]

3.3.5 Luxorparts 8-channel relay module

Sometimes it is necessary to be able to automatically perform a switch between two different inputs from the same gateway output. For example, this happens when having the same terminal for voltage and current output. The voltage is measured by one of the LabJack ADC's, and the current is measured by a shunt across another ADC. The choice of the module to perform this fell on a Luxorparts 8-channel optoisolated relay module. It has plenty of room for expansion and can break relatively large currents if such test were to be implemented in future expansions. Unfortunately, it lacks a serial interface, but considering the LabJack's many digital I/O's, it was thought better to utilize these than to spend money on a more competent module. [25]

3.4 Planning the test sequences

In this chapter, block diagrams (in figures 24 to 33) will be presented to show how the different test sequences were planned. They are general in nature and designed with the purpose to get a feel for the structure of the various tests and what tasks need to be done to perform them.

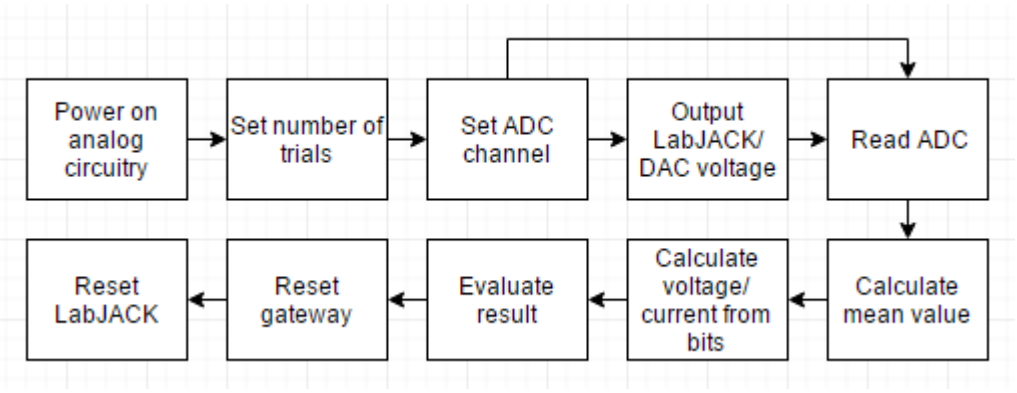


Figure 24: PTC, rails and ADC voltage/current sequence

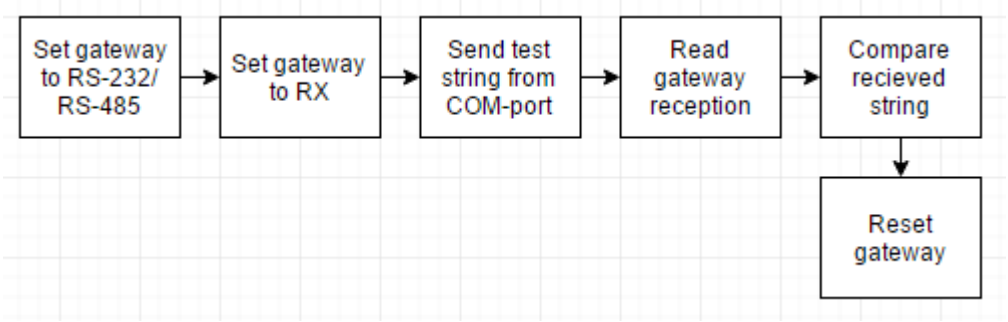


Figure 26: RS-232/RS-485 receive sequence

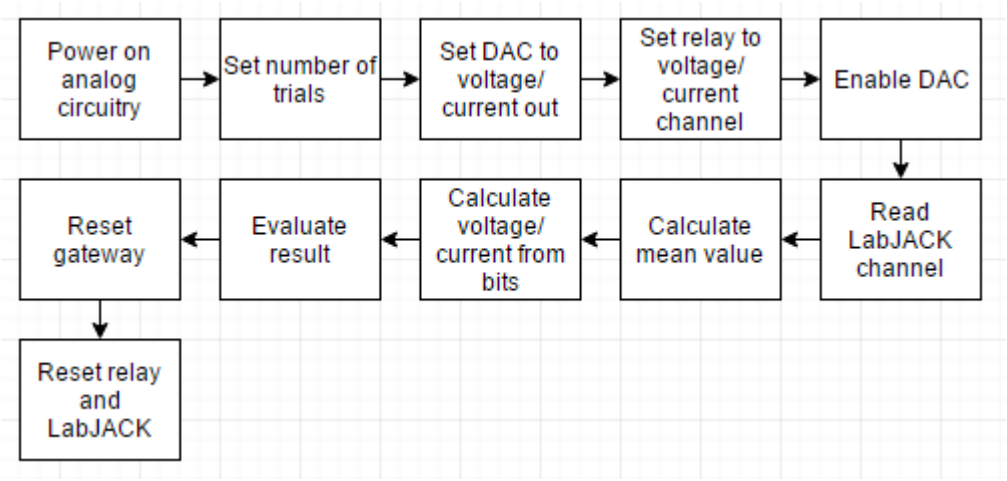


Figure 25: DAC voltage/current sequence

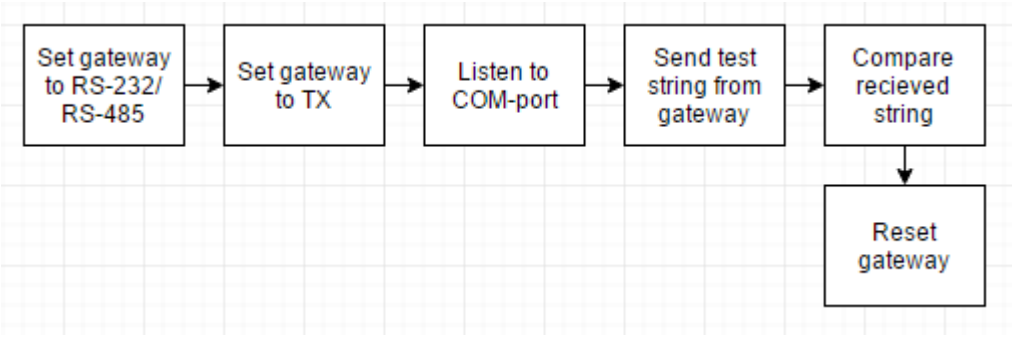


Figure 27: RS-232/RS-485 transmit sequence

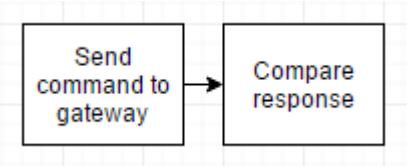


Figure 28: Accelerometer sequence

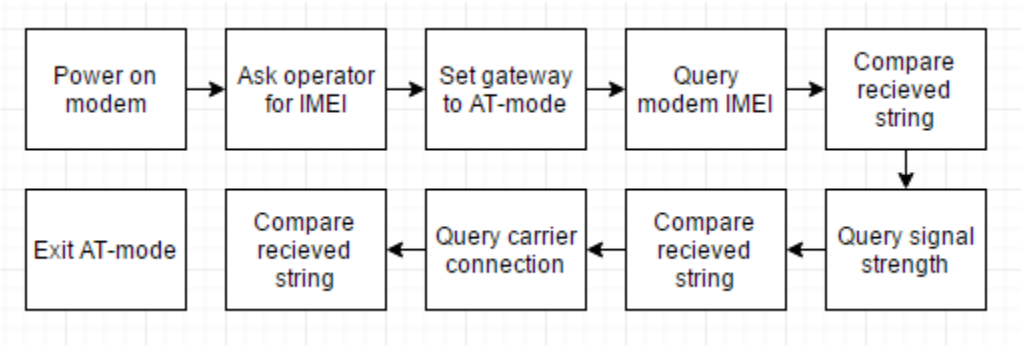


Figure 29: Modem sequence

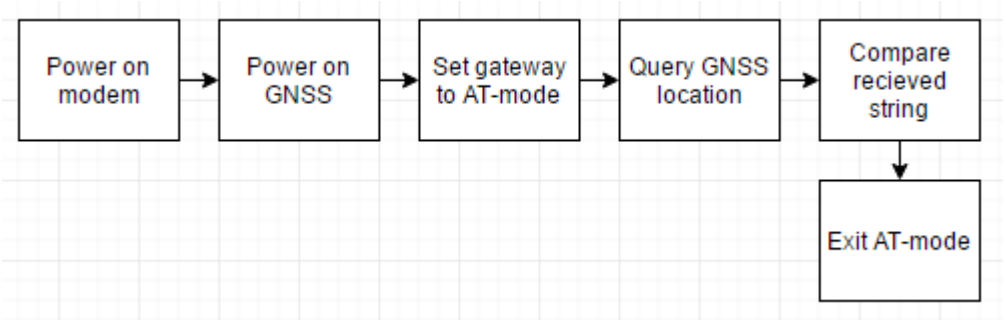


Figure 30: GNSS sequence

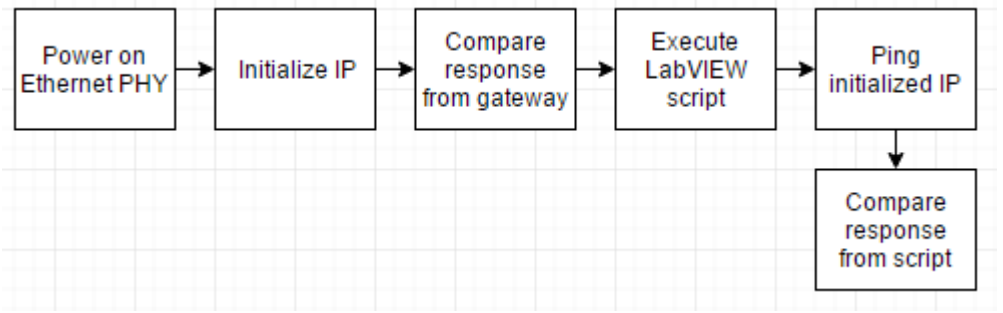


Figure 31: Ethernet sequence

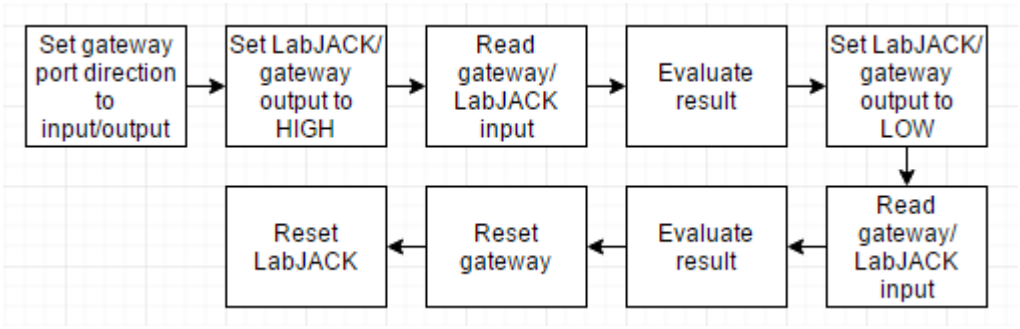


Figure 32: Digital input/output sequence

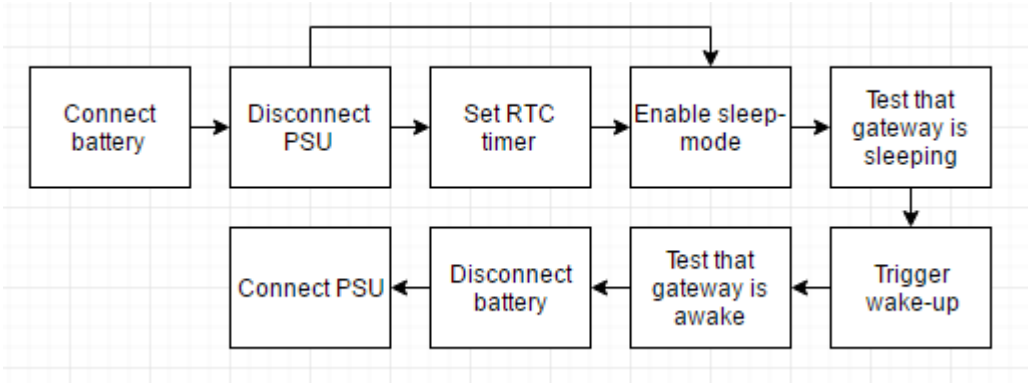


Figure 33: Wake-up sequence

4 Implementation

When the preparation phase was completed, the implementation could begin. This chapter will describe the process. Note that the sub-VI section of this chapter will not describe all the individual tests that were created, as these share many similarities, but will instead describe the key components of the program.

4.1 Setting up hardware

The gateway uses a terminal block for power and most of the external features. Appendix B shows the terminal block pin configuration and figure 1 in chapter 2.1 shows a picture of the female end where all the wires from the other test devices are being connected.

For the power and debug port, it was possible to re-use a cable from the manual tests. It already had the cables wired to one of the USB-to-serial adapters and banana jacks for the PSU. These are placed on the same plug. The connection between the USB-to-485 adapter and the corresponding terminal plug could also be made using a single cable to a single plug.

As can be seen in appendix B, the other functions share plugs. It was undesirable to connect cables from multiple devices across multiple plugs, so an adapter was manufactured using male terminal plugs and a D-SUB connector. A picture of the adapter is shown in figure 35. The cross-overs is made inside of the connector. It allowed for proper connections to the test devices from the D-SUB instead of from the gateway.

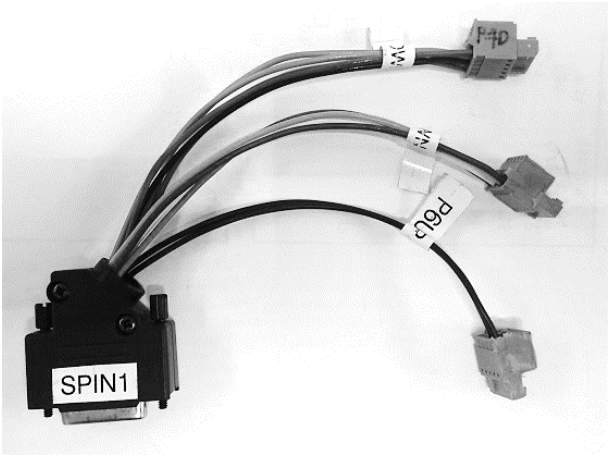


Figure 35: The terminal block-DSUB adapter

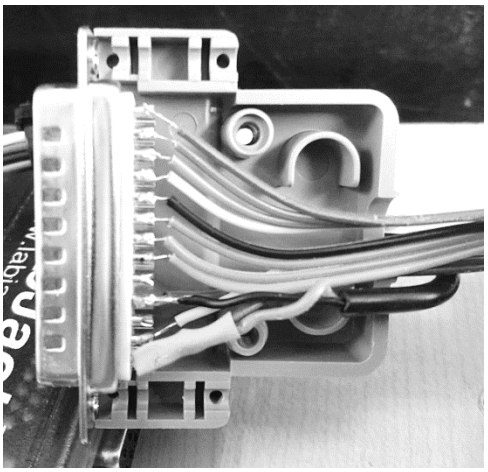


Figure 36: The pin header-DSUB adapter

Most of the LabJack's digital outputs are available through another D-SUB at the short end of the unit, and they were to be connected to the pin headers on the relay module. It was necessary to create an adapter also between these devices. A picture of the inside of that adapter can be seen in figure 36.

When testing the relays intended to switch between DAC output current/voltage and for connection of the battery when testing sleep-mode, it proved impossible for the LabJack to drive their coils. It was due to that when supplied by USB, as it was, the LabJack did not have enough headroom to provide the 90mA coil current and operate its own components at the same time. USB only allows for 100mA before the overcurrent protection circuitry sets in. A generic 5 V cell-phone charger able to provide 1A was used as a remedy, using this to supply the relay module with power. Far from an ideal solution, but it was being made to make progress. Besides that, the project was still in a prototype stage, and final optimizations were planned for later. [26]

All system components were connected to the USB-hub. Initially, the connections were being made as a "rat's nest", but at a certain point in the development, it was necessary to tidy up the wires a bit. That was done by the manufacture of a table for the devices to sit on. Holes were drilled to be able to hide the cables from view. When the adapters and table had been constructed, connections were successively made as test modules were programmed for each function. Figure 37 shows the complete table with the LabJack, relay module and battery on the top row, and the USB hub and

gateway on the bottom. Adapters and wires are hidden on the underside of the board.

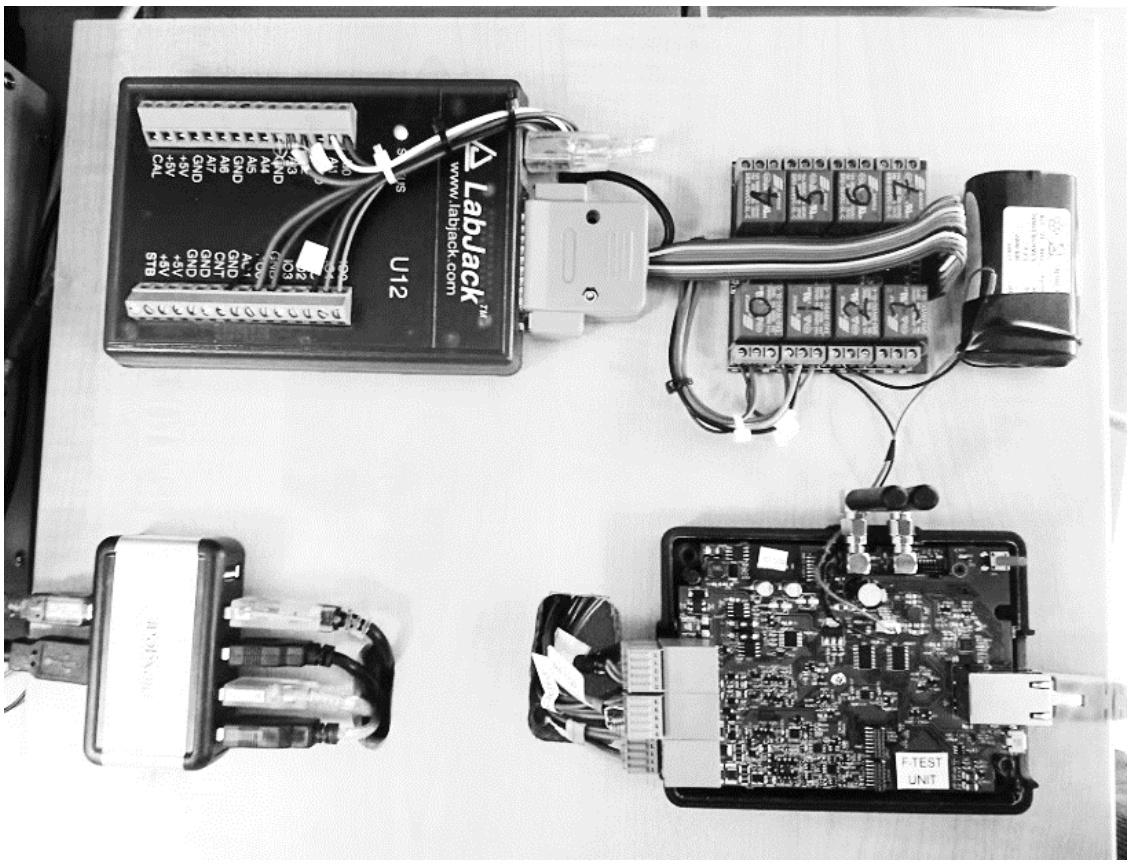


Figure 37: The hardware setup

4.2 Setting up communication

USB-to-serial is being used for the LabJack, the power supply, two RS-232 adapters and one RS-485 adapter. They count as five virtual COM-ports to the host computer. A subsection of LabVIEW called “Measurement and automation explorer” is where the setup of correct parameters for all the ports is handled. The ports are listed in a column and for each, settings of bitrate, data, stop bits and parity is made. It is possible to name the ports, for later convenience. All the devices that were being setup had the common 8-bit data, one stop bit, no parity settings, but the bitrate varied across them. The PSU and LabJack are set to 9600 baud while the gateway is set to 115200 baud. These transfer rates are the speed limits of the system.

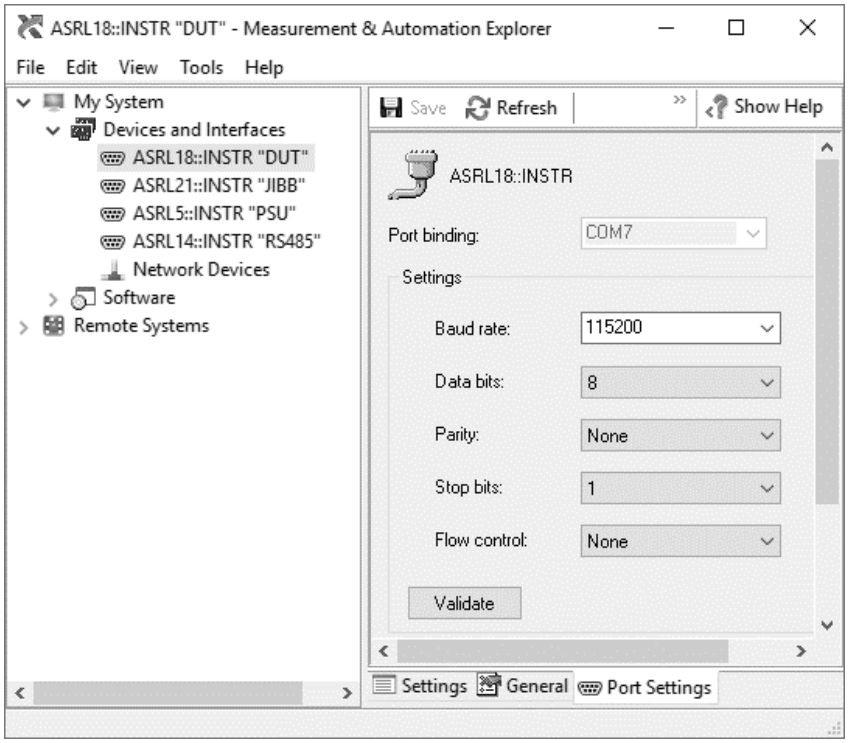


Figure 38: Measurement and automation explorer

4.3 Building Sub-VI's

LabVIEW and sub sequentially the whole project revolves around sub-VI's. While the main VI was built for the user interface, all the different tests were constructed as one sub-VI for each test. Not every one of the individual tests will be presented here. Instead, two different test types will be shown. At the core of all tests is the serial communication needed for both the gateway and the other system devices. This chapter begins with a presentation of how a VI for that was created. Figure 39 shows the block diagram, and it is also shown in Appendix C for a better view of the details.

4.3.1 Serial communication VI

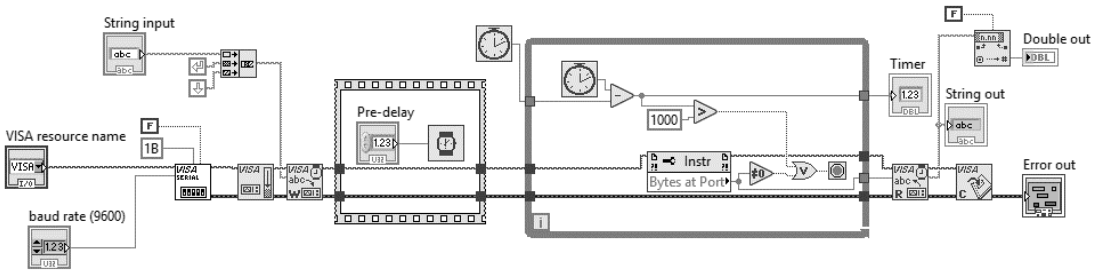


Figure 39: The serial communication sub-VI.

Beginning in the left area of the figure, there is a block called “VISA serial”. It is LabVIEW’s built-in functionality for serial data handling. The “VISA resource name” and “baud rate” blocks that proceeds, is configured as variable inputs that are made available (by routing) when calling the serial-VI from elsewhere.

The hexadecimal “1B” sets the port termination character, and the “F” is a false Boolean value set not to use parity control. Both are static because all devices are using these settings and that does not change. The wires that lead out from the VISA block are the “serial resource” (i.e. the port) that have been set in the block, and an error tracking function that is being passed along all blocks to spot where eventual errors in the execution occur. These two wires lead to a block without text, but with a downwards pointing arrow. It is a function to flush the serial input buffer before proceeding. The buffer might or might not be empty; use of the flush guarantees that no leftover data can corrupt future commands.

Wires are passed along to the “VISA write”-block and the execution continues. The variable string input that is to be written to the line follows. Between the input and the write-block, the incoming string is concatenated with a “line feed” and a “carriage return” symbol that is needed to terminate commands properly.

After the write-block is a “flat sequence” box with a “pre-delay” double input connected to a “wait”-function. It can slow down the process a bit for increased stability. The next step is the while-loop, and outside of its frame is a clock symbol. It will note the time just before entering the loop. The other clock inside runs for as long the while-loop is active, and when compared by subtraction to the first; if the loop runs for more than 1000 milliseconds, a Boolean “true” emerges which breaks the loop and continues with the execution of the VI. This is a time-out function, designed to prevent locking of the program if there were to be errors in the communication.

Another way to break the loop is via the function “Bytes at port”. If the loop is running, the function reads any incoming bytes into a buffer. These bytes are then passed to a “not equal to zero” function, used for detecting a state where there are no more bits to read. If there isn’t, the Boolean “or”-function that follows will be true, and the loop will break. The time it took to read

will be noted and presented in the “Timer” indicator, using the same time comparison as was used to detect a time-out.

At the same time as the buffer is being compared to zero, the data inside it is passed along to outside the while-loop, as can be seen in figure 39. After the break, data is then read from the buffer using the “VISA read” function. Digital data in the byte stream is converted to a string, and if that string is a number, the string is also converted to a double. Lastly, the “VISA close”-function closes the serial port and terminates the VI.

The constructed VI consists of a number of inputs and outputs. The inputs are the serial port, baud rate, and string input. Outputs are execution time, response string and response double. Figure 11 in chapter 2.3.1 shows an illustration of a created sub-VI’s block when used in another VI; the same principle applies with all VI’s. The block has all the inputs and outputs that were created in the programming sequence, and that was routed in the front panel.

Almost every test requires some parameter to be set on the gateway before requesting data. As an example of this, figure 40 shows a sequence where the power to the analog circuitry is turned off at the INT-board, and on at the MCU-board. The same serial VI as created before is used, but in this case, the gateway will simply not respond anything, and the read part of the serial VI will not see any data; the buffer will be zero. Although not seen in the figure, the sub-VI block has outputs on the opposite side of the shown inputs. They are not connected to anything and simply ignored.

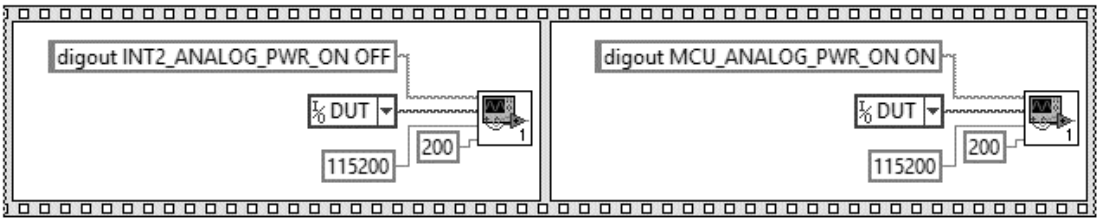


Figure 40: A typical command sequence

In some test cases, only the serial VI is required to determine a pass/fail. The gateway is given a command and responds, the answer is compared to what is to be expected, and if it matches, the test has passed. An example of such

a sequence is shown in figure 41. The accelerometer, modem, and GNSS are tested in this way, solely.

The simplest test is the one for the accelerometer. A command is sent requesting current X, Y, and Z-coordinates. Upon response, these are not evaluated but only checked for the proper syntax. That is because the uncertainty about the flatness of the surface holding the test rig. Figure 40 shows the simplicity of the sequence.

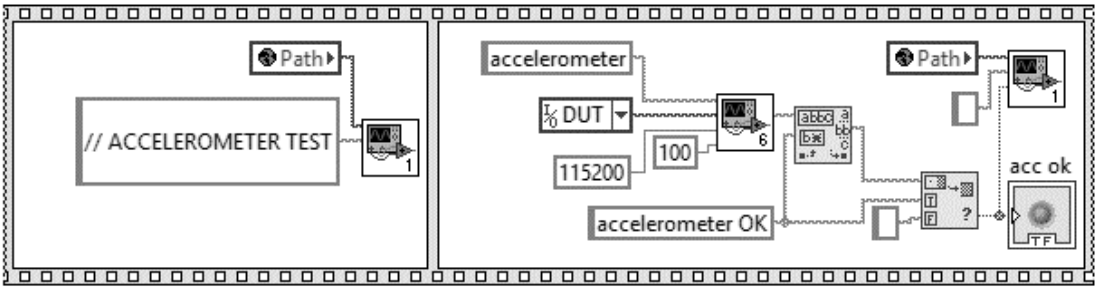


Figure 41: The simple accelerometer sequence

The modem has a whole lot of commands available to it. They are called AT-commands and is a standard set for modems. Before they can be executed, power to the modem must be turned on using a serial command. By yet another serial command, setting the gateway in “AT-mode”, AT-commands can be sent to the modem. The command creates a “bridge” between the debug port and the modem. AT-commands are then used to request data and set parameters. The modem test sends three requests; connection to carrier, signal strength, and IMEI readout. There are no pass limits for the signal; tested is only that the modem reports one. The strength varies due to lots of different uncontrollable circumstances, so to include limits would not work properly. IMEI is compared to what the user inputs at the beginning of the test and the carrier should match the SIM provider of the inserted card.

The GNSS chip is not directly connected to the MCU, but instead to the modem, and therefore GNSS is tested in the same way using AT-commands to request NMEA strings from the module. The “bridge” is used here also, this time extending to the GNSS. Only the syntax is being evaluated when the module responds because the test rig might be placed at another location then what it is expected to be, and the signal might not be totally accurate.

Ethernet is tested by sending a command to the gateway to initialize an IP at a certain address. This is to be chosen to match the computers network environment. To see if the IP exists on the network it is then tested with a LabVIEW script that uses the Windows command prompt and the “ping”-function that can be found there. The response of the script, with the result of the ping, is then evaluated in the sequence, and the IP is found, the test has passed.

The RS-232 and RS-485 ports are tested for the ability to send and receive data between the FTDI adapters and the gateway. The MCU’s UART is shared between the two protocols so a command to an analog switch must first be sent to switch between the two before sending any data on the ports. Both protocols are tested in the same way, only using different serial ports and adapters. First, the chosen port on the computer is opened by LabVIEW’s “VISA serial”-function; the same that was used in the construction of the serial-VI. The gateway is then commanded to send some test data to the computer, which if everything goes well, receives it, closes the port and allows the sequence to move on to test communication in the other way around. Doing so, the gateway is put in a “listen”-state waiting for incoming characters. Test data is then sent by the computer, and upon reception, the gateway leaves the “listen”-mode and reports the data back to the debug port. It can then be verified that the sent and received bytes of data matches.

Other tests require not only serial communication, but also the LabJack to read or write from its ports. A LabJack LabVIEW plugin is used for this. The control and display panel of the plugin can be seen in figure 42. It shows the state of the ports and allows the user to manipulate them. The LabJack inputs and outputs can also be automated by using blocks, wires, and inputs in the same way as any other LabVIEW object. Figure 43 shows the palette for LabJack where blocks are chosen from.

4.3.2 Analog measurement VI

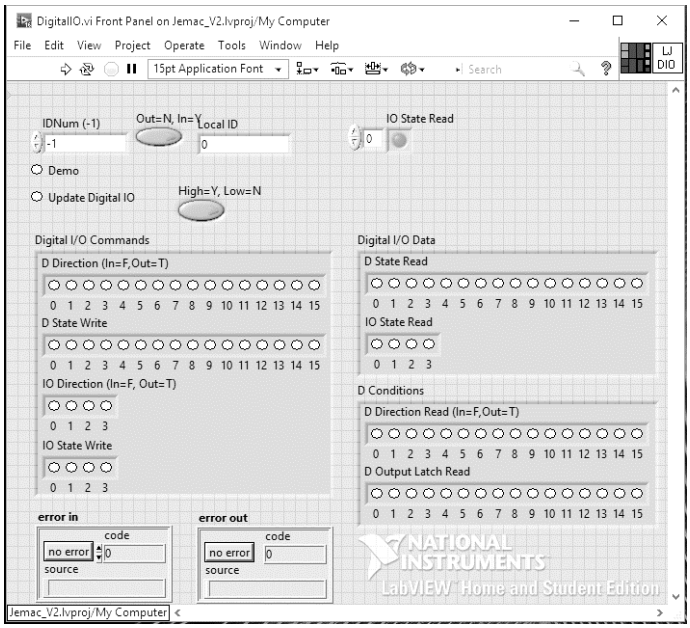


Figure 42: LabJack plugin control panel

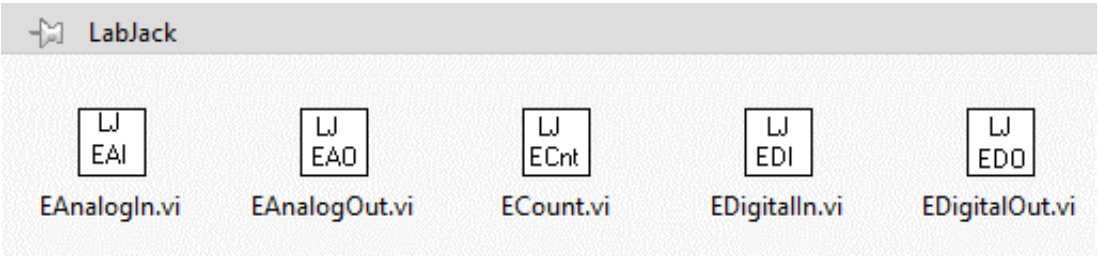


Figure 43: The LabJack plugin palette

Besides comparing the gateway output to expected values as done with all the above tests, another sub-VI was needed to evaluate the ADC/DAC accuracy. The front panel of a VI that accomplishes that is shown in figure 43. The “tick boxes” in the upper left corner is the same kind of inputs that were used in the serial-VI earlier. “CHX” and “_Y” is variable to be able to command any of the gateways four ADC’s. Actually, the sub-VI needs to be able to read six different inputs, but some of them share the same ADC and are switched by another command. There are four combinations to access the different channels: X0Y0, X0Y1, X1Y0, and X1Y1. The chosen values are concatenated with static values in the VI to form proper commands for the gateway. An example of such a command followes later, where “digin ADC-CH ” and “_ ” is the parts that are added to the integer user input.

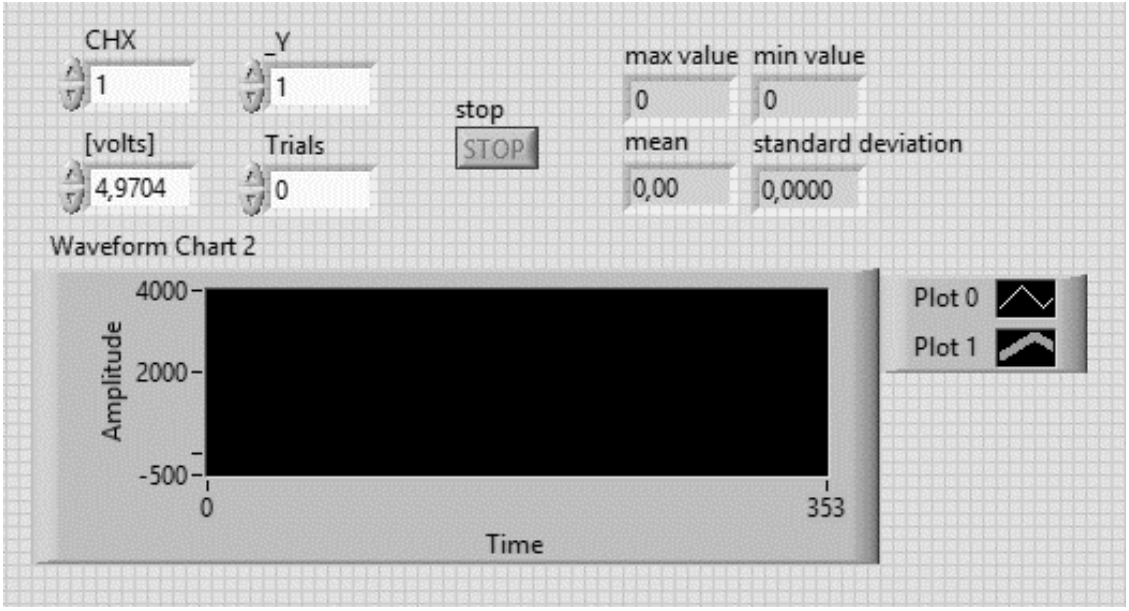


Figure 43: The front panel of the analog measurement VI

There can be fluctuations in the measurements, and for more stable results there is also an input to set the number of times that the ADC should be read. Unfortunately, due to the time it takes to execute each trial, the sampling rate is limited to only about 25Hz. Still, a hundred measurements do not take more than a few seconds.

Analog outputs from the LabJack's provide the ADC's with a known voltage that can be adjusted in the last input box on the upper left. The range is 0-5V, but there is a current limiting resistor in the LabJack's circuit that drops the voltage somewhat. This is discussed in chapter 5.

The converters are read by sending a command using the serial-VI created previously; for example, "digin ADC-CH1_1". By doing so, the requested ADC channel is being sampled, and the gateway will respond with the digital value. The serial VI will output this value to the measurement VI that is being created, and the resulting value is plotted on the graph. The graph is mostly there for manual testing purposes, such as when doing a large number of trials.

In the top right corner is the outputs of the VI. A "peak detector"-function keeps track of the maximum and minimum values. Another function computes the mean value and standard deviation of the set of data samples. That data can then be compared to threshold limits laid down for each test.

Data from the “measurement-VI” described above is used in the respective test sub-VIs for the two ADC voltage channels, the two ADC current channels, the two external temperature channels, and the two rails (battery and 3.3V).

A similar test is done with the DAC but in the reverse order. Digital values from the gateway provide a voltage at the LabJack input. It is being sampled there and evaluated in the same way.

All but one of the wake-up trigger tests (the one for a button press) are executed with the serial VI and the LabJack plugin in conjunction. A requirement for the sleep mode to work is that the battery is connected. To not having to do this manually, the battery connection is controlled via the relay module. The sequence starts with a digital output from the LabJack to the relay which closes the battery circuit to the gateway. The power supply is then turned off, but since on battery power now, the gateway is commanded to go to sleep.

To check that the gateway really is in sleep mode a dummy command to turn a LED on is sent to it. No response is expected; if a confirmation of the LED command is received, the gateway is awake, and the test halts and fails. Otherwise, the test continues by activating one of the wake-up triggers. In the first case, a high signal is sent to the gateways digital I/O. The unit is expected to wake-up, and after delaying the test to allow the MCU to boot up, another dummy command is sent to the gateway, also trying to activate a LED. This time the gateway should respond with a confirmation of the action for the test to pass. When this is determined, the last steps in the sequence turns the LED off, reconnects the power supply, disconnects the battery and resets the LabJack output.

The general idea is the same for the “power-on”, RTC and “button press” wake-up functions. In the “power on”-test, the only difference is that reconnection of the PSU is supposed to trigger the wake-up. The same dummy command method for determining if the gateway is asleep and has woken up correctly is used, as is the battery and PSU connection/disconnection, and reset of the LabJack and gateway.

For testing that the unit can wake itself up using the onboard RTC chip, one additional command must be sent to the gateway to set the duration of the sleep period. When putting the gateway to sleep, it is expected to wake-up after that time. The dummy method confirms that the gateway is sleeping, then the test sequence waits, allowing for the fixed period to pass and the gateway to boot. If everything is right, the second time the dummy test is performed, a confirmation is received that the unit has woken up and the test passes.

With the “button press”-test things are a little different. No automation of the button pressing is being made. The test operator must perform this action manually. Everything is the same as for the first two wake-up tests up to the point when the gateway is confirmed that it is sleeping. Instead of automatic triggering, a pop-up message is shown to the operator requesting a button press (on the gateway) and then for pressing an OK-button on-screen. After the OK-button is pressed, the test proceeds by waiting for the gateway to boot and continues with the confirmation that the gateway is awake, power disconnection and reset.

4.4 Building the interface

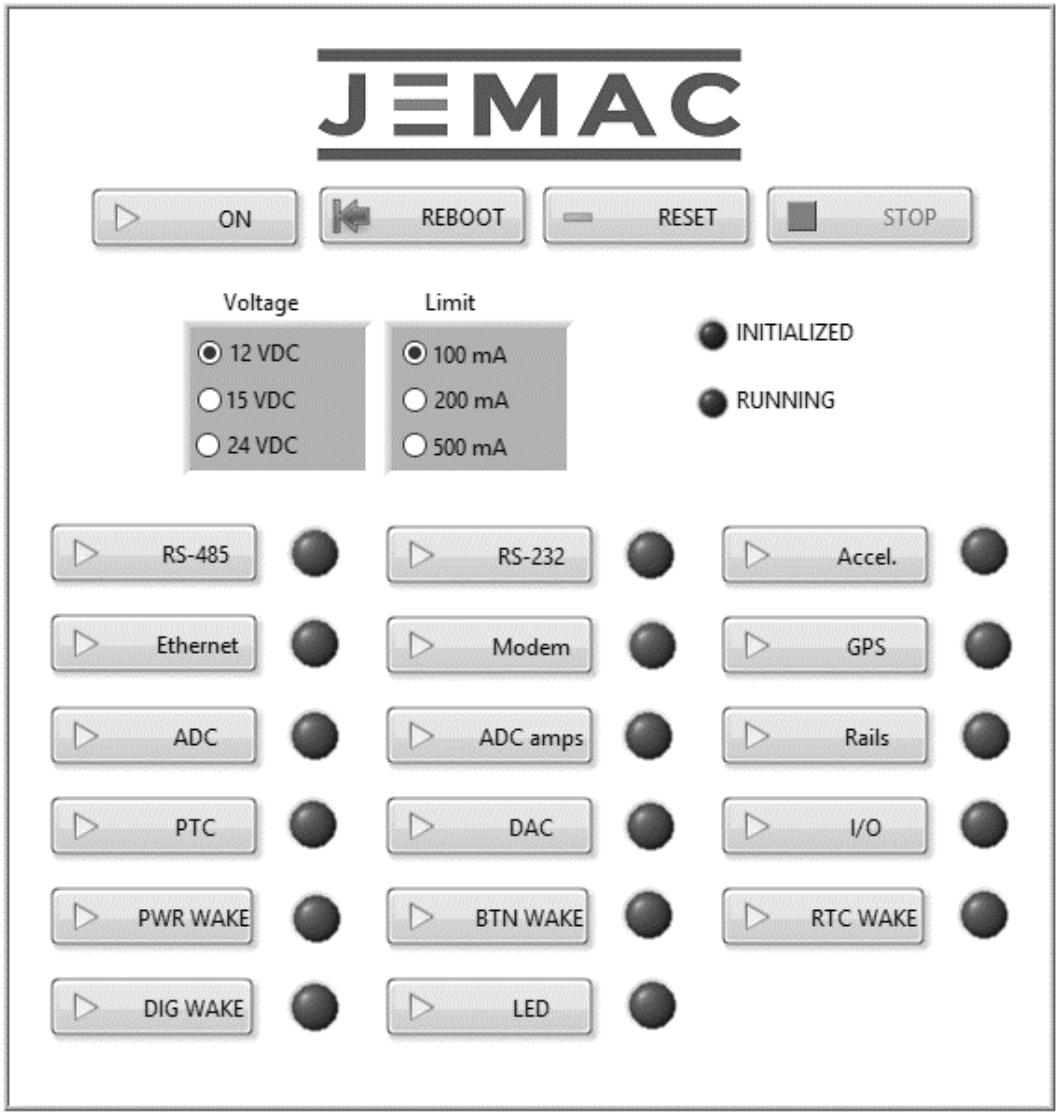


Figure 44: The finished interface

Figure 44 shows the finished interface. It is built around the LabVIEW event-blocks. All the other tests sub-VI's are contained within the main VI's block diagram. There is a while loop constantly running if the stop button is not pressed. That will halt execution and terminate the main VI. The buttons are Boolean controls, and they are associated with different events that then associate with the sub-VI's of all the different tests. Some buttons as the power control directly send commands to both the PSU and the gateway.

When running the main VI, the program will display a few user input requests. The first of the two reminds the user to plug in the right cables for

the terminal block and Ethernet and battery and to put in an SIM-card. An image of this can be seen in chapter 4.6.

The “voltage” and “limit” radio buttons set the voltage and current limit for the PSU. It is done in the same manner as in the case with the ADC’s in chapter 4.3; using concatenation of numbers and strings to form commands. The selectable voltage is in the range of what the gateway can handle, and the limits are set a little higher than what it consumes at most, for each voltage.

Pressing the “on” button turns the power supply on, and the button changes face and function to “off”. “Reboot” sends a command to the gateway to perform a soft reboot. The “initialized” indicator will turn off and turn on again when the gateway has responded that boot is complete. “Reset” resets all tests and turns off all the indicators for them. “Stop” of course stops the execution of the program. Then there are different buttons for each test, which when pressed will light up the “running” indicator and if passed, turn on the green indicator next to the button.

Using the main-VI and its interface allowes for a semi-automated test. It was created while building the different test as a way to make sure that the VI’s behaved properly. As the final product, the interface will not be used. All of the VI test execution will be automized by using Teststand. Chapter 4.6 presents and describes the fully automated finished product.

4.5 Setting pass limits

When testing gateways, a certain number of samples will be taken from each ADC/DAC. These samples will have a mean and a standard deviation of values that should both be within certain limits. To calculate what they should be set at, five gateways was sampled to find the average value and standard deviation for that population. Establishing the boundaries for an individual gateway to be $\pm 3\sigma$ from the populations mean and mean standard deviation, make the limits statistically represent practically all the gateways used to set them. Therefore, the limit reflects how the manufacturing process and design are working right now, with the current batch. A future batch might have values drifting from the current mean or with a larger spread, indicating that something has changed and needs to be addressed.

Setting the limits was done by the following procedure:

1. Sample five gateways 99 times each.
2. Calculate the mean value of each gateway as M_{Gx}
3. Calculate the standard deviation of each gateway as $\sigma_{Gx.}$
4. Calculate the mean value of the population as $M_{M,Gx}$
5. Calculate the standard deviation of the population mean values as $\sigma_{M,M,Gx}$
6. Set high and low mean pass thresholds to $M_{M,Gx} \pm 3 * \sigma_{M,M,Gx}$
7. Set high and low standard deviation thresholds to $\pm 3 * \sigma_{Gx}$

It can be challenging to follow the algorithm, so an example of the above, calculated for the 4mA DAC output, is given:

	M_{Gx}	$\sigma_{Gx.}$
G1	3,95	0,006
G2	3,93	0,004
G3	4,02	0,006
G4	3,97	0,009
G5	3,98	0,005

Table 1: Measurement data

$M_{M,Gx}$ is calculated as: $\frac{3,95+3,93+4,02+3,97+3,98}{5} = 3,97$

$\sigma_{M,M,Gx}$ is calculated as: $\sqrt{\frac{1}{5}\sum_{x}^{5}(M_{Gx} - 3,97)^2} = 0,032$

$M_{\sigma,Gx}$ is calculated as: $\frac{6+4+6+9+5}{5*1000} = 0,006$

The mean DAC current pass limits are set to:

$$M_{M,Gx} \pm 3 * \sigma_{M,M,Gx} = 3,97 \pm 0,096 \text{ mA}$$

The standard deviation pass limits for each set of measurements is set to:

$$\pm M_{\sigma,Gx} * 3 = \pm 0,006 * 3 = \pm 0,02$$

4.6 Automating the process

To automate the whole testing process proved an easy thing with NI's Teststand. When all the test VI's were completed and confirmed working in the manual interface, they were ready to be imported into Teststand. Because NI is the company behind both software products, the integration is seamless. When importing a VI into Teststand, all its routed variables will show up in the settings pane. In the example of figure 45, available input parameters are "Voltage" and "Limit", which is being set to 15 and 0.2. "Voltage" and "Idle current" are the possible outputs, where the latter is the one evaluated by Teststand to determine if the idle current is between set limits. The first is simply ignored but could be used as a test input as well.

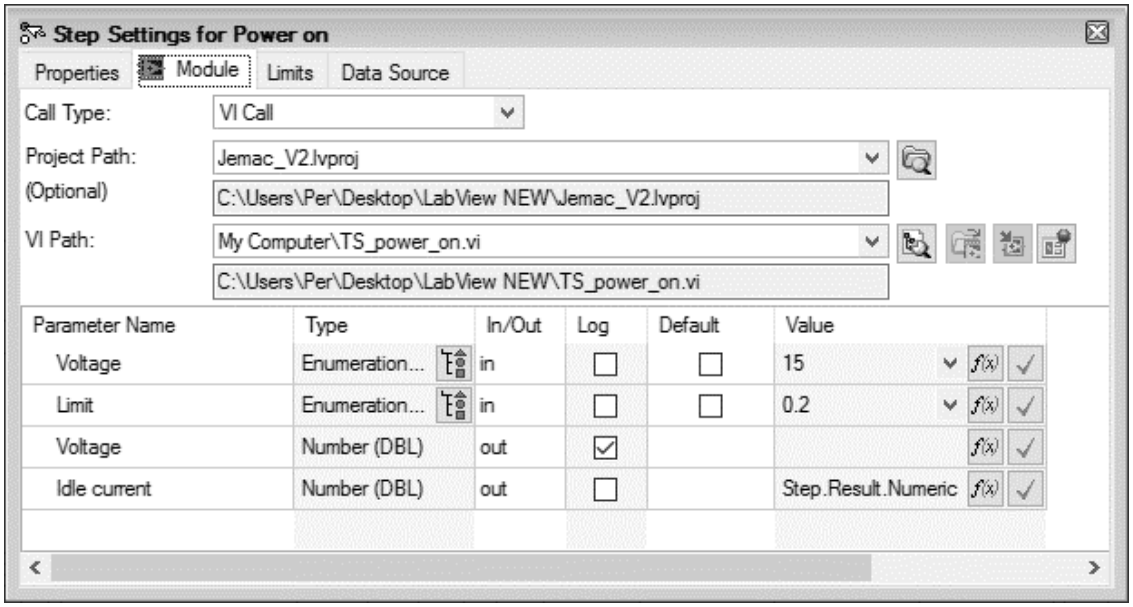


Figure 45: Teststand module settings

Importing the VI's one by one, soon the test sequence is completed. Mainly two types of tests were used in the projects test sequence; the "pass/fail" test and the "numerical limit" test. Most of the created VI's will run their command sequences to the gateway and evaluate the results from them, then return the result a Boolean true or false. The variable is then set up in Teststand to a "pass/fail" test. Analog tests use the "numerical limit" test of course. With each of them, the limits are configured in Teststand using the values calculated for each corresponding analog function.

Steps: MainSequence

Step	Description	Settings
Setup (4)		
Connection check	NameOf(Step)	Result Recording: Disabled
Power on	Numeric Limit Test, 30 <= x <= 40, milliampere, Jemac_V2.lv...	Additional Results
Initialize modem	Action, Jemac_V2.lvproj, TS_modem_pwr_on.vi	Result Recording: Disabled
Initialize analog circuitry	Action, Jemac_V2.lvproj, TS_analog_init.vi	Result Recording: Disabled
<End Group>		
Main (48)		
Transmitting RS232 data	Pass/Fail Test, Jemac_V2.lvproj, TS_RS232_send.vi	
Recieving RS232 data	Pass/Fail Test, Jemac_V2.lvproj, TS_RS232_recieve.vi	
Transmitting RS485 data	Pass/Fail Test, Jemac_V2.lvproj, TS_RS485_send.vi	
Recieving RS485 data	Pass/Fail Test, Jemac_V2.lvproj, TS_RS485_recieve.vi	
Initializing accelerometer	Pass/Fail Test, Jemac_V2.lvproj, Accelerometer_sequence.vi	
Initializing ethernet IP	Pass/Fail Test, Jemac_V2.lvproj, TS_ethernet_init.vi	Additional Results
Pinging gateway	Pass/Fail Test, Jemac_V2.lvproj, TS_ethernet_ping.vi	
IMEI	NameOf(Step)	Result Recording: Disabled
Checking IMEI	Pass/Fail Test, Jemac_V2.lvproj, TS_modem_imei.vi	Additional Results
Checking signal quality	Pass/Fail Test, Jemac_V2.lvproj, TS_modem_csq.vi	
Checking carrier status	Pass/Fail Test, Jemac_V2.lvproj, TS_modem_carrier.vi	Additional Results
Initializing GPS	Pass/Fail Test, Jemac_V2.lvproj, GPS_sequence.vi	
Measuring ADC CH1 0V	Numeric Limit Test, -1 <= x <= 1, volt, Jemac_V2.lvproj, TS	

Figure 46: The final test sequence in Teststand

Besides showing a list of imported VI's and the test types described in the previous paragraph, figure 46 also shows some steps for user input in the sequence. Teststand has a feature where the program can halt and display messages and pictures to the operator. Not only that, but it can also take input such as the modem's IMEI number that is being requested a bit into the sequence.

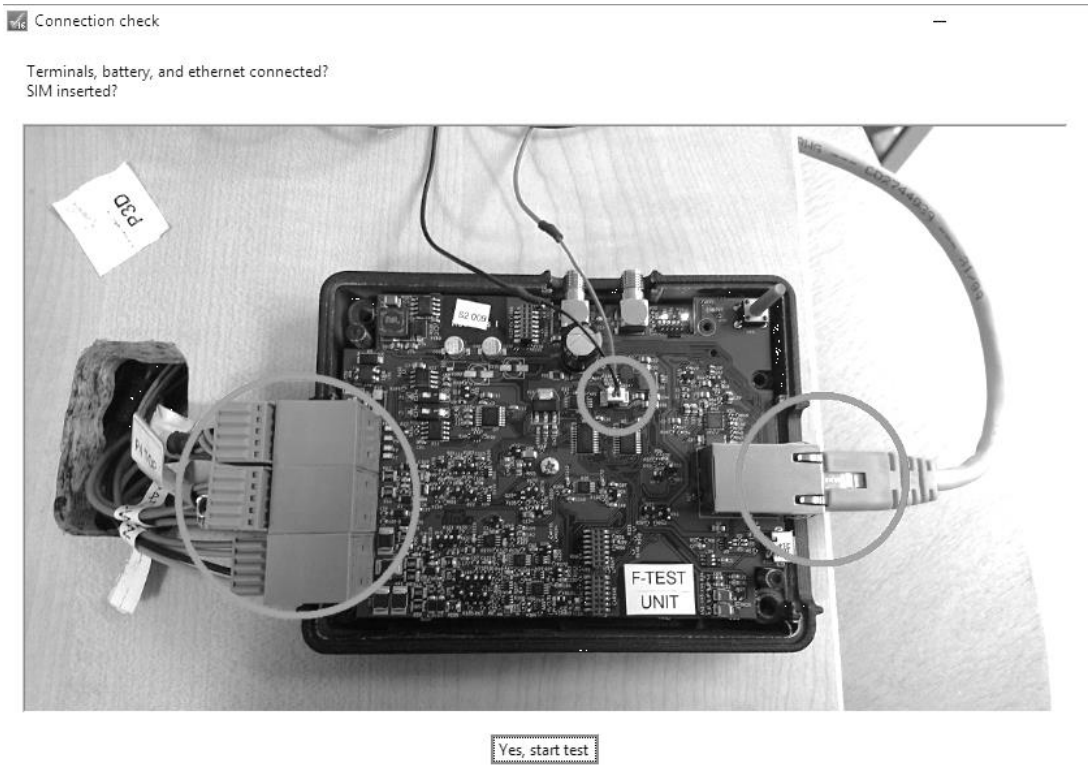


Figure 47: Message shown to user at startup

The first step in the list of sequences displays a picture; this is shown in figure 47. A photo has been taken and edited to clearly instruct the operator of what to do and where to do it. Pedagogy and clarity are keys when it is not known who will perform the tests. Many things can be misunderstood or neglected, and this needs to be avoided. After the confirmation, some steps are setting up the PSU and the gateway before the main test sequence starts. Tests are performed one by one and give a green or red indication of the result. The read and write delays to the gateway have been optimized for maximum speed so that the test should be quick.

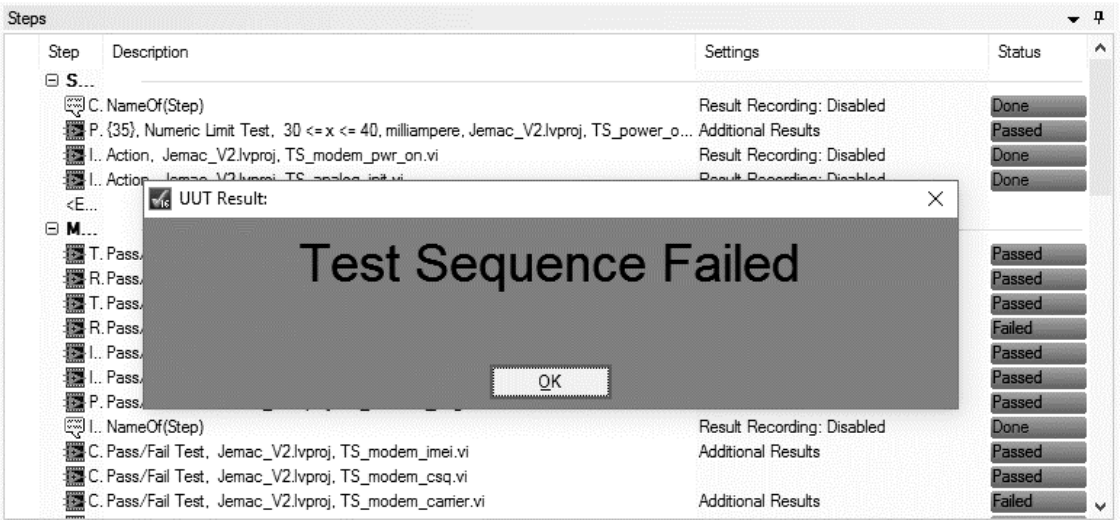


Figure 48: The sequence has halted, and the test has failed.

Figure 48 shows a complete test execution. This example has failed, and an explicit notification showing this is displayed to the operator. Following the completion of a test, the operator gets the choice of either continuing with testing of another unit or end testing and review the test reports. A serial number that has been asked for as the absolute first step; not programmed in the sequence but implied by Teststand itself, identifies the gateway. Reports save to a local directory as an HTML file with DUT serial and time/date as the filename. The reports can also be uploaded to a database, but JEMAC does not currently use one, so local storage was considered sufficient. The report that was generated from the test example above can be found in its whole in appendix D.

5 Verification

As a part of the project objective was beside building the test rig, also to verify that it works as intended. This was done in two ways; by measuring values externally on the devices, and by provoking errors by various means. Even though not explicitly stated in the problem specification, basic calibration of the analog measurements was also done to eliminate offset errors.

5.1 Provoking errors

To have something to compare against and verify that the tests worked, and not gave positive results no matter what circumstance, errors were provoked in various ways. One obvious thing was to simply not plugging in cables to the terminal block. Besides proving that the tests were OK, this method also caused confusion a lot of times when a test was expected to work. With the digital I/O's, wrong commands were being sent from the gateway and the LabJack to check for test correctness. The signals were also verified high and low, respectively; measuring with a multimeter to see that there were voltage and ground on the LabJack's terminals.

For modem verification, the wrong or no SIM-card was inserted, making all the tests fail. Commands to turn on internal circuitry were omitted as another way to verify tests. The battery was connected when it was not supposed to and disconnected when it was to, rendering the wake-up tests to fail. RS-232 and 485 had their COM-ports set to mismatching baud rates. The Ethernet cable was left unconnected, or the IP was initiated in the wrong way. All these kinds of provocations to the system was constantly made, and gradually increased the confidence of its stability.

5.2 Measurements

The analog parts of the system were tested by measurement. LabJack's terminals were a good place to probe for voltages to confirm that they were present when the system thought that they were, and the other way around. As stated before, calibration was simultaneously performed. Expected values were compared with real values, and factors of correction were calculated and inserted into the test VI's. When the DAC and ADC current input sequences were being programmed, the actual current was measured by a four-digit Fluke multimeter to calculate a relationship between current

and bits. However, only the most evident and easy to do calibrations were being done and not much time was put into that matter. The project was about functional testing and was to ensure that the unit’s work, not on how good they are working. Verification that the reality was what it was pictured as in the test system was of course needed to make the project trustworthy and the test rig solid.

6 Results

The project has produced a test system in LabVIEW for a client in need of improvements in functional testing. This chapter describes the results of the project from both the perspective of the goals specified in chapter 1.3 and from a general standpoint.

6.1 Goals of the project

A test specification was to be created, and this has been done and is found in chapter 3.2. It defines the pass criteria for each of the tested features. Since the project is about functional testing, this specification is not an extensive description of the underlying technical functionality that makes it possible for the features to function, but merely of a very basic level. After verification testing, it is assumed that the circuits should work.

The goal of creating a hardware and software system has also been met. The client requested the possibility to extend and continue to work on the test rig; this has been provided. The system runs on a well-used software and should be flexible enough to handle changes in hardware. Further development and possible improvements are discussed in chapter 7.

Verification of the test system has been done to the best of abilities. With a small population of gateways to test, limited time, limited budget, there is still some uncertainty left in the system. However, verification has concluded that the system is stable enough to be used in development, due to that the rig has been tested in conditions that would expect a fault, and correctly recorded these faults. Long-term stability is not assessed but is anticipated to be mainly a hardware issue; some improvements will be discussed.

6.2 Reliability

Doing the manual test, a technician needed to spend hours reading the cluttered and confusing bench test document. There was a huge possibility for the test not to be carried out in the same way, as they were being done by different persons. For a functional test, not being able to compare the products fairly, against a fixed reference, makes it impossible to guarantee each unit.

The reliability of the test system was gradually improved to the point that it finally was considered stable. Many “bugs” were found in the way, causing confusion at first but later proved to be of critical importance. As the test rig was developed, hardware engineers were doing verification tests simultaneously. They found some unexpected characteristics of the gateway that led to that the functional testing failed some of the tests. After having technicians correcting the problems, the tests passed. This was considered a good marker for the rig’s sturdiness.

Somewhere the middle of the project, the firmware developers ran into some issues when trying to initialize an IP using Ethernet. The problem caused the firmware to crash and the unit to lock up. Ethernet testing had been implemented in the test rig at this time, and this circumstance provided an opportunity to perform functional testing in a live case. While testing continuously and making changes to the gateway, the cause of the failure was narrowed down and finally found. The ability to repeat tests quickly contributed to the hardware engineer’s efforts to resolve the problem.

Also, the verification described in chapter 5 all suggests that the test system works as planned. In agreement with the client, the author draws the conclusion that the company’s goal of test standardization has been met.

6.3 Time-savings

No quantification of the time needed to perform manual testing was made, but it is estimated to be something close to one hour. The automated test, on the other hand, performs all the steps in approximately one minute, time which is documented by Teststand in every generated test report. The project has provided a significant improvement in this aspect. There is not much more to conclude about this; the improvement in time-savings make the stakeholders consider the process streamlined.

6.4 Ease of use

As stated, the manual tests were complicated and confusing to perform. The developed test software makes the process effortless. It is not complicated to use. All that is required from the user is to connect the cables, enter the gateway identification and IMEI-number, and press the button when requested. The test sequence runs indefinitely, processing gateways one by

one until the operator exits the test program. There needs to be no training or education for personnel to be able to perform the task of testing; instructions are provided by the program. The fact that the information is graphical, makes it understandable internationally. Since mass production usually takes place in Asia, this is a great feature. Not having to hire engineers or technicians for the task can make contributions to the company's economy, especially when hitting large volumes.

6.5 Risk

When delivering gateways for evaluation to potential customers, the company wants to be able to perform one final test when the gateway has been assembled, before packaging it and letting it go. Delivering a lot of units, testing could not be carried out due to the time it would take. Uncertainty and risk of the gateway to fail when being assessed by a potential customer was, of course, the result of not testing every unit; problems that have been eradicated with the development of the test rig.

6.6 Omitted features

When defining the test system requirements, the goal was to test all the features of the gateway. But not being able to acquire a new version of the BIST limited the possible tests that could be made; not all the features could be tested. CAN and USB were omitted. There were efforts made at the earliest stages, but the company that was supposed to develop and deliver the BIST did not succeed with doing so in time.

The implied improvement of replacing the 5 V relay supply, found in chapter 4.2, was also omitted. In a future hardware build, it is suggested to find a neater solution.

7 Conclusions and possible improvements

This chapter will first answer the questions in the problem specification found in chapter 1.4. The author concludes that the development of the rig has laid some groundwork for more extensive testing. The core software features can be used as the base for more detailed testing, manual or automatic. A comment on this is found in the forthcoming chapter 7.4. Some other remarks about the result of the project is commented in the other subsequent chapters.

1. How should the testing be performed?

   It was decided that the testing should be a continuation of existing practices. That is; testing should be performed with serial communication with the gateway, using pre-programmed firmware to send and receive commands to verify different functionality. Instead of recording and evaluating results manually, this was to be done automatically in the test system software.

2. What are the requirements for each parameter to be considered OK?

   The pass requirements for the analog tests have been addressed in chapter 4.5. Conclusively, they have not been set to an arbitrary value, but have been calculated to reflect the state of the manufacturing process as it is currently performed, and thus serving a double purpose; ensuring proper function of the units and at the same time benchmarking the manufacturing process.

3. What parts should constitute the test system?

   To be able to develop an automated version of the manual testing the system needs to consist of a PC with suitable software, a power supply, a relay module, a measurement device and USB adapters for the serial communication. The software has been described in chapter 2.3 and the hardware in chapter 3.3.

4. What are the requirements of the test system hardware?

   The client request that the test system hardware should be replaceable in case of failure, calibrated for confidence in measurements and reasonably inexpensive to fit the project budget.

5. How should information be shown in the user interface?

   The interface has been discussed in chapter 4.4 and 4.6. It was to be simple enough to be operated by a person with minimal or no technical knowledge. It was also to be easy to understand and require minimal human interaction.

6. How should settings and data be stored?

   Due to the fact that the company has no servers with the ability to host databases, it was considered good enough to save files to a shared directory on the test system PC. Another reason for this is also that the test system so far handles small batches of units and there is no real need for organization in a database.

7.1 Terminal blocks

Some issues remains regarding the terminal block connections. The wire gauge on the USB-to-serial adapters proved too thin at several times during development. It was addressed and fixed, but the long-term stability of the blocks is in question. An improvement would be to alter the wires that are connecting to the terminal block to a proper gauge. The male connector of the terminal itself could also be modified to be easier to connect to the block female. As is, some force is required to disconnect it, suggesting long-term fatigue of the operator. The male part of the headers could also be fixed to each other to eliminate the possibility for a wrong insertion.

7.2 Battery simulator

Another more advanced test is to check the battery charger; an inclusion of a battery simulator circuit would suit this purpose. The circuit should behave so that a control signal can be used to simulate the battery's voltage at empty and full (3.0 and 4.2 V). When set to 3 V, the circuit should handle the battery chargers charging current, which should be recorded and evaluated.

The simulated voltage should then change to 4.2 V, and at this voltage, the battery charging should be confirmed to have stopped. The PSU can be used to measure current or using a shunt and one of the LabJack's free analog inputs. Control signals are available in BIST to read charger status.

7.3 Casing and wiring

As can be seen in the pictures of the finished hardware; the layout is quite primitive. This was not considered a problem from the client's side, but there is room for improvement. For prolonged use of the test rig in unknown environments, the rig is vulnerable. The problem is thought to be in the casing and wiring underneath the table.

It is suggested that all the test devices are neatly and safely placed inside a shielded box for protection of external sources of influence. The wiring could be changed to shielded cables to reduce the risk of electromagnetic interference. The "rats nest" underneath the table should be sorted out by shortening the serial adapters and at the same time handling the gauge issue.

7.4 Usage of extra I/O

As suggested in the first paragraph of this chapter; the availability of extra I/O's on the LabJack and relay module lends itself to the development of more detailed testing, for example testing linearity and offset errors of the analog interfaces, or verifying the digital signal thresholds. It can also be used to connect/disconnect a USB power supply, test the gateways ability to drive the relays, and surely many other things conceivable if doing further development.

7.5 ESD protection

JEMAC has taken measures to minimize the risk of ESD damage in their premises. While working on the test system's hardware development, handling of the gateway was done on an ESD mat, and before entering the workspace, one would be encouraged to discharge by touching a metal piece, potentially being charged from walking around in the building. These measures provide protection while developing, but there could be ESD protection built into the system. A small ESD mat could be placed on the test table, and a bracelet could also be attached, with the user being encouraged to use it through a message at the beginning of the testing procedure.

8 References

[1] “JEMAC official webpage”, http://www.jemac.se, [2017-05-03]

[2] “LabView forums”, http://forums.ni.com/t5/LabVIEW/bd-p/170, [2017-05-03]

[3] “Application note 83: Fundamentals of RS-232 Serial Communications”, http://ecee.colorado.edu/~mcclurel/dan83.pdf, [2017-05-03]

[4] “Maxim Integrated MAX3232 datasheet” https://datasheets.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf, [2017-05-03]

[5] “Basics of the RS-485 standard”, http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/Basics-of-the-RS-485-Standard.aspx, [2017-05-03]

[6] “What’s The Difference Between The RS-232 And RS-485 Serial Interfaces?”, http://electronicdesign.com/what-s-difference-between/what-s-difference-between-rs-232-and-rs-485-serial-interfaces, [2017-05-03]

[7] “Maxim Integrated MAX1487 datasheet”, https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf, [2017-05-03]

[8] “Introduction to the Controller Area Network”, http://www.ti.com/lit/an/sloa101a/sloa101a.pdf, [2017-05-03]

[9] “Understanding and using the Controller Area Network “, http://inst.cs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf, [2017-05-03]

[10] “Introduction to USB On-The-Go”, http://www.usb.org/developers/onthego/USB_OTG_Intro.pdf, [2017-05-03]

[11] “NXP MMA8653FC datasheet”, http://cache.freescale.com/files/sensors/doc/data_sheet/MMA8653FC.pdf, [2017-05-03]

[12] “SARA-U2 series product summary”, https://www.u-blox.com/sites/default/files/SARA-U2_ProductSummary_%28UBX-13004142%29.pdf, [2017-05-03]

[13] “MAX-M8 series product summary”, https://www.u-blox.com/sites/default/files/products/documents/MAX-M8-FW3_ProductSummary_%28UBX-16008997%29.pdf, [2017-05-03]

[14] “SIGNAL CHAIN BASICS Series (Part 4): Introduction to analog/digital converter (ADC) types,”, http://www.eetimes.com/document.asp?doc_id=1272411, [2017-05-03]

[15] “NXP Kinetis K64 datasheet”, http://cache.freescale.com/files/microcontrollers/doc/data_sheet/K64P144M120SF5.pdf, [2017-05-03]

[16] “White Paper: Presicion Clock Synchronization”, https://www.belden.com/docs/upload/Precision_Clock_Synchronization_WP.pdf, [2017-05-03]

[17] “Velleman PS3005D datasheet”, http://www.velleman.eu/downloads/2/labps3005da5v06.pdf, [2017-05-03]

[18] “Kjell & Co power supply offerings”, https://www.kjell.com/se/sortiment/el-verktyg/stromforsorjning/labbaggregat, [2017-05-03]

[19] “eBay webpage”, http://www.ebay.com, [2017-05-03]

[20] “ELFA webpage”, https://www.elfa.se/sv/verktyg-maetteknik/maetinstrument/dataloggar-och-pc-maetinstrument/maetning-och-styrning-via-usb/maetning-och-styrning-via-usb/c/cat-31130, [2017-05-03]

[21] “LabJack product webpage”, https://labjack.com/products/u12, [2017-05-03]

[22] “FTDI Drivers and Counterfeit Chips”, https://www.sparkfun.com/news/1629, [2017-05-03]

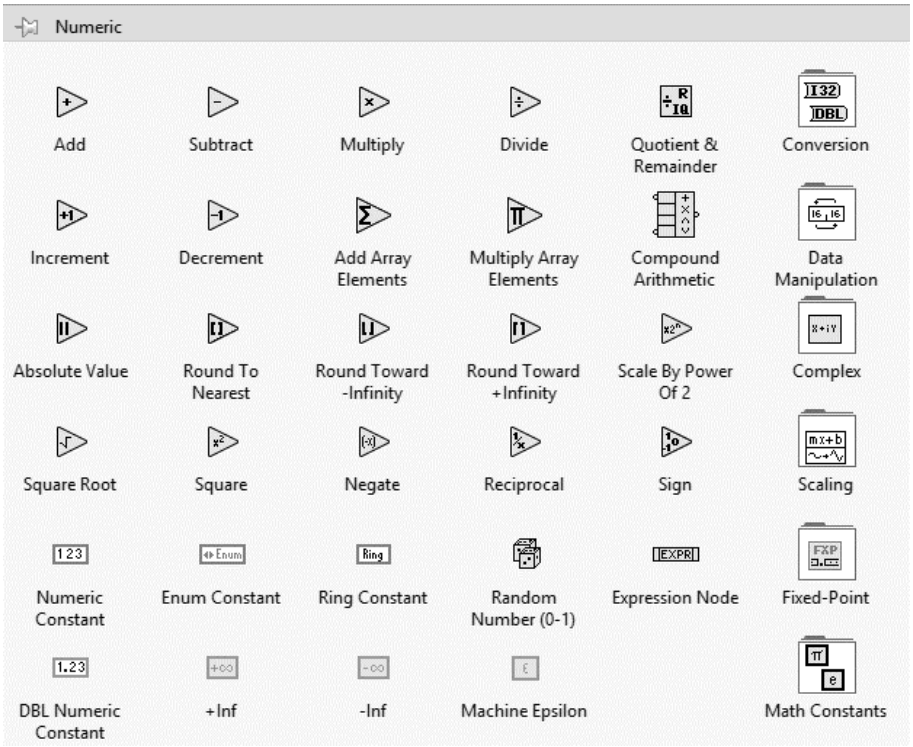
[23] “FTDI official webpage”, http://www.ftdichip.com/, [2017-05-03]

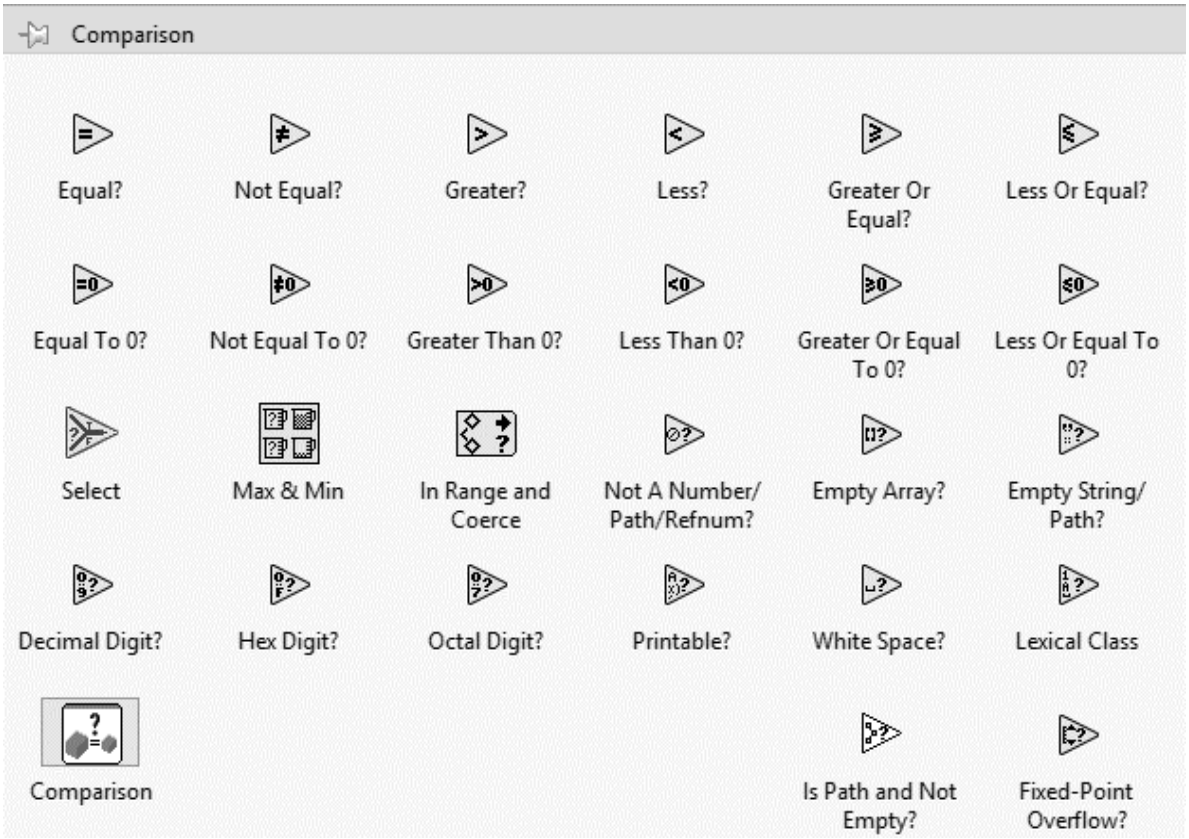
[24] “Kjell & Co USB-hub offerings”, https://www.kjell.com/se/sortiment/dator-natverk/datortillbehor/usb-tillbehor/usb-hubbar, [2017-05-03]

[25] “Kjell & Co module offerings”, https://www.kjell.com/se/sortiment/el-verktyg/elektronik/arduino/moduler, [2017-05-03]

[26] “Songle SRD05VDCSL datasheet”, http://www.datasheet4u.com/datasheet-pdf/Songle/SRD-05VDC-SL-C/pdf.php?id=720556, [2017-05-03]

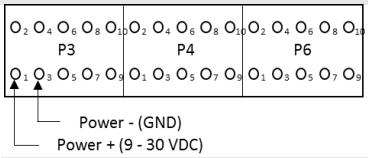
Appendix A


Numeric
Add
Subtract
Multiply
Divide
Quotient & Remainder
Conversion
Increment
Decrement
Add Array Elements
Multiply Array Elements
Compound Arithmetic
Data Manipulation
Absolute Value
Round To Nearest
Round Toward -Infinity
Round Toward +Infinity
Scale By Power Of 2
Complex
Square Root
Square
Negate
Reciprocal
Sign
Scaling
Numeric Constant
Enum Constant
Ring Constant
Random Number (0-1)
Expression Node
Fixed-Point
DBL Numeric Constant
+Inf
-Inf
Machine Epsilon
Math Constants

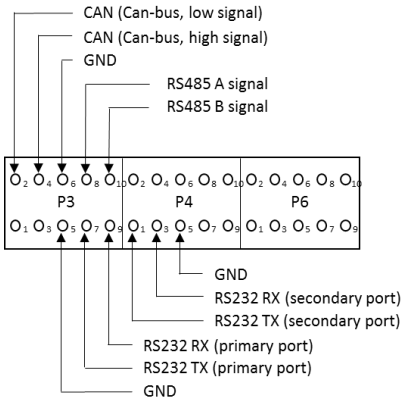

Comparison
Equal?
Not Equal?
Greater?
Less?
Greater Or Equal?
Less Or Equal?
Equal To 0?
Not Equal To 0?
Greater Than 0?
Less Than 0?
Greater Or Equal To 0?
Less Or Equal To 0?
Select
Max & Min
In Range and Coerce
Not A Number/ Path/Refnum?
Empty Array?
Empty String/ Path?
Decimal Digit?
Hex Digit?
Octal Digit?
Printable?
White Space?
Lexical Class
Comparison
Is Path and Not Empty?
Fixed-Point Overflow?

Appendix B

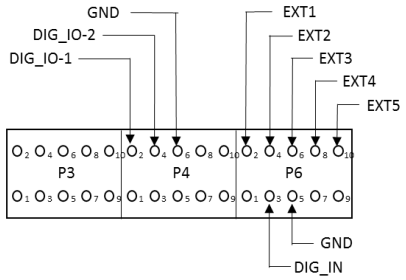
1. Power



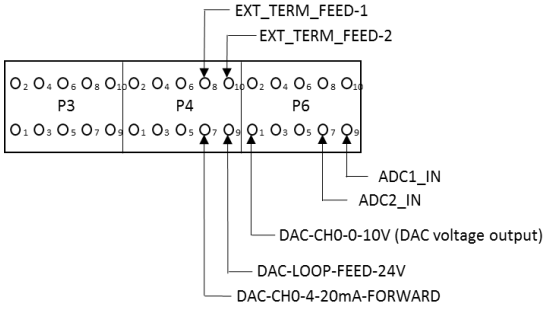
2. Can Bus and Serial Interfaces



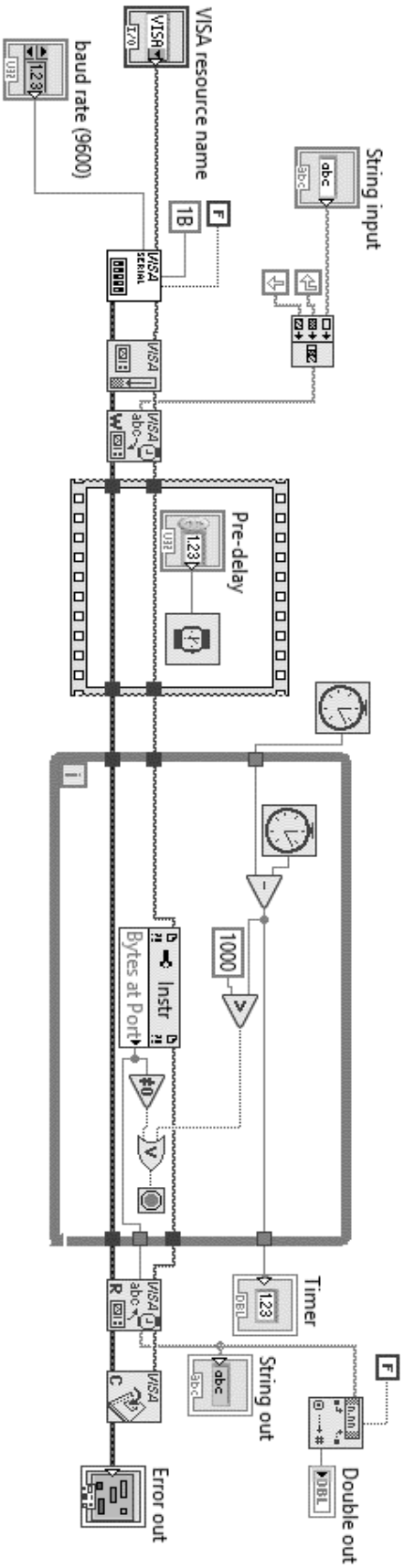
3. Digital IO and Extension Signals (connections to Add On Board on internal header J6)



4. Analogue IO



Appendix C


VISA resource name
baud rate (9600)
String input
Pre-delay
1000
Bytes at Port
Instr
Timer
String out
Double out
Error out

Appendix D

UUT Report

Station ID:	**JEMAC-172**
Serial Number:	**14853397**
Date:	**den 24 april 2017**
Time:	**14:18:23**
Operator:	**administrator**
Execution Time:	**17.1559061 seconds**
Number of Results:	**11**
UUT Result:	**Terminated**

Begin Sequence: MainSequence
(C:\Desktop\LabView NEW\Sequence File 1.seq)

Connection check	
Status:	Done
Button Index:	1
Power on	
Status:	Failed
Measurement:	35
Units:	milliampere
Limits:	
Low:	45
High:	70
Comparison Type:	GELE (>= <=)
Voltage [In]:	2
Module Time:	3.0417263
Transmitting RS232 data	
Status:	Passed
Module Time:	0.5751838
Recieving RS232 data	
Status:	Passed
Module Time:	0.4831744

Transmitting RS485 data	
Status:	Passed
Module Time:	0.599104
Recieving RS485 data	
Status:	Passed
Module Time:	0.6039543
Initializing accelerometer	
Status:	Passed
Module Time:	0.1867405
Initializing ethernet IP	
Status:	Passed
Module Time:	0.5855339
Pinging gateway	
Status:	Passed
Module Time:	4.9387234
Checking IMEI	
Status:	Passed
Module Time:	0.5834398
Checking signal quality	
Status:	Terminated
Module Time:	0.5585506

End Sequence: MainSequence
End UUT Report