Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Design of a battery electrolyte wetting measurement machine

Bram Vanderwegen

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

PROMOTOR :

Prof. dr. ir. Mohammadhosein SAFARI

BEGELEIDER :

dr. Saeed YARI

Gezamenlijke opleiding UHasselt en KU Leuven

UHASSELT KU LEUVEN

Universiteit Hasselt | Campus Diepenbeek | Faculteit Industriële Ingenieurswetenschappen | Agoralaan Gebouw H - Gebouw B | BE 3590 Diepenbeek

Universiteit Hasselt | Campus Diepenbeek | Agoralaan Gebouw D | BE 3590 Diepenbeek
Universiteit Hasselt | Campus Hasselt | Martelarenlaan 42 | BE 3500 Hasselt

UHASSELT KU LEUVEN

2024
2025

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Design of a battery electrolyte wetting measurement machine

Bram Vanderwegen

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

PROMOTOR :

Prof. dr. ir. Mohammadhosein SAFARI

BEGELEIDER :

dr. Saeed YARI

UHASSELT KU LEUVEN

Design of a battery electrolyte wetting measurement machine

Bram Vanderwegen

August 24, 2025

Acknowledgement

The completion of this thesis would not have been possible without the support and guidance of several individuals, to whom I wish to express my deepest gratitude.

First and foremost, I extend my sincere thanks to my supervisor, Prof. Dr. Momo Safari, for introducing me to the fascinating world of electrochemistry and for teaching me the fundamental principles of batteries that formed the essential groundwork for this research.

My profound gratitude goes to Dr. Saeed Yari, whose unwavering guidance and patience were invaluable throughout this entire process. Thank you for consistently making time to answer my countless questions, for providing crucial direction, and for your steadfast support from inception to completion.

I am also deeply grateful to the experts who provided their specialized knowledge. My sincere thanks to Frank Labiau of Brusaert, and to Simon Standop and Prashanth Chinta from Waygate Technologies, for their invaluable time and expertise in explaining the intricacies of ultrasonic transducers. Their insights were pivotal to the experimental component of this work.

On a personal note, I owe an immeasurable debt of gratitude to my family. To my parents, Hilde Vanderwegen and Jos Gennar, and my brother, Michiel Vanderwegen. Thank you for your unconditional love, unwavering belief in me, and for providing a constant source of encouragement and support throughout my studies.

Lastly, a special mention must go to my cat, Cem. His relentless companionship, calming presence, and entertaining antics were a continuous source of motivation and joy during the long hours of writing and research.

To all, thank you. This achievement is as much yours as it is mine.

Contents

List of Figures

Glossary

LUT	Lookup table
Package (Linux)	A software component that can be installed, updated, or removed as a single unit (e.g., a web browser or a compiler toolchain).
FPGA	Field-Programmable Gate Array
SoC	System on a Chip
VHDL	VHSIC Hardware Description Language
ADC	analogue-to-Digital Converter
DAC	Digital-to-analogue Converter
CNC	Computer Numerical Control
TOF	Time of Flight

Abstract

Batteries are widely used in applications ranging from portable electronics to electric vehicles, driving research into performance improvements. In batteries using liquid electrolyte, a critical design parameter affecting performance is electrolyte distribution, or battery wetting. However, non-destructive assessment of wetting remains challenging.

This thesis describes the design and construction of a low-cost (€5,000) ultrasonic scanning tool, specially suited for pouch-type batteries, for mapping battery wetting using 2 MHz ultrasound. The prototype utilises two ultrasonic transducers driven by a custom PCB featuring amplifiers and an Analogue to Digital Converter (ADC) for signal conversion. Digital control and processing employ a multi-level architecture: A Field Programmable Gate Array (FPGA) handles high-speed operations and parallel processing, a microcontroller manages low-level device control, and a Raspberry Pi 5 serves as the central controller and user interface. Transducers are submerged in a water or silicone oil bath for acoustic coupling and scanned linearly using a gantry repurposed from a 3D printer.

While the current system performs linear scans and that way differentiates material types, it does not perform an accurate full-area battery mapping due to unoptimised signal processing. However, with further optimisation and iterative development, the system offers prospects for cost-effective analysis of battery wetting.

Chapter 1

Introduction

1.1 Situation

In the past decades, battery technology has advanced substantially. With the commercialisation of the lithium-ion battery in the early 90s, new applications were made possible. This revolutionised portable electronics as this chemistry is able to store substantially more energy in a smaller, lighter package than its predecessor, giving rise to laptops and smartphones. A significant challenge of the 21st century is the energy transition from carbon-based energy sources towards more sustainable sources such as solar panels and wind turbines. The unreliable nature of these energy sources invokes the need for energy storage solutions. Lithium ion technology proves to be an effective way to store these amounts of energy. Another facet of the decarbonisation is the electrification of transport. Electric Vehicles (EVs) provide a renewable method of transportation and also run off lithium ion technology. Future applications could be a revolution of the aerial transport sector where projected higher capacity batteries could replace existing fossil ones.

The functional principle of lithium ion batteries is the transfer of lithium ion charge carriers between electrodes to store charge. Lithium Ion batteries follow the same construction principle of most batteries, with each cell consisting of two electrodes, an anode, cathode and electrolyte and a separator preventing electrical connection between electrodes. The electrodes are subsequently connected to charge collectors which connect the chemistry to an electrical interface. The anode is usually made of graphite and the cathode of different lithium oxides, defining their chemistry. These oxides can be Lithium Cobalt Oxide ($LiCoO_2$), Lithium Manganese Oxide ($LiMn_2O_4$) or Lithium Iron Phosphate ($LiFePO_4$). [2] The electrodes are porous so the lithium ion charge carriers can insert and de-insert themselves. The pores consist of carbon with active material attached to it. The lithium ions connect to the active material. This is then connected with a binder polymer to the electrode. The electrolyte is added to fill the pores and have the minimum required volume to make the battery work. The pores cause the electrolyte to have maximum surface area with the electrode.

The electrolyte is the medium through which the lithium migrates. Since a battery covers a surface, the electrolyte spread, or wetting, is different across the battery. These differences can influence battery performance as they locally change the resistance. This leads to lower capacity and locally higher temperatures, which lead to higher degradation and lower cyclability. There-

fore, knowledge of this distribution is useful in design as the effect of the amount of electrolyte can be monitored. Electrolyte spread also is an indicator for battery age as gas forms with cycling, displacing the electrolyte. [3]

Ultrasonic sound wave transmission responds to electrolyte density since the different materials have different acoustic impedances. This causes a transmitted wave to attenuate based on the material it passes. Furthermore, different materials have different transmission speeds. Which means differentiation can be done based on time of flight.

A comprehensive method of using ultrasonic sounds to scan a battery can offer insight in the wetting as Deng et al. demonstrated. [3] Therefore there is interest in research group for a device capable of performing this type of scan. This thesis describes a method of constructing such a machine.

Chapter 2

Literature study Ultrasonics

2.1 Introduction

Ultrasound technology has many different applications that range from measurement to imaging and treatment. In order to produce these ultrasounds, a device is needed, called a transducer. Fundamentally, an ultrasonic transducer is a component that converts one type of energy into ultrasonic acoustic waves. This initial type of energy can be electric, but also optical and mechanical. Electrical transducers are the most common because of the piezoelectric transducer which is easy to make and implement. Furthermore, electrical transducers are versatile, as electrical signals can be relatively easily manipulated, allowing for a wide range of applications. Transducers come in configurations for different purposes. There are different element configurations such as single element, dual element, and phased arrays. Furthermore, there are also different configurations for the ultrasonic carrier such as contact transducers to transfer the ultrasound to solid materials and immersion transducers to transfer the ultrasound to liquids. [4][1]

2.2 Ultrasound

Ultrasound is acoustic waves with a frequency greater than 20 kHz [1]. This frequency is the upper limit of human hearing and therefore is not audible. As ultrasound is an acoustic wave, it needs a medium to propagate through. This is useful compared to electromagnetic waves, while not needing a medium to propagate through, they are often blocked by certain materials, while ultrasound can pass through with ease. Ultrasound also experiences reflection when the wave propagates to another. Based on reflection/transmission behaviour, information can be obtained about the composition of the material.

2.3 Generator materials

There are different types of ultrasonic transducers. For electric drivers you have piezoelectric, electromagnetic and capacitative. Predominantly the first two. Furthermore there are optical and mechanical transducer, however these are not applicable to this project due to the complexity to make them work in this context.

2.3.1 Piezoelectric

Piezoelectric transducers use the piezoelectric effect to convert the electrical signal into ultrasound. The piezoelectric effect is a property that some materials posses that causes them to generate an electrical signal whenever pressure is applied. It does this by applying pressure to a polarised material. This causes the internal structure of the material to shift. In piezoelectric materials this shift causes a dipole to be formed. This dipole is measurable as a voltage linear to the pressure.

2.3.2 Electromagnetic

Electromagnetic transducers work based on inducing magnetic forces to bring forward a ultrasonic wave. There are three types of sub-mechanisms that fall under this category. First the Lorentz forces. These are forces that occur when a current flows under a magnetic field. The forces are proportional to the current. By driving this current with specific frequencies, the Lorentz force can generate an ultrasonic vibration bringing forth ultrasound. The second way is by bringing a ferromagnetic material in an electromagnetic field. Due to different factors such as coil design and the skin effect. Different levels of flux density exist which generate forces. The final one is magnetostriction. Ferromagnetic materials can change form under a magnetic field. Change in volume and length under a changing magnetic field can induce ultrasound proportional to the changing magnetic field.

2.3.3 Piezoelectric Ceramics

Piezoelectric ceramics are mostly in the form of perovskites. These are materials with a chemical structure of $\mathbf{AB}O_3$ with **A** and **B** being cations. The initial material was $BaTiO_3$. This material demonstrated its capabilities in generating a dipole. After new materials were discovered and researched, most applications now use lead zirconate titanate ($Pb(Zr_xTi_{1-x})O_3$, $0 \leq x \leq 1$) (PZT) as it has the highest performance for practical applications. [4]

2.3.4 Piezoelectric Polymers

Piezoelectric polymers, such as Polyvinylidene fluoride (PVDF), offer a flexible alternative to rigid ceramic transducers. While they exhibit a lower piezoelectric constant, making them less suitable for high-power applications, their material properties provide distinct advantages. Key benefits include mechanical flexibility, which enables conformal geometries; simplified processing via methods like extrusion and nano-structuring; and low acoustic impedance, which improves coupling with water or biological tissues. These characteristics also grant them superior biocompatibility, reducing the need for additional coatings. Consequently, piezoelectric polymers are primarily employed in specialized applications including biomedical devices, energy harvesters, and flexible sensors [5].

2.3.5 Piezoelectric Crystals

Crystals were the first type of piezoelectric material to be used, however it has been mostly superseded by ceramics. However for specific applications they can be desirable. Such as high temperature or for situations where chemical inertness is desirable.

2.3.6 Piezocomposites

In some cases there is a large acoustic mismatch between the transducer and the propagation medium. To solve this a piezocomposite can be used. A piezocomposite is a material made with ceramics or crystals with polymers sandwiched in a matrix shape. Allowing for customisation of acoustic impedance. However it also has disadvantages in cost, complexity and certain characteristics such as temperature resilience. Compromising the advantages of some of the constituent piezoelectric materials.

2.4 Types of transducers

While the fundamental principle of a transducer stays the same, there are different methods on how to transmit the ultrasound to the desired location. They can mainly be subdivided in two types: The contact transducer and the immersion transducer.[6]

2.4.1 Contact transducer

Contact transducers make direct contact with the material to transmit the waves. This requires the material to scan to be solid materials with simple geometries. They are coupled with a small layer of couplant to ensure the waves permeate. They are often employed for fault inspection for solid materials. They have different configurations in which they are made. [6]

Single element

A single element transducer is the simplest form of a transducer. It is a single homogeneous generator material with complementary components to ensure encapsulation, sound transfer etc. In terms of communication this is a single line. [6]

Dual element

A dual element transducer is essentially two single element transducers in a single housing. This allows for separation of concerns and is often used in a dual line communication. One element sending the signal and the other receiving it. [6]

Phased array

A phased array is a topology with a lot of single element transducer in one package. This allows for electronic control of the beam and focal point. [6]

2.4.2 Immersion transducer

An immersion transducer is submerged in a medium that transfers the acoustic waves to the desired object. The application is easier to do as once submerged it doesn't need an additional couplant for transmission. Furthermore, it can be focused to create a focal area, a focal point in a spherical focus, and a line in a cylindrical focus. This allows for pinpointing the desired evaluation area. [6] The focal point in a spherical focus can be defined with the following formula.

$$r_0 = \frac{0.61\lambda}{NA} = \frac{0.1\lambda}{sin\theta_0}$$

[1]

Where r_0 is the focal point size, λ is the wavelength, NA is the numerical aperture and θ The angle of the cone, See figure 2.1 for dimensions

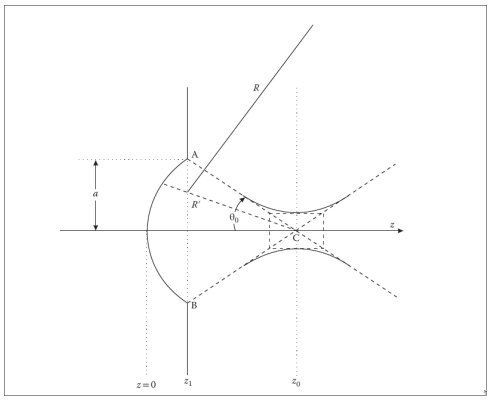


Figure 2.1: Drawing illustrating the dimensions of a spherical focus beam [1]

Chapter 3

Requirements

3.1 General Idea

This project aims to visibly show the electrolyte wetting in a battery. This can be represented with a two dimensional map showing the concentration of electrolyte. And do this at in a cost-effective way.

3.2 Desired result

A digital design is preferred to interface with existing digital technologies and allow for further processing of the data. This means the map will consist of discrete boxes with a value representing the concentration, we can define this visibly with a colour mapping the intensity to the concentration.

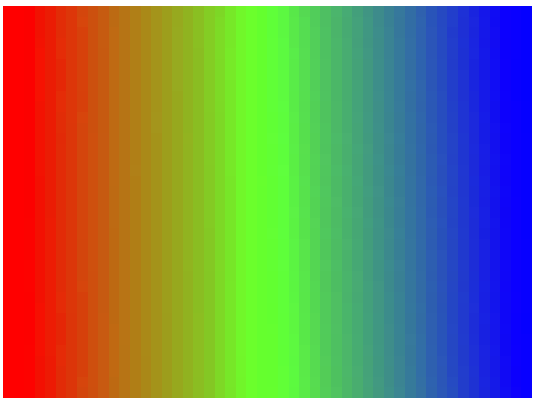


Figure 3.1: Example of a battery scan, where the Red part on the left shows a low attenuation, going from the red part on the left to the blue part on the right where the attenuation is the greatest

3.3 Component specification

3.3.1 Transducer specification

To measure these spots, an ultrasonic transducer is required. This ultrasonic transducer should be able to focus its beam to a focal point, with the scan diameter being 0.385 mm. The transducer also needs to be driven with enough power to stay above the noise floor, even at scanning spots with high attenuation. They also need to be of the immersion type, to be able to submerge them in a acoustically conducting liquid.

3.3.2 Analogue electronics specification

The transducers need to be driven with an electronic assembly. This electronic assembly should be able to drive the transducers within their specification regarding noise, voltage and frequency. On the other hand, the PCB needs to serve as a bridge between the analogue transducers and the digital upstream hardware whose final destination is the display. Specifically, it needs to be able to amplify a 2 MHz signal at a logic voltage such as 3.3 V or 5 V to a desired voltage, ensuring the signal stays above the noise floor. The receiving system should be able to handle a wide range of return voltages, ranging from almost no attenuation (good wetting) or a lot of attenuation (bad wetting). Then it should convert this into a digital signal to be read by upstream hardware.

3.3.3 Digital electronics specification

The upstream digital hardware should consist of components capable of handling the high frequency signals. It should also provide a control loop to drive the PCB components and make the scans function. It should also provide room for possible additional signal conditioning to get a clear results. The results should be two fold, the first being an analogue measurement of the attenuation, the second should be a time of flight measurement which should the time it takes between sending a signal and receiving it back through the loop. At this stage, the IO interface of this piece of hardware should be able to input a position to measure and the output should be a measurement together with a positional argument or a timestamp to aid in position determination.

3.3.4 Immersion container

The transducers require a liquid medium for the acoustic waves to permeate through. This container should have enough submerged space to house the transducers, batteries and additional supporting pieces to ensure movement. It also requires a system to hold the batteries in place and the optimal position for scanning. It should also have a means of viewing the contents in order to visibly check the optimal working of the system and aid in potential debugging. Since this component will be dealing with a volume of liquid, potentially being of greater weight, a adequate support structure should be provided.

3.3.5 Movement system

A suitable movement system should be used that can move the transducers to different spots on the battery to perform the scan. Since the image is in two dimensions, The system will

require the same dimensions of movement. The transducers will be submerged in the immersion container, so the system needs to be able to move within the container.

3.3.6 Controller

The final digital controller stage should be able to drive all the downstream components. This includes driving the movement system as well as driving the electronics subsystem. This means it needs the software tools available as well as the correct hardware to interface with the systems. It should also contain the means to receive inputs and send outputs. This can be in the form of connections to peripheral devices such as monitors and keyboard/mouse inputs.

The controller also needs to host the software displaying the wetting map. To increase usability it should also have methods to control the display and manipulate the data to give maximum insight into the measurements. This can be in the form of defining display ranges in value or in dimension of the measurement. Beside that, it should have inputs to control the machine. The inputs should be an abstracted method of the underlying operation of the machine, i.e. a button to do the full scan. Lastly it should have a method of exporting the data and move it from the machine to another computer for processing, analysing or using the data.

Chapter 4

Methods

The methods will describe the components used and the motivation for choosing them as well as comparisons.

This chapter will detail the choice of components and their rationale. This is done by defining the requirements for the components in both at a micro level, such as specific requirements and a macro scale, such as total costs and time constraints.

4.1 Transducers

Ultrasonic transducers constitute the core measurement components, requiring careful selection due to their specialized nature, significant cost (€1200/unit), and extended procurement lead times.

An online search resulted in two companies. Imasonic and Waygate Technologies. Waygate Technologies was chosen due to them providing more information regarding the working principle for the specific transducers. As well as providing detailed information on the practical aspects, such as the connector type.

Two transducers of the model Waygate Krautkrämer H 2 K were used. One for sending, one for receiving. They have a nominal centre frequency of 2 MHz, though the specific units used measured at 2.16 MHz and 2.18 MHz. and a 25 mm focal point distance. They can be driven with voltages from 0 V to the magnitude of 100 V. They are the most expensive component (1200€ per piece). With these transducers the distance between them should be 50 mm, because it needs two times the focal point distance: send → focal point → receive.

Now we calculate the wavelength using the frequency and the speed of sound in water. [7]

$$\lambda = \frac{v_{water}}{f} = \frac{1481\ m/s}{2000000\ Hz} = 0.000741\ m = 741\ um$$

Then we can calculate the angle θ using the element diameter D with radius $r_{element}$ and focal distance d.

$$\theta = tan(\frac{r_{element}}{d_{focus}}) = tan(\frac{25\ mm}{5\ mm}) = 1.3734\ rad$$

Finally we can calculate the focal resolution.

$$r_0 = \frac{0.61 \times 0.000741\ m}{sin(1.3734\ rad)} = 0.000461\ m$$

The pixel size can be defined as this value. Since the maximum battery size is defined at 100 mm by 100 mm, the total size of the image is 100 mm / 0.461 mm = 217 (216.92). resulting in a final image resolution of 217 x 217 pixels (47,089 pixels total).

Figure 4.1: Image showing the H 2 K probe.

4.2 Mini Computer

4.2.1 Architecture

A mini computer architecture was selected to optimize device compactness, portability, and cost-efficiency compared to traditional PC solutions. This approach requires essential functionalities including display output, peripheral interfaces, and versatile data I/O capabilities. Critical requirements include:

- Support for synchronous communication protocols (SPI, UART) with downstream hardware
- Sufficient processing power for display rendering and real-time data processing
- Compact form factor with standard connectivity options
- Cost-effective single-board computer (SBC) implementation
- Linux OS environment for simplified driver management, native hardware access, and reduced software complexity compared to Windows-based solutions

The Raspberry Pi 5 (8GB RAM variant) running Linux fulfils these requirements while providing enhanced performance over previous generations, with the Linux ecosystem offering superior support for embedded interfaces and development tools.

Figure 4.2: Image showing Raspberry Pi 5.

Raspberry Pi 5 Specifications [8]

The selected platform features:

- **I/O Interfaces**:
  - Video: 2× micro HDMI (dual 4Kp60 support)
  - USB: 2× USB 2.0, 2× USB 3.0
  - Expansion: 40-pin GPIO header
  - Camera/Display: 2× 22-pin CSI/DSI connectors
  - PCIe: ×1 and ×4 interfaces via FFC connector
- **Connectivity**:
  - Gigabit Ethernet with PoE+ support
  - Dual-band 802.11ac Wi-Fi (2.4/5 GHz)
  - Bluetooth 5.0/BLE
- **Power**:
  - USB-C (5V/5A) with peripheral current limit (600 mA)
  - PWM fan connector
- **Storage**: microSD card slot (for Linux OS boot)
- **Miscellaneous**:

- Dedicated UART connector
- RTC battery connector

BCM2712 System-on-Chip [9]

The Broadcom BCM2712 (16nm process) delivers 2–3× performance uplift over Raspberry Pi 4:

- **CPU**:
  - Quad-core Arm® Cortex®-A76 @ 2.4 GHz (ARMv8-A ISA)
  - Cache: 64 KB I/D per core, 512KB L2/core, 2 MB shared L3
- **Memory**: 32-bit LPDDR4X interface (17 GB/s bandwidth)
- **Graphics**:
  - VideoCore® VII GPU (1 GHz)
  - OpenGL ES 3.1, Vulkan 1.3 support
- **Video Processing**:
  - 4Kp60 HEVC hardware decoder
  - Raspberry Pi ISP (1 gigapixel/sec throughput)
  - Hardware video scaler
- **Display Pipeline**: Dual 4Kp60 HDMI controllers
- **I/O Management**: PCIe interfaces for RP1 south bridge integration
- **Linux Compatibility**: Full driver support for all hardware features

4.3 System-on-Chip Platform Selection

4.3.1 System-on-Chip Architecture

To process high-frequency signals (2 MHz pulses) from the transducers, a high-speed sampling system is required. Conventional microcontrollers were initially considered but deemed unsuitable due to limitations in sampling rates (typically $\leq$10 MSPS) and software-dependent latency. An FPGA-based solution with an external analogue-to-digital converter (ADC) was selected for its advantages in parallel processing, hardware programmability, and deterministic timing. To maintain cost efficiency and leverage the designer's expertise, the **TUL Pynq-Z2** development board featuring the **Xilinx Zynq XC7Z020-1CLG400C** SoC was adopted. This platform integrates FPGA flexibility with ARM processor versatility, enabling both high-speed signal processing and streamlined communication with upstream systems. Development tools include AMD Vivado™ and Vitis™ (no-cost licenses).

Pynq-Z2 Hardware Specifications

The Pynq-Z2 board provides the following resources:

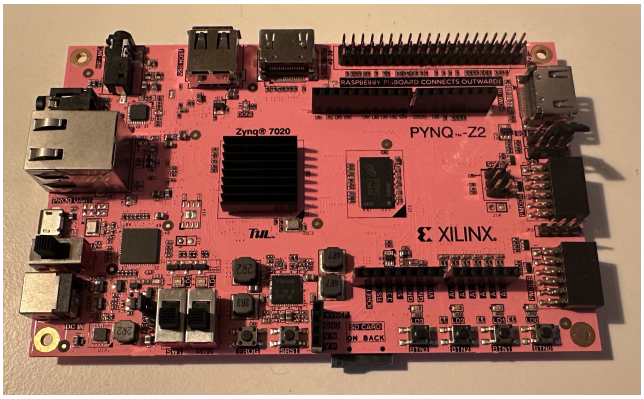


Figure 4.3: The Pynq-Z2 System on Chip development board

- **Memory:** 512 MB DDR3 RAM (16-bit bus @ 1050 Mbps), 16 MB Quad-SPI flash.
- **Storage:** MicroSD card interface.
- **Power:** 7–15 V DC input or USB 5 V.
- **Connectivity:**
  - Micro-USB with JTAG/UART bridge and USB 2.0 OTG PHY.
  - Dual HDMI input/output ports.
  - Audio: 24-bit I^2S DAC (TRRS jack), line-in 3.5 mm jack.
- **User I/O:** 4 push buttons, 2 slide switches, 4 green LEDs, 2 RGB LEDs.
- **Expansion:** 2 PMOD ports, Arduino-compatible header, Raspberry Pi connector.

[10]

Zynq XC7Z020-1CLG400C SoC

The Zynq architecture combines programmable logic and processing subsystems:

- **Processing System (PS):** Dual-core ARM® Cortex®-A9 CPU @ 650 MHz.
- **Programmable Logic (PL):**
  - 13,300 logic slices (each with four 6-input LUTs, eight flip-flops).
  - 220 DSP slices for arithmetic acceleration.
  - 630 kB block RAM for on-chip data storage.
- **Integrated Peripherals:** Xilinx analogue-to-Digital Converter (XADC) for auxiliary measurements.

[10]

4.4 PCB and electronic design

4.4.1 general overview

The PCB serves three functions.

1. Convert the input power (12 V) to a stable voltage for components (5 V)
2. make sure the signal sent from the Pynq-Z2 is amplified and buffered for the sending transducer
3. Receive the signal from the receiving transducer, amplify it and output it as an 8 bit digital signal to be read by the Pynq-Z2

Power conversion

The power is converted using a monolithic buck converter IC that steps down to voltage from 12 V to 5V.

Sending circuit

The signal received from the Pynq-Z2 is buffered using an OPAMP, ESD protected and then high pass filtered using a single capacitor NP0 100F capacitor before being sent to the sending transducer.

Receiving circuit

The receiving function starts with the signal from the receiving transducer. It firsts gets chopped at 5V to prevent high voltages from damaging the downstream components. Then it is band pass filtered to extract the signal. Then it is fed in the VGA to get the signal as best as possible within 1.024 V. This is done by setting the gain using a DAC which is controlled over the SPI protocol from the Pynq-Z2. The amplified signal is then low pass filtered to filter out the high frequency switching noise. Then it is level shifted so the differential signal has 900 mV as common mode voltage which is required by the ADC where the signal is read and converted to an 8 bit parallel digital output.

4.4.2 Variable Gain Amplifier

Component parameters

The project employs a variable gain amplifier as a means to selectively amplify the signal. It is created by chaining a low noise amplifier (LNA) with a selective attenuator. It has a gain resolution of 10 mV. and an amplification ranging from -4.5 dB to +43.5 dB in LO gain mode. It features active input impedance matching and a 3dB cut-off at 100 MHz. In terms of noise, the low noise preamplifier has a voltage noise of 0.74 nV/$\sqrt{Hz}$ and current noise of 2.5 pA/$\sqrt{Hz}$. [11]

Implementation

The input signal is first chopped off at 5 V by the bidirectional diode D1 to ensure the voltage doesn't exceed the IC maximum input voltage. The signal is subsequently filtered by C1 and

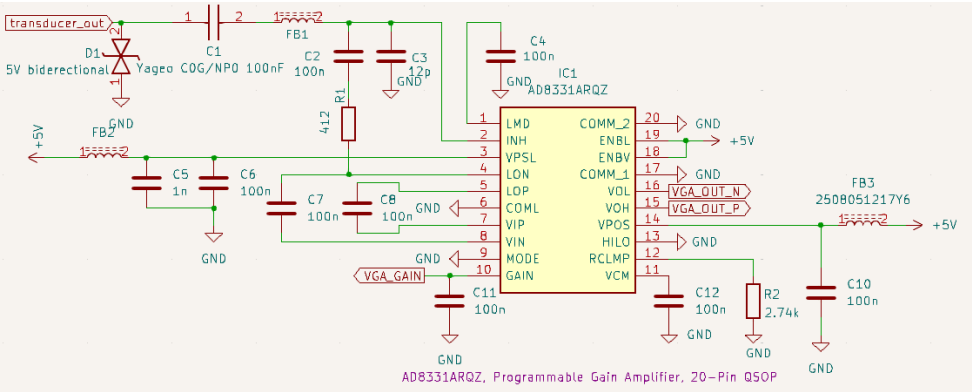


Figure 4.4: Image of the VGA in schematic view

FB1 before entering the VGA. The impedance matching is set to 75 Ω to match the transducer impedance. It is done by connecting C2 and R1 between the signal input and the low noise amplifier output and connecting C3 between signal input and ground. C2 is chosen to be as a decoupling capacitor at 100 nF. R1 and C3 are chosen according to table 7 in the data sheet. [11] C7 and C8 are 100 nF connect the LNA to the attenuator and implement DC filtering. The gain voltage comes from the Digital to Analogue Converter (DAC) and is decoupled for stability. LMD is decoupled to ground per specification, and enable and ground pins are connected to 5 V and ground respectively. The 5 V input voltage is filtered and decoupled before being fed to the VGA. The HILO pin is set to low to operate in the LO amplification mode. The RCLMP is connected to a 2.74 kΩ resistor as shown in Data sheet Table 8 to clamp the voltage at 1.0 V peak-to-peak to protect the Analogue to Digital Converter (ADC). VCM can provide a common mode input voltage between 1.5 V and 3.5 V. This is too high for the ADC, therefore the functionality is not used by bypassing it to ground with a 100 nF capacitor.

4.4.3 Digital to Analogue converter

Component parameters

The LTC2640CTS8-LM8 is an 8 bit Digital to Analogue Converter driven via an SPI interface. It has a 2.5 V 10 ppm/°C output range. It has an operating voltage range of 2.7 V to 5.5 V. A maximum integral nonlinearity of 0.5 LSB. By power on, the voltage level at power up is at mid-scale.

Implementation

The implementation requires little external components. The SPI connection pins are wired to J3 which interface with the Pynq. The output is fed to the VGA. It uses a precise voltage reference in the form of an ADR510 1.0V. Which requires a single additional current limiting resistance to enforce the voltage division. [12]

4.4.4 Level shifter

Component parameters

The AD8138 is a differential ADC driver. It is a type of amplifier that can be used in single-ended or differential mode. it has a -3 dB bandwith of 320 Mhz and a low input noise of 5nV/$\sqrt{Hz}$. It

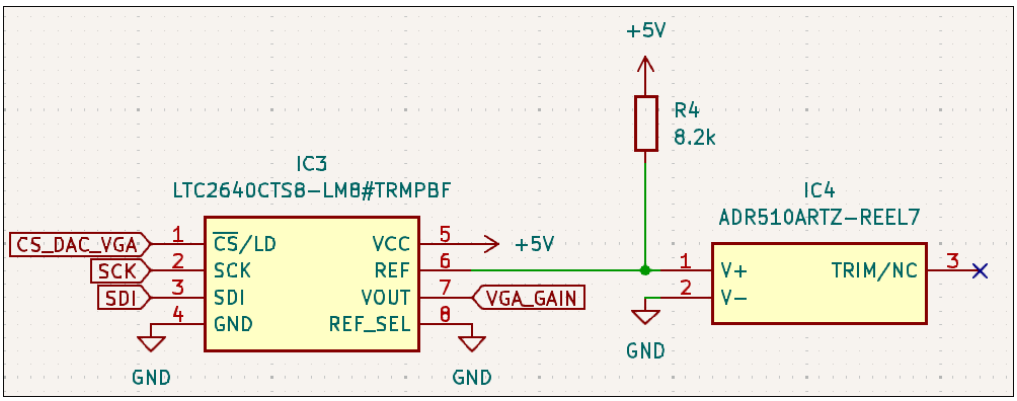


Figure 4.5: Image of the DAC in schematic view

has the property of shifting the voltage by applying the desired shifting voltage to the component.

Implementation

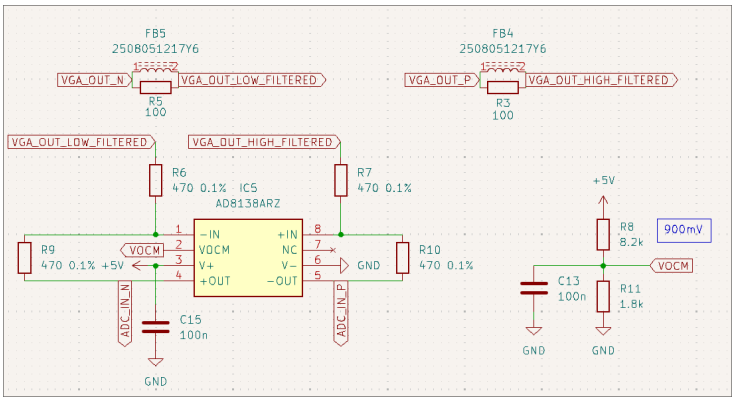


Figure 4.6: Image of the differential filters, level shifter and 900 mV voltage divider in schematic view

The differential input from the VGA is first filtered for high frequency switching noise using ferrite beads, parallel with 100 Ω resistors then it is fed through the level shifter where 2 times 2 resistors with a value if 470±0.1% Ω to give it an amplification value of 1. 900 mV from a resistive divider is fed to VOCM to set the level shift value. The output is fed to the ADC.

4.4.5 Analogue to digital converter

Component parameters

The project employs the Analog Devices AD9283BRS-100, an 8-bit resolution analogue-to-digital converter (ADC) operating at 100 MSPS. The monolithic IC integrates an on-chip track-and-hold amplifier and voltage reference, minimizing external components. It operates from a single 3.0 V supply (range: 2.7–3.6 V) with 90 mW typical power dissipation at full sampling rate, reducible to 4.2 mW in power-down mode. The device accepts differential or single-ended analogue inputs biased at 0.3 × VDVD (900 mV at V_D = 3.0 V), with a 1.024 V peak-to-peak input range and 475 MHz bandwidth. Dynamic performance includes a 46.5 dB SNR and 7.3 effective number of bits at 10.3 MHz. Digital outputs use offset binary coding with a four-cycle pipeline latency and interface natively with 2.5 V or 3.3 V TTL/CMOS logic. Housed in a 20-lead SSOP package, the ADC is rated for industrial environments (–40°C to +85°C) [13].

Implementation

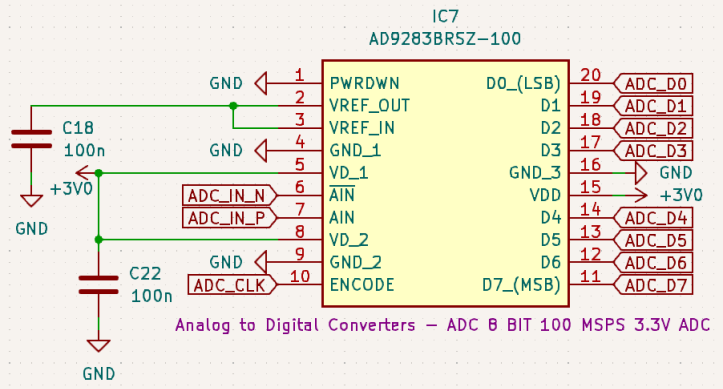


Figure 4.7: Image of the ADC in schematic view

Figure 4.7 details the implementation of the AD9283BRS-100 ADC (IC7). The device utilizes separate analogue (V_D, Pins 5/8) and digital (V_{DD}, Pin 15) 3.0 V supplies, each decoupled with 100 nF capacitors to mitigate high-frequency noise and maintain power integrity. Grounding of the PWRDWN pin (Pin 1) ensures continuous operation as power draw is not an important parameter. The internal 1.25 V reference is enabled by coupling VREF_OUT (Pin 2) to VREF_IN (Pin 3), bypassed with a 100 nF capacitor per design specification. Digital outputs (D0–D7) interface with the Pynq-Z2 through connector J2, adhering to /CMOS standards. The encode clock (Pin 10), driven by the Pynq-Z2 via J3.

4.4.6 Power management

Component parameters

The power management is driven by an AP63205 synchronous buck converter that converts the 12 V input voltage to 5 V main power voltage. This IC can handle an input voltage of 3.8 V to 32 V and can provide a 2 A continuous current. It has a 0.8 V internal reference voltage at 1% accuracy. [14]

The 3 V power for the ADC is provided by a TPS7A2030 linear regulator. It has an input voltage range of 1.6 V to 6.0 V. This variant has a fixed output voltage of 3.0 V. It has a Low output voltage noise of 7 μV_{RMS} and requires no noise bypass capacitor. [15]

Step down

Since the power supply from the CNC will be running on 24 V and we need 12 V for the components. A monolithic stepdown converter is used, converting 24 V into 12 V.

4.4.7 PCB layers and planes

The PCB is made out of four layers: a power layer, a ground layer, a signal layer and another ground layer. The layers are chosen so that the top two can provide power and ground anywhere on the board independent from routing. The two final layers are to give a clean signal layer and encase it around two zero reference ground planes.

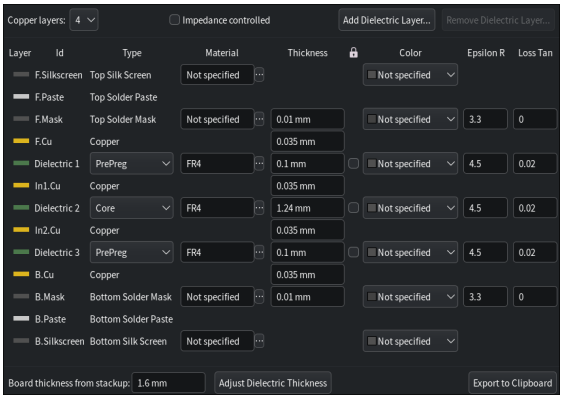


Figure 4.8: Image showing the PCB stackup

4.5 3D printer/CNC

To perform a scan, the sensors need to be moved over each virtual point on the 3D printer. The position also need to be logged to be attached to the measured value. This requires a precision assembly, in the form of a Computer Numeric Control (CNC). To cut down on development time, a ready product was chosen, a 3D printer of the type Prusa Mini. By not implementing one axis we have the required 2 dimensions of movement. This option for an off the shelf component was chosen to simplify design. This machine is within constraints of movement 100 mm by 100 mm. The model is open source, this translates to the hardware, control PCB and firmware to be freely availiable. This allows for easy development and modification of the printer. The Prusa Mini is partially constructed with 3D printed parts out of PLA.

4.5.1 PRUSA MINI Description

A 3D printer is a device that can print in 3D, it does this by using a gantry to move a printing head which extrudes molten material and solidifies it in place. This allows it to construct shapes in three dimensional space. The gantry is build with three axes driven by stepper motors. The control is precise and electronically driven. The software control for the device is based around G-code which is a set of command the machine needs to execute. The can range from configuration commands to movement commands where it needs to move to a specific place in three-dimensional space.

The PRUSA MINI is a 3D printer model by PRUSA Research. It has a build volume of 180 mm by 180 mm by 180 mm. As well as a USB port and RJ45 Ethernet port for connectivity. [16] The printer includes a 24 V 150 W power supply. [17] The heatbed itself is rated at 85 W, as shown in figure 4.9 It uses a PRUSA BUDDY MINI Board, which comes flashed with the open source firmware. [18] This firmware allow for G code commands over USB Serial connections.

4.5.2 Modification

Since the printer will be used as a CNC slave, it will not use the normal operating modes of a 3D printer and needs to be modified.

Figure 4.9: Image showing the heatbed power rating

Part removal

The 3D printer has unnecessary parts for our applications. Therefore the following components will be removed

- The extruder assembly: located on the X-axis is the assembly which extrudes the filament in 3D printer operation. Since this functionality is not used, all components are removed except for the connecting piece, which will later complement the transducer holder.
- The Y-axis: The application requires two dimensions of movement, a third one included in the printer is unnecessary. The Y-axis was removed as the X-axis and Z-axis are on an arm that can be put in a container for measurement. Which makes them more useful to remain.

Since the heat bed will not be used, there is 85 Watts of power available for the other components of the machine.

Part Addition

Parts were added to the 3D printer assembly in place of the extruder assembly to attach the transducers. Three different parts were added. The connecting piece and the spacers were 3D printed in PLA and the support arms are laser cut acrylic glass.

- The connecting piece: This piece connects to the original component connecting to the X-axis and belt. The end result gave an equal distance from the centre X-axis rods
- The spacers: The spacers are put on both sides of the connection assembly to act as physical and structural extensions for the arms to have proper distances so the transducers can be put at the correct distance from each other. They are printed from PLA.
- The Arms: These are components that extend the connection to deeper in the basin where the transducers are submerged. They are made out of acrylic glass to ensure rigidity as PLA proved to be too pliable. They also have two reinforcements each for both parts of the shape. One set of reinforcements was left out through unintentional omission.

G code

Since there is a Serial port over which the a master device can communicate with a slave, G code can be put directly in the printer. The G code commands are transferred as a string in a UART

communication at 115200 BAUD speed. The following commands are used to control the 3D printer

- G28: this command resets the axis by trying to reach the highest value on the X-axis and lowest value on the Z-axis. These points are defined as (X,Z) (0,0).
- G0 Xx Zz: this command instructs the machine to move to the desired position of x X and z Z.
- M220 Sx: this command sets the speed with x being the percentage of the maximum speed

4.6 Basin

Since the transducers need to be submerged in a liquid for their acoustic waves to transmit, a container capable of holding the liquid, as well as be attached to the rest of the CNC assembly. Furthermore it houses a battery holder assembly, consisting of two plates with a window. The battery can be put in the window and the wrapping can be sandwiched between the plates and screwed tight. The plates have rests attached to them that can rest on top of the basin, leaving the battery hanging submerged. This assembly allows for easy battery replacement.

4.6.1 Design

The design is a box with an internal dimensions of 300 mm by 300 mm by 120 mm. The material used is acrylic glass to allow for visual inspection of the measurement. It gives information on the current position of the transducers as well as the position of the battery in the holder. By the nature of laser cutting manufacturing, the basin and battery holder were cut in a flat piece of acrylic. Afterwards the pieces had to be glued together to form the 3D shape. This was done using hard plastic glue. To aid with structural integrity a teethed construction was used to maximise connection area on which glue can be applied. Afterwards the basin was sealed waterproof with silicon sealant.

4.6.2 Liquid

The transmissive liquid was chosen to be water and 10 cSt silicon oil. Firstly the water was used for testing and early measurements. Since water is easy to procure and dispose of. Afterwards silicon oil was used because of it inert properties and proven functionality [3], making it safer to use with lithium based battery chemistries

4.6.3 CNC attachment

To fix the basin to the CNC, and make sure it is aligned, a supporting piece was 3D printed. This piece can slot directly into the CNC assembly on one side, levelling it with the top of the CNC basis. On the other side two feet support the basin, bringing it also level with the rest of the construction. Finally a border around the basin is included in this attachment. This allows for the attachment to be put in place and the basin simply to be slotted in the attachment to attach it well to the rest of the machine.

4.7 Software and HDL

The software/HDL is the code used to program the Raspberry Pi and Pynq-Z2. The Pi uses python as a high level and library rich language to achieve good results with less effort compared to lower level languages. This comes at a cost of less fast code, but the Pi has enough processing power to make this point irrelevant. The Pynq uses two languages for programming, C is used to program the Processor and VHDL is used to program the FPGA. The choice for C over vitis HLS is mainly due to the designer being more familiar with the language. The same applies for VHDL on the FPGA where it was chosen over the language Verilog due to the designer being more familiar with it. Each level of code provides an abstracting of the functionality i.e. each functionality is intended to be limited to its inputs and outputs.

4.7.1 Raspberry Pi

The raspberry pi uses Raspberry pi OS, which is a fork of the Linux distribution Debian. Debian is a verson of linux, called a distribution or distro, that is known for its well tested packages (pieces of software) and stability. The downside for this is that newer features take longer to be released. Raspberry Pi OS however has custom packages for the Pi ecosystem and has the necessary features for this application.

The Raspberry Pi was programmed using the language Python. Its design handles three main topics. User input, user output and back end

User input

The user input handles the controls for the machine. Such as scan start, scan type setting, but also the interaction with the output map and file exporting.

User output

The user output mainly handles the view of the battery map. This is in both the analogue values and the time of flight measurement. It also needs to allow for zooming and elaboration on points, such as its value and position.

Back end

The back end controls the CNC assembly and reads from the Pynq-Z2. It initiates serial connections with both the devices over USB. Then when a scan is started, it will give the appropriate G code commands to the CNC. It can scan in continuous mode or intermittent. For speed continuous mode is preferred, but intermittent is more straightforward to implement. In continuous a single command is issued to scan a line. During this, the Pynq-Z2 will be sampled at constant intervals. By knowing the time of movement of the sensor head and the timestamps from the measurement, it can be linked back to the position of the measurement. This requires careful tuning. For that reason, intermittent scans were implemented first. Here a single unit of x mm is moved then it is scanned. This loop is repeated. While it gives a more straightforward result, the CNC is slower to do this movement as it needs time to process the command each time, introducing latency. During the scan and after it is completed, a 2D map of the results can be

plotted to the front end. The scan is saved in a data structure which can be easily exported to a csv file or image file.

4.7.2 SoC FPGA design in VHDL

The FPGA design interacts directly with the PCB, it is completely digital and therefore works with the digital connections on the PCB. It can be divided in three parts. The ADC driving, the pulse driving and the VGA gain control driving. The design at the highest level is a block design connecting all the relevant block abstractions.

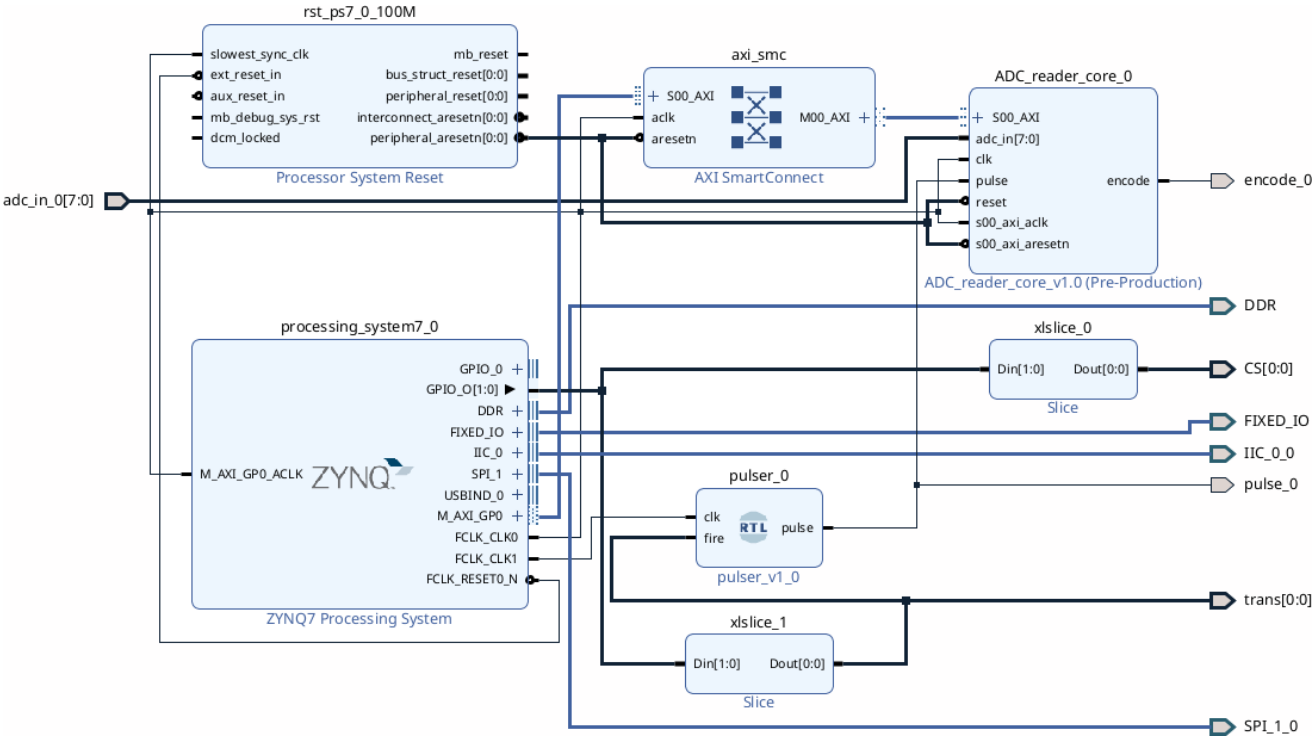


Figure 4.10: The high level block design for the FPGA. The left shows the processing system with its outputs on the right. It also shows the AXI connection to the ADC_reader_core_0 block with the helper blocks rst_PSY7_0 and axi_smc. It also shows xslice blocks which split the GPIO vector for output and for feeding it into the pulser_0 block.

Processing system (PS)

The processing system (PS) is the implementation of the virtual block that connects the processor to the programmable logic in the FPGA. It mainly allows for controlling output clocks and peripheral IO.

For clock outputs it has two.

- FCLK_CLK0 at 100 Mhz to generate the clocking signal for the AXI peripherals as well as the clock signal for the ADC.
- FCLK_CLK1 at 2 Mhz to supply the correct frequency to the pulser to generate the 2 Mhz pulses

It has multiple (used) outputs

- GPIO: these are for controlling the pulser and as a chip select

  - GPIO0: Chip select, this replaces the chip select from SPI since the DAC requires an active low signal, which the Xilinx driver does not supply upstream
  - GPIO1: This signal starts a pulse run by activating the pulser block.
- SPI_1: This output connects directly to the DAC and allows the processor to drive it directly, the Chip select pin is externally controlled by the GPIO
- IIC_0: This is the I2C output, which is intended to be used by the raspberry pi to read the processed data from the processor.
- M_AXI_GP0: this connects the PS to the ADC_reader block, allowing it to read the values directly as Memory Mapped IO.

It has more outputs, but these are mostly generated by the design software itself and provide no functional value to the design. It also generated reset and clocking signals for the AXI communication.

ADC driving

The ADC driving can be split in input and output. An output for the encode pin, making the ADC sample. This is internally connected to the system clock, making the device sample constantly. This is because the input is constantly reading. The input is an 8 bit vector representing the analogue voltage at the ADC input in a range of 0 V to 1.024 V.

The ADC reader consists of a state machine with three states: Idle, Expecting and Reading.

- Idle: This is the neutral state where there is no internal process happening. In this state the calculated data is already in the output registers. The data is thus deemed good to be read, and an according flag is set to signal it. This state is entered when the reading state goes into timeout.
- Expecting: This state is changed to when in idle and a signal that a pulse is sent is received from the MCU. In this state the timer for the Time of Flight data is started.
- Reading: This state is entered when during the expecting state, a signal is detected by the ADC. This stops the Time of Flight counter and starts reading the signal on the ADC and extracting the amplitude of the signal. This data together with the Time of Flight duration is put in the output registers. When no more signal is received, the state switches after a timeout back to Idle.

The values read by the ADC are filtered for any noise by using a simple moving average filter with a window size of 9 point.

The output registers can be read by the processor via a custom AXI block using the AXI lite interface. This allows the registers to be read as Memory Mapped IO. There are four registers serving different purposes, including data and debug. There are four registers in total:

1. The measured amplitude of the signal
2. The Time of Flight measurement
3. zeros prefix before current state and read_ok signal
4. alternating 0 and 1 for 24 units as a debug state, followed by the current 8 bit ADC value.

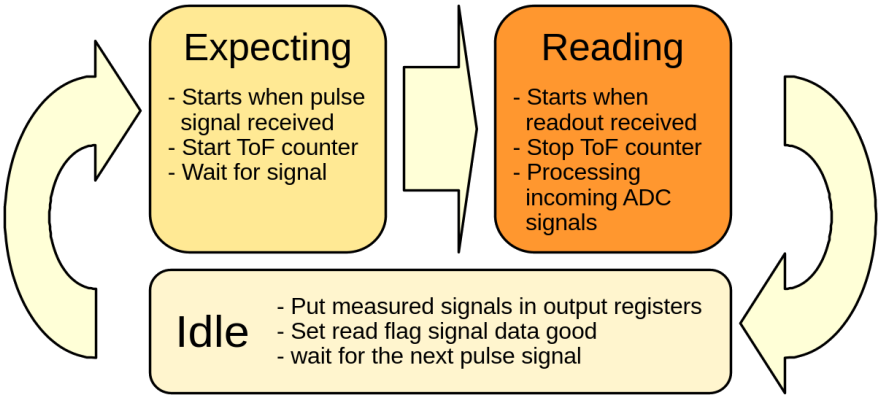


Figure 4.11: Flowchart illustrating the state machine of the ADC reader HDL design

Pulser

In order to send the pulses to the transducers in a controlled manner a subcircuit is made. This subcircuit connects the clock to the output for ten pulses if a signal is applied. It waits until it is low to reset itself to prevent looping if the signal application is slower than the ten pulses at a combined duration of 5 µs. The external signal is provided by a GPIO from the processing system.

4.7.3 SoC processor design

The processor host the C code which mainly abstracts the low level functions of the FPGA to a value-timestamp format. In its function it drives the FPGA functions. The functions can be subdivided into three parts:

- Pulse driving: using a GPIO to start a pulse train.
- SPI gain control: Set the DAC using SPI to set the VGA gain
- Read the ADC_reader block: read the three registers mentioned above to get the analogue value and the TOF value. As well as reading the read_ok flag. Besides those, it can read debug data such as current state and raw ADC value.

Reading

A read is typically performed by setting the gain value followed by a pulse, followed by a read. In the demo code, the read_ok flag was not read, since the delay between CPU instructions is greater than the reading action.

Since the read value is tied to the amplification a formula can be used to calculate the "absolute" value.

$$U_{abs} = U_{adc}/GAIN$$

Since the value read on the ADC is already amplified, the value read does not correspond the the true value. By in software dividing this value by the gain set, the actual voltage at the transducer output is calculated.

This mechanism allows the machine to read a large range of voltages using lower range hardware.

Because of this effect, and the fact that the amplification change is subtle between gain steps. The demo code tries to find the moment the analogue voltage is 200 (combining to about 288 mV) and using the set amplification as a baseline measure value.

Auto search algorithm

To look for the amplification value at 200 ADC out, a searching algorithm had to be used. Binary search was implemented to look for the optimal value. This binary search featured beside the algorithm itself, a sample reduction when it was close to its value, as well as a hard stop at 30 searches, to ensure timely handling of the measurement.

Filtering

Also on this level filtering was done at first a filter was employed by taking samples, ordering them and taking the average of the 33% middle values. This improved measurement consistency due to the significant signal noise.

The first algorithm used was bubble sort. However since this was an $O(n^2)$ sorting algorithm, a processing overhead became dominant, especially at higher sample sizes. This was subsequently replaced with an algorithm found on an online repository that fixed the overhead.

Chapter 5

Results

5.1 measurement

The measurement is not at an accuracy level yet where battery scanning can be done. It has a drift over its range and posses significant noise. It can however differentiate between different materials. It can differentiate between acrylic glass, water and aluminium.

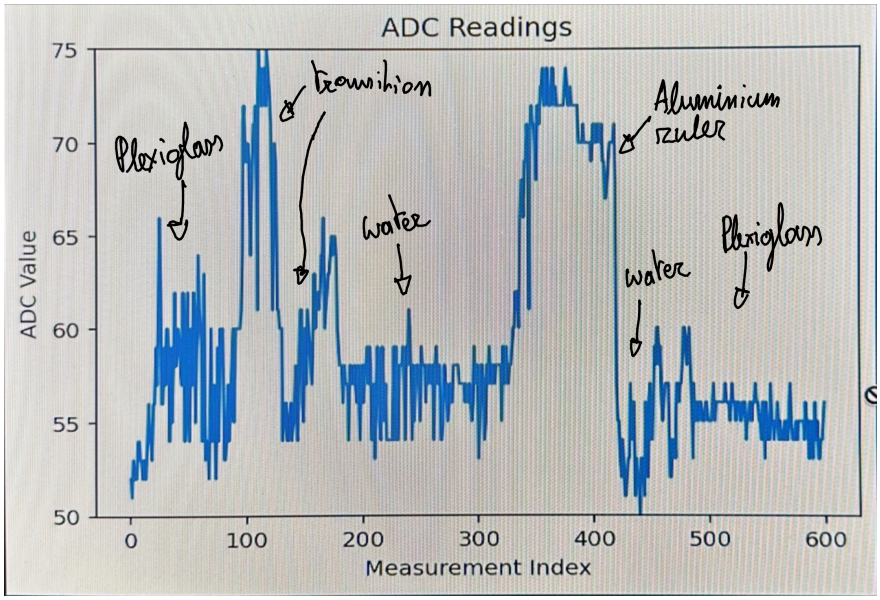


Figure 5.1: graph returning the measured results. The x axis shows the measurement in an asynchronous measuring loop. The y axis show the amplification needed for the ADC to read a value of 200.

Figure 5.1 shows the scan with multiple different artifacts. First the signal varies peak to peak in a single material at 10 amplification points. It peaks around material transitions. Note near the end of the scan the value at the acrylic glass is lower. Figure 5.2 Shows the picture of the setup where a line is scanned from right to left, and the result corresponds with the results on figure 5.1.

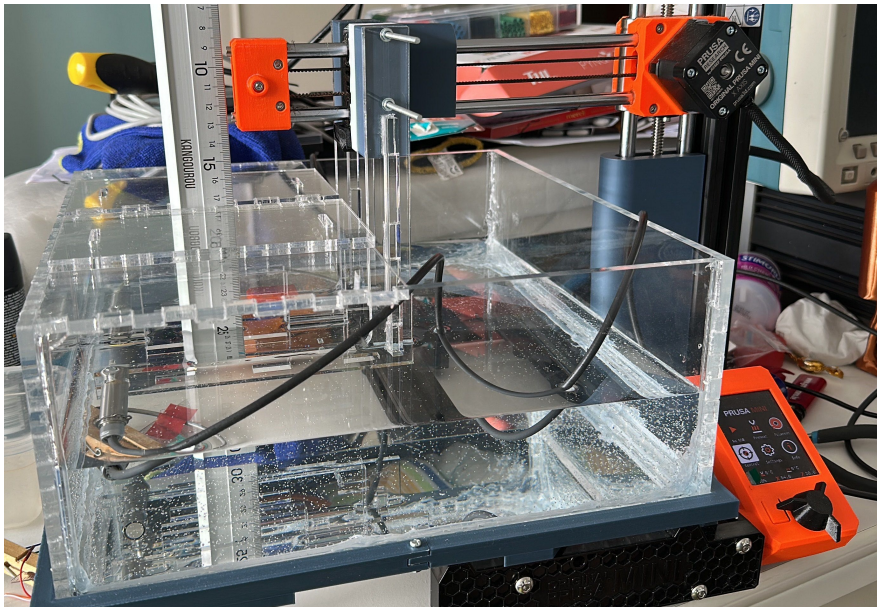


Figure 5.2: Image of the scan setup

5.2 machine

The machine itself has been built and can achieve a linear scan and differentiate between materials with different attenuation values. The design itself has some shortcomings on the implementation. This improvements will also be described in this section as well as potential improvements.

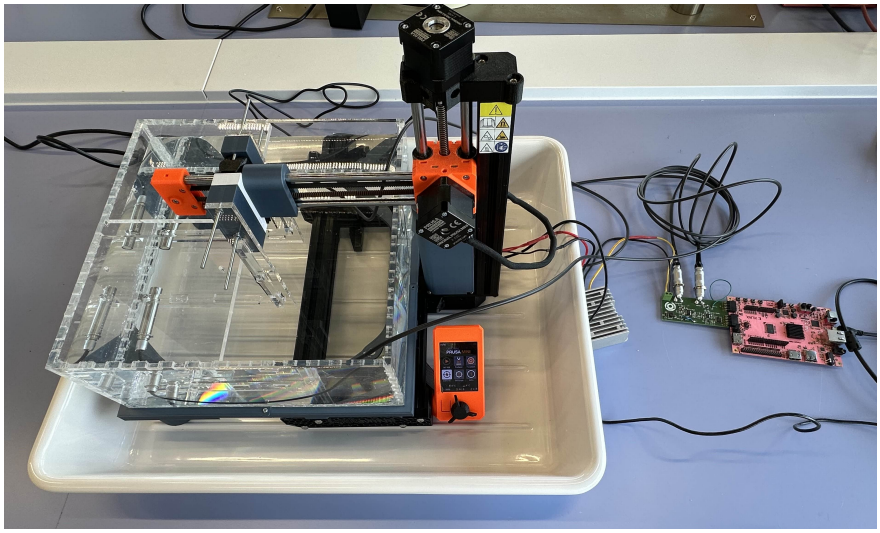


Figure 5.3: Full image showing the CNC with the basin and arm structure holding the transducers. The basin is filled with water. On the right is the 24 V to 12 V step down converter, followed by the PCB and the Pynq-Z2. The Raspberry Pi is absent from the picture.

5.2.1 Transducers

The transducers work as expected, they can send and receive a 2 MHz signal and can be attached successfully with LEMO connectors.

5.2.2 Mini Computer

The Raspberry pi 5 works as intended and can drive the devices by accessing the connections on the Serial Port.

5.2.3 System on Chip

The system on chip works within specification. All connections can be made successfully. The FPGA uses only a small fraction of its resources, with 95% of its logic slices remaining. Specifically, it uses 2.02% of slice LUTs, 0.01% of LUT as RAM, 0.90% of slice registers and 16% of IO paths.

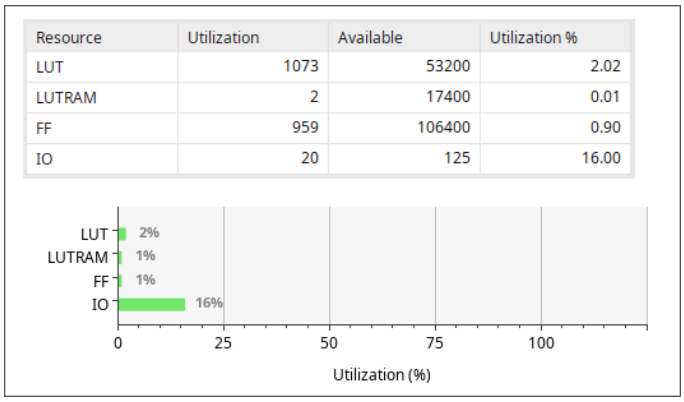


Figure 5.4: Table and graph showing the resource utilisation on the FPGA. It shows the slice LUTs (LUT), LUT as RAM (LUTRAM), slice registers (FF) and used IO paths (IO). This is generated in Vivado

The MCU uses a negligible amount of memory.

5.2.4 PCB design

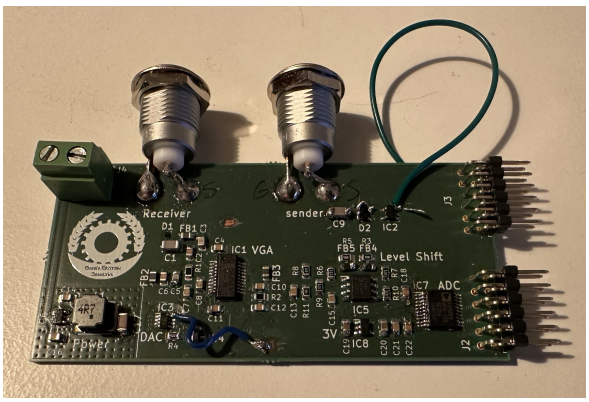


Figure 5.5: The final PCB, with debug wires and fixes

The PCB design works as intended. The power conversion works, outputting a stable 5 V from a 12 V input signal. The sending signal loop was modified by removing the ESD protection diode D2. Due to the diode not switching fast enough to pass the 2 MHz signal. The receiving signal worked as well, the signal got amplified and read in the ADC, the DAC signal was controllable via SPI but it only a written value of 0 to 100 changed the voltage. The DAC did initially not

respond, the cause was that PIN8 was connected to ground. By connecting it to 5V, the DAC worked as intended.

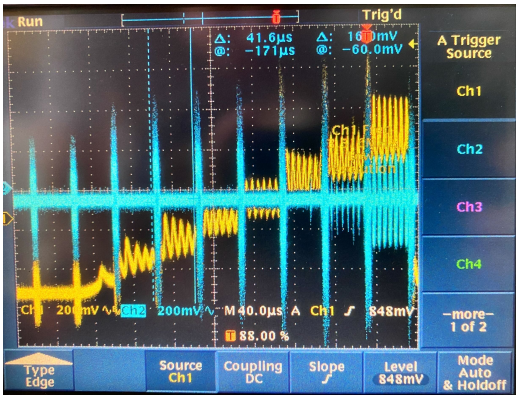


Figure 5.6: Oscilloscope image showing the DAC voltage (yellow) and pulses after being amplified by the VGA (blue). It shows the connection between the set GAIN voltage and the output, as the higher the DAC voltage, the higher the amplification on the output values. The DAC value was incremented with 10, corresponding to 100 mV steps, between a value of 0 and 100

The PCB was connected to the SoC using two male 12 pin connectors on the PCB, connecting to the two PMOD ports on the side of the SoC.

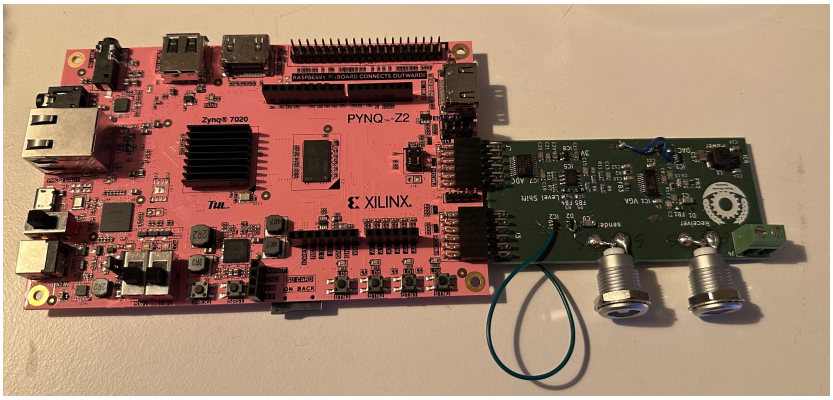


Figure 5.7: Image showing the PMOD connection between the PYNQ-Z2 and the PCB

5.2.5 3D printer/CNC

The 3D printer was successfully hacked and could be driven using the G code. There was a processing delay between commands. Causing the commands to have a minimum spacing of 500 ms. The modification was also successful as the unnecessary parts were removed and the connecting parts were attached to the extruder assembly. The standard firmware was used.

5.2.6 Basin

The basin was successfully constructed, the initial cutout was 200 µm too large and had to be filed down to fit. The watertight seal was successfully applied. The CNC attachment was 3D printed and required some filing to fit in the CNC assembly. It was printed in four parts to fit on the printer and were screwed together. The basin fits tightly in the attachment. The battery holder can be inserted under an angle, if the X-axis is moved all the way up. The basin was filled with water.

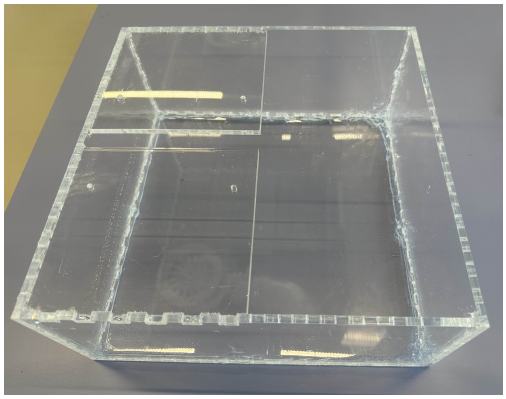


Figure 5.8: Image showing the basin

5.2.7 HDL and Software

The software was developed as a minimal viable product for demonstration purposes. It has more room for improvement in accuracy, robustness and user quality.

Raspberry Pi 5

The Raspberry pi Software consists of a single python file, performing a single fixed scan. It connects to the Pynq-Z2 and the CNC via serial connection. It then recalibrates the CNC and puts it in its starting position. It then performs the scan by either moving it in pulsed scanning, or a single continuous scan. the results then gets plotted.

Pynq-Z2 - FPGA

The software on the Pynq-Z2 FPGA was adequate and read the results at a desired speed. The accuracy of the scan was unstable however. with deviations on same point between scans ranging for 5 points.

Pynq-Z2 - MCU

The MCU software worked to drive everything and get the desired results from the FPGA stage. The timing however, is unstable. The search algorithm isn't efficient, as accurate measurements can take multiple cycles of 300 sample size scans. This combined could bring the time per scan up to 60ms.

Chapter 6

Discussion

While the project achieved a goal, it leaves room for a lot of improvements.

6.1 Mechanical

The mechanical aspect was not satisfactory. It had an amount of wobble which negatively impacted the scan. Furthermore, it introduced an error near the far end of the scan as seen in figure 5.1 illustrating the difference between the first part of acrylic glass and the last part.

With these issues and the ones discussed below, the 3D printer and the laser cut part do not provide the accuracy necessary to do accurate scans. A complete redesign of the mechanical part would solve these issues. As a structurally more rigid gantry can be provided as well as code specifically designed for this application.

6.1.1 Transducer fixation

The transducer is fixed in place using an M3 screw which tightens it to the hole in the arm. This method has large tolerances and this negatively impacts the scans. It is possible to change the angle at which the transducers are positioned. By changing the alignment, the focal point changes and the point of measurement is unknown, voiding the data. This can be fixed by adding a second set of arms to enforce the alignment. Because of the large tolerances the transducers weren't exactly parallel, which also causes signal loss. In this case, the read spot should be the focal point of the sending transducer.

6.1.2 3D printer full range

The full range of the 3D printer was not reached. This was an error caused by the extended margins of the arms and battery holder, causing the top part of the window in the battery holder to be unreachable. This is not necessarily a problem for smaller batteries as they can be position lower, still within the scannable area.

6.2 Electronics

While the electronics performed a full read cycle, there are areas of improvement.

6.2.1 First PCB design

The PCB design was functional as intended besides two points, the ESD protection diode D2 and the wrongly connected pin 8 of the DAC. It also lacked output pins for 5 V.

6.2.2 Transducer driving voltage

In this implementation, the transducers are driven at 3.3 V. This was chosen mainly due to a lack of clarity on what voltage should be used, as well as having a preliminary method to get some result out of the PCB before the deadline. This proved to be insufficient and caused the received signal to be low and requiring a higher level of amplification to work. This causes two problems: First, the signal is closer to the noise floor and other environmental noise. This also gets amplified and causes the need for more aggressive filtering. Secondly The measurement range is smaller, as a neutral reading at theoretical maximum received signal is read at 60% amplification. This leaves half of the amplification potential unused. Since 24 V is available this should boost the voltage by about 7 times or an improvement equivalent to 17 dB.

Another potential improvement would be to lift the voltage beyond 24 V. This can be done using a boost converter, and can lift the voltage to higher voltages such as in the 100 V range. Which would generate an even stronger signal. Precaution should be taken, however, that in case of good transmission that the power should not exceed the protection on D1. Which is defined by a maximum current of 1.8 A over a period of 8/20µs [19].

6.2.3 filtering

Filtering was introduced on datasheet recommendation. However, they were not tested if they actually influenced the measurements. Since the signal has quite some noise, this could be a source, or lack of adequate functioning, of these filters. Therefore, testing the filters at different frequencies and if they actually filter the items should be done.

6.2.4 Power supply input voltage

The 5 V power supply input voltage is 12 V, while the input voltage from the 3D printer is 24 V. At first this was done to power the Pynq of 12 V, however it can also be powered from 5 V. Therefore the step-down converter from 24 V to 12 V is unnecessary and could be removed by feeding the converter directly with 24 V. Since the controller IC is monolithic and has a higher input voltage, this should work without any adjustments to the design.

6.2.5 PCB general connections

The first PCB revealed shortcomings in operation. It was not clear when the PCB was on due to the lack of status indicator LEDs. These can easily be added with an LED and resistor on the 12/24 V trace and on the 5 V trace. There is also no output for the generated 5 V to the Raspberry Pi and the Pynq. Therefore terminals should be applied. The Raspberry Pi 5 is recommended to be used with a 27 W power supply over the Power Delivery (PD) protocol. [8] This implies it could draw this wattage and the buck converter should be able to provide this. For the final connection to the Pi, a higher power target as well as a PD negotiation IC should

be provided for full power operation. However, since the demands of the control software aren't as intense. It should be able to run on a lower power target.

6.3 Software

The software did not meet end product specifications and beside implementing the remainder of the functionality some optimisations could already be done.

6.3.1 Pynq-Z2 to Raspberry Pi connection

The connection over USB between the Pynq-Z2 and Raspberry Pi is unstable as it disconnects often. A connection over I2C would be better and more predictable.

6.3.2 Search algorithm optimisation

The search algorithm was implemented in a simple manner to meet a demo stage. It was not optimal and used too much time to calculate. Multiple optimisations can be implemented to improve the performance.

Binary search replacement

While binary search is a very simple search, it is rather inefficient. In the implementation it took samples and checked if they were above or under the 200 ADC value mark. But since the read value and amplification combination is a representation of the real value, it can be used as a heuristic to calculate the optimal amplification where the resolution is highest. This would besides increasing speed, reduce the scans per value read to two.

Amplitude extraction improvement

The method to extract the amplitude is to make a moving average filter and taking the maximum value. This method however, is prone to noise. While the moving average filter should take out transients, it also reduces the read amplitude, decreasing the maximum reading accuracy. A point not addressed by this filter is the fact that noise with a constant frequency can be introduced and this can be superimposed on the original signal. A way to solve this is using a Fast Fourier Transform. Xilinx provides an IP called the XFFT to perform this. It was briefly considered to be implemented but it was not made functional within time. This should fix most issues of noise on the measured signal as it provides a more robust method of amplitude extraction. Another method could be to provide a band pass filter before the ADC stage to only let the MHz signal pass.

Time of Flight debugging

A time of flight measurement was implemented and worked when simulating the ADC_reader block. However, the MCU could not receive a valid reading after each scan despite the ADC value being read. This was not further explored due to time constraint. In case this is debugged, it can provide a second source of wetting data.

Sorting in hardware

The sorting algorithm was implemented in software on the MCU, but this takes away processing time that can be used for further scans to increase accuracy. Since an FPGA is also used, the sorting algorithm could be moved to hardware and by using parallelism and hardware sorting algorithms. The sorting speed could be greatly accelerated. To implement this. The same procedure of the ADC_reader block can be used. By making a custom AXI block and putting all the data in there to get the result. This could be even more optimised by also doing the selection and averaging in hardware or even modifying the ADC_reader block to make a direct pipe into the filter.

Chapter 7

Conclusion

The machine demonstrated minimum capabilities by scanning a straight line and differentiating materials. This proves the machine is capable of handling the ultrasonic scanning loop. Consisting of sending a pulse, amplifying it, receiving it, amplifying it again and reading it. Then taking this data and using it in combination with movement from the gantry to make a plot taking all the different parameters into account.

Despite these achievements, the device did not meet the expected goal of scanning a battery and mapping the battery wetting. Besides the minor improvements the main issue is accuracy. The accuracy can be subdivided in two types: The positional accuracy and the signal accuracy.

The positional accuracy entails the accuracy of the gantry to move correctly to a position as well as the rigidity of the entire setup to accurately transfer the position intended by the stepper motors to the transducers.

The signal accuracy is a more complex accuracy issue as it is an accumulation of noise through different components. First the signal generated from the FPGA gets amplified before being looped through the transducer setup where it gets attenuated. Then it gets amplified again and level shifted before being read by the ADC where we receive the noisy digital signal. Then different stages of hardware filtering and software filtering are applied to get a final value for the position.

Addressing these issues would improve the performance of the machine and make it more likely to perform the battery scan. Therefore this platform provides a basis to iterate on to construct a functional automatic battery electrolyte scanner.

Bibliography

[1] J. D. N. Cheeke, *Fundamentals and applications of ultrasonic waves*. Boca Raton: CRC Press, 2nd ed (online-ausg.) ed., 2012.

[2] M. Ghalkhani and S. Habibi, "Review of the Li-Ion Battery, Thermal Management, and AI-Based Battery Management System for EV Application," *Energies*, vol. 16, p. 185, Dec. 2022.

[3] Z. Deng, Z. Huang, Y. Shen, Y. Huang, H. Ding, A. Luscombe, M. Johnson, J. E. Harlow, R. Gauthier, and J. R. Dahn, "Ultrasonic Scanning to Observe Wetting and "Unwetting" in Li-Ion Pouch Cells," *Joule*, vol. 4, pp. 2017–2029, Sept. 2020.

[4] K. Nakamura, ed., *Ultrasonic transducers: materials and design for sensors, actuators and medical applications*. No. no. 29 in Woodhead publishing series in electronic and optical materials, Cambridge, UK: Woodhead Pub, 2012.

[5] M. Smith and S. Kar-Narayan, "Piezoelectric polymers: theory, challenges and opportunities," *International Materials Reviews*, vol. 67, pp. 65–88, Jan. 2022. Publisher: SAGE Publications.

[6] Baker Hughes Company, "Krautkrämer ultrasonic transducers For flaw detection and sizing," June 2023.

[7] "Water - Speed of Sound vs. Temperature."

[8] "Raspberry Pi hardware - Raspberry Pi Documentation."

[9] "Processors - Raspberry Pi Documentation."

[10] Technology Un-Limited, "PYNQ-Z2 Reference Manual v1.1," tech. rep., Oct. 2019.

[11] Analog Devices, "Ultralow Noise VGAs with Preamplifier and Programmable RIN," AD8331/AD8332/AD8334 Datasheet, 2006. Revision I.

[12] Analog Devices, "1.0 V Precision Low Noise Shunt Voltage Reference," ADR510 Datasheet, 2007. Revision B.

[13] Analog Devices, "8-Bit, 50 MSPS/80 MSPS/100 MSPS 3 V A/D Converter," AD9283 Datasheet, 2001. Revision C.

[14] Diodes Incorporated, "3.8V TO 32V INPUT, 2A LOW IQ SYNCHRONOUS BUCK WITH ENHANCED EMI REDUCTION," AP63200/AP63201/AP63203/AP63205 Datasheet, 2024. Revision 3 - 2.

[15] Texas Instruments, "TPS7A20 300mA, Ultra-Low-Noise, Low-IQ, High PSRR LDO," TPS7A20 Datasheet, Mar. 2020. Revised July 2024.

[16] "Original Prusa MINI+ Semi-assembled 3D Printer | Original Prusa 3D printers directly from Josef Prusa."

[17] "External PSU 24V 150W | Original Prusa 3D printers directly from Josef Prusa."

[18] "prusa3d/Prusa-Firmware-Buddy," Aug. 2025. original-date: 2019-12-19T14:33:13Z.

[19] Nexperia, "PESD5V0X1BCAL Extremely low capacitance bidirectional ESD protection diode," PESD5V0X1BCAL Datasheet, 2023.

Appendix A

Attachment - Controller software

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jul 25 16:37:28 2025

@author: bram
"""

import serial
import time
import re
import matplotlib.pyplot as plt

def ser3D_write(string):
    ser3D.write((string + "\r\n").encode())

def get_ADC():
    # Define regex pattern to match 'power: <number>'
    pattern = rb'power:\s*(\d+)'  # r for raw string, b for bytes pattern
    matches = []
    # wait until valid string
    while len(matches) == 0:
        data = serADC.read_all()
        time.sleep(0.01)
        data = serADC.readline()
        print(data)
        matches = re.findall(pattern, data)
    return matches[0]

def pulsed_scan():
    values = []
    # move and scan in loop
    for x in range(180, 63, -1):
        ser3D_write("G0 X" + str(x))
        time.sleep(0.5)
        values.append(int(get_ADC()))
        print(get_ADC())
        print(x)
    return values

```

```
def running_scan():
    values = []
    #start move and scan 200 times during movement
    ser3D_write("G0 X64")
    for x in range(0,200):
        ser3D_write("G0 X" + str(x))
        time.sleep(0.5)
        values.append(int(get_ADC()))
        print(get_ADC())
        print(x)
    return values


ser3D = serial.Serial('/dev/ttyACM0', 115200);
print(ser3D.name)

serADC = serial.Serial('/dev/ttyUSB1', 115200);
print(serADC.name)

# reset axes and set into neutral position
ser3D_write("G28")
time.sleep(10)
ser3D_write("G0 X180 Y0 Z20")
ser3D_write("M220 S20")
time.sleep(4)

# perform scan
values = pulsed_scan()

# plot result
plt.figure()
plt.plot(values)
plt.ylim(min(values), max(values))  # Set y-axis limits to min/max values
plt.title("ADC Readings")
plt.xlabel("Measurement Index")
plt.ylabel("ADC Value")
plt.show()

ser3D.close()
serADC.close()
```

Listing A.1: Controller code for Raspberry Pi

```

/***************************** Include Files ********************************
    */

#include "xparameters.h"  /* EDK generated parameters */
#include "xspips.h"   /* SPI device driver */
#include "xil_printf.h"
#include "sleep.h"
#include "xgpiops.h"
#include <xil_types.h>
#include "xiltimer.h"
//#include <cstdint>

```

```c
/************************** Constant Definitions *****************************/

/*
 * The following constants map to the XPAR parameters created in the
 * xparameters.h file. They are defined here such that a user can easily
 * change all the needed parameters in one place.
 */

#define SPI_DEVICE_ID   XPAR_XSPIPS_0_DEVICE_ID
#define GPIO_DEVICE_ID    XPAR_XGPIOPS_0_BASEADDR



/*
 * The following constant defines the chip select value used to select
 * the device on the SPI bus. This value corresponds to the slave select
 * signal, which is typically connected to the chip select (CS) pin of the device.
 */
#define VGA_SPI_SELECT 0x01

// Define the connection pins for the SPI interface
#define CS_PIN 54  // Chip Select pin
#define CLK_PIN 55 // Clock pin

// Delay define
#define GENERAL_DELAY 100000 //us

// Array size for the filter
#define ARRAY_SIZE 250

/***************** Macros (Inline Functions) Definitions *********************/

/************************** Function Prototypes ******************************/


void SPIWrite(XSpiPs *SpiPtr, u16 Address, u8 ByteCount,
        EepromBuffer Buffer);


int StartSPI(XSpiPs *SpiInstancePtr, UINTPTR BaseAddress);

void DAC_write(u8 value);

void launch_pulse();

void print_ADC_memory();

void bubbleSort(int arr[], int n);
void mySort(int arr[], int n);

int auto_search(int amp, int power, int depth);
```

```
64
65 float array_read(int size);
66
67 /************************** Variable Definitions ***************************
   ****/
68
69 /*
70  * The instances to support the device drivers are global such that the
71  * are initialized to zero each time the program runs.  They could be local
72  * but should at least be static so they are zeroed.
73  */
74 static XSpiPs SpiInstance;
75 static XGpioPs Gpio;  /* The driver instance for GPIO Device. */
76
77
78 /*****************************************************************************
   ****/
79 /**
80  *
81  * Main function to call the Spi example.
82  *
83  *
84  * @return  XST_SUCCESS if successful, otherwise XST_FAILURE.
85  *
86  * @note    None
87  *
88  ******************************************************************************
   ****/
89 int main(void)
90 {
91   int Status;
92   int DelayCount = 0;
93   XGpioPs_Config *ConfigPtr;
94
95
96   xil_printf("SPI EEPROM Polled Mode Example Test \r\n");
97
98   // init gpio
99   ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
100
101   Status = XGpioPs_CfgInitialize(&Gpio, ConfigPtr,
102                   ConfigPtr->BaseAddr);
103   if (Status != XST_SUCCESS) {
104     return XST_FAILURE;
105   }
106   //cs
107   XGpioPs_SetDirectionPin(&Gpio, CS_PIN, 1);
108   XGpioPs_SetOutputEnablePin(&Gpio, CS_PIN, 1);
109   XGpioPs_WritePin(&Gpio, CS_PIN, 1);
110   //clk
111   XGpioPs_SetDirectionPin(&Gpio, CLK_PIN, 1);
112   XGpioPs_SetOutputEnablePin(&Gpio, CLK_PIN, 1);
113
114
115   /*
```

```
116     * Run the Spi Interrupt example.
117     */
118
119   Status = StartSPI(&SpiInstance, XPAR_XSPIPS_0_BASEADDR);
120   if (Status != XST_SUCCESS) {
121     xil_printf("SPI EEPROM Polled Mode Example Test Failed\r\n");
122     return XST_FAILURE;
123   }
124   u8 buf = 100;
125   //u32 debug_read = *(volatile u32*)(XPAR_ADC_READER_CORE_0_HIGHADDR - 3);
126   print_ADC_memory();
127     int loop_nr = 1;
128     u32 time_sum = 0;
129   while(1){
130         auto_search(50, 50, 0);
131         usleep(10);
132   }
133
134   xil_printf("Successfully ran SPI EEPROM Polled Mode Example Test\r\n");
135   return XST_SUCCESS;
136 }
137
138 void DAC_write(u8 value){
139   //data starts at cs low and is executed at cs high
140   // 00110000xxxxxxxx00000000cc
141   u8 data[3];
142   data[0] = 0 | 0b0011 << 4;
143   data[1] = 0 | value;
144   data[2] = 0;
145   XGpioPs_WritePin(&Gpio, CS_PIN, 0);
146   XSpiPs_PolledTransfer(&SpiInstance, &data, NULL, 3);
147   XGpioPs_WritePin(&Gpio, CS_PIN, 1);
148 }
149
150 void launch_pulse(){
151   XGpioPs_WritePin(&Gpio, CLK_PIN, 1);
152   //usleep(GENERAL_DELAY);
153   XGpioPs_WritePin(&Gpio, CLK_PIN, 0);
154   usleep(5);
155 }
156
157 void print_ADC_memory(){
158   xil_printf("debug0: %d\r\n", *(volatile u32*)(
      XPAR_ADC_READER_CORE_0_BASEADDR));
159   xil_printf("debug1: %d\r\n", *(volatile u32*)(
      XPAR_ADC_READER_CORE_0_BASEADDR + 4));
160   xil_printf("debug2: %d\r\n", *(volatile u32*)(
      XPAR_ADC_READER_CORE_0_BASEADDR + 8));
161     if((*(volatile u32*)(XPAR_ADC_READER_CORE_0_BASEADDR + 8) & 6) == 6){
162         xil_printf("reading\r\n");
163     }
164     else if ((*(volatile u32*)(XPAR_ADC_READER_CORE_0_BASEADDR + 8) & 6) == 4)
      {
165         xil_printf("expecting\r\n");
166     }
```

```
    else if ((*(volatile u32*)(XPAR_ADC_READER_CORE_0_BASEADDR + 8) & 6) == 2)
    {
        xil_printf("idle********\r\n");
    }
  xil_printf("debug3: %X\r\n", *(volatile u32*)(
   XPAR_ADC_READER_CORE_0_BASEADDR + 12));
    xil_printf("------------------------------\r\n");
}

//https://www.geeksforgeeks.org/c/c-program-to-sort-an-array-in-ascending-order/
void bubbleSort(int arr[], int n) {
   for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void mySort(int arr[], int n) {
    if (n <= 1) return;

    #define INSERTION_THRESHOLD 20
    #define MAX_STACK_SIZE 300

    int stack[MAX_STACK_SIZE];
    int top = -1;

    stack[++top] = 0;
    stack[++top] = n - 1;

    while (top >= 0) {
        int high = stack[top--];
        int low = stack[top--];
        int size = high - low + 1;

        // Use Insertion Sort for small subarrays
        if (size <= INSERTION_THRESHOLD) {
            for (int i = low + 1; i <= high; i++) {
                int key = arr[i];
                int j = i - 1;
                while (j >= low && arr[j] > key) {
                    arr[j + 1] = arr[j];
                    j--;
                }
                arr[j + 1] = key;
            }
            continue;
        }

        // Median-of-Three pivot selection
```

```
        int mid = low + (high - low) / 2;
        int pivotIndex;
        if ((arr[low] <= arr[mid] && arr[mid] <= arr[high]) ||
            (arr[high] <= arr[mid] && arr[mid] <= arr[low])) {
            pivotIndex = mid;
        } else if ((arr[mid] <= arr[low] && arr[low] <= arr[high]) ||
                   (arr[high] <= arr[low] && arr[low] <= arr[mid])) {
            pivotIndex = low;
        } else {
            pivotIndex = high;
        }

        // Swap pivot to end
        int temp = arr[pivotIndex];
        arr[pivotIndex] = arr[high];
        arr[high] = temp;

        // Lomuto partition
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (arr[j] <= pivot) {
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        // Place pivot in correct position
        int temp2 = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp2;
        int p = i + 1;

        // Push subarrays onto stack
        if (p - 1 > low) {
            stack[++top] = low;
            stack[++top] = p - 1;
        }
        if (p + 1 < high) {
            stack[++top] = p + 1;
            stack[++top] = high;
        }
    }

    #undef INSERTION_THRESHOLD
    #undef MAX_STACK_SIZE
}

int auto_search(int amp, int power, int depth){
    if(power <= 1){
        power = 1;
        depth++;
    }
    if(depth == 30){
```

```
274         //auto_search(50, 50, 0);
275         return 0;
276     }
277     //xil_printf("%d\r\n", amp);
278     DAC_write(amp);
279     float read;
280     if(power > 1){
281         read = array_read(10);
282     }
283     else {
284         read = array_read(ARRAY_SIZE);
285     }
286     if(read > 190 && read < 210 && power == 1){
287         xil_printf("read: %d, power: %d\r\n", (int) read, amp);
288     }
289     else if(read > 200){
290         auto_search(amp - power, power/2, depth);
291     }
292     else {
293         auto_search(amp + power, power/2, depth);
294     }
295     return 0;
296 }
297
298 float array_read(int size){
299     float read = 0;
300     float array[size];
301     for(int i = 0; i < size; i++){
302         array[i] = 0;
303     }
304     for(int i = 0; i < size; i++){
305         launch_pulse();
306         array[i] = *(volatile u32*)(XPAR_ADC_READER_CORE_0_BASEADDR);
307     }
308     mySort(array,size);
309     for(int i = 0; i < size/3; i++){
310         read += array[(size/3) + 1];
311     }
312     return read/(size/3.0);
313 }
314 /****************************************************************************
    */
315 /**
316  *
317  * The purpose of this function is to illustrate how to use the XSpiPs
318  * device driver in polled mode. This test writes and reads data from a
319  * serial EEPROM. The serial EEPROM part must be present in the hardware
320  * to use this example.
321  *
322  * @param SpiInstancePtr is a pointer to the Spi Instance.
323  * @param SpiDeviceId is the Device Id of Spi.
324  *
325  * @return  XST_SUCCESS if successful else XST_FAILURE.
326  *
327  * @note
```

```
 *
 * This function calls functions which contain loops that may be infinite
 * if interrupts are not working such that it may not return. If the device
 * slave select is not correct and the device is not responding on bus it will
 * read a status of 0xFF for the status register as the bus is pulled up.
 *
 ************************************************************************
   */

int StartSPI(XSpiPs *SpiInstancePtr, UINTPTR BaseAddress)
{
  int Status;
  XSpiPs_Config *SpiConfig;

  /*
   * Initialize the SPI driver so that it's ready to use
   */
  SpiConfig = XSpiPs_LookupConfig(BaseAddress);

  if (NULL == SpiConfig) {
    return XST_FAILURE;
  }

  Status = XSpiPs_CfgInitialize(SpiInstancePtr, SpiConfig,
                SpiConfig->BaseAddress);
  if (Status != XST_SUCCESS) {
    return XST_FAILURE;
  }

  /*
   * Perform a self-test to check hardware build
   */
  Status = XSpiPs_SelfTest(SpiInstancePtr);
  if (Status != XST_SUCCESS) {
    return XST_FAILURE;
  }

  /*
   * Set the Spi device as a master. External loopback is required.
   */
  XSpiPs_SetOptions(SpiInstancePtr, XSPIPS_MASTER_OPTION |
  XSPIPS_FORCE_SSELECT_OPTION);

  XSpiPs_SetClkPrescaler(SpiInstancePtr, XSPIPS_CLK_PRESCALE_64);


  /*
   * Assert the VGA chip select
   */
  XSpiPs_SetSlaveSelect(SpiInstancePtr, VGA_SPI_SELECT);


  return XST_SUCCESS;
}
```

Listing A.2: MCU code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity ADC_reader is
  Port (
    adc_in  :   in  std_logic_vector(7 downto 0);
    clk        :   in std_logic;
    pulse   :   in std_logic;
    reset   :    in std_logic;

    encode :   out std_logic;
    amplitude    : out std_logic_vector(31 downto 0);
    TOF :   out std_logic_vector(31 downto 0);
    read_ok :  out  std_logic

  );
end ADC_reader;

architecture Behavioral of ADC_reader is


    -- state machine
    type state_machine_type is (idle, expecting, reading);
    signal state_machine : state_machine_type := idle;

    signal value_read   :   std_logic_vector(7 downto 0) ;
    signal highest_val  :   integer;

    signal counter : integer;
    signal tof_int  :   integer;

    signal pulse_sent   :   std_logic;
    signal ADC_non_zero :   std_logic;
    signal time_out :   integer;

    signal reset_done   :   std_logic;

    --type lag_array_type is array (0 to 2) of std_logic_vector(31 downto 0);
    --signal lag_array : lag_array_type;
    --signal lag_array_index  :   integer;
begin

    -- conversions for output
    TOF <= std_logic_vector(to_unsigned(tof_int - 4, 32));
    amplitude <= std_logic_vector(to_unsigned(highest_val, 32));
    encode <= clk;
```

```
56
57
58      -- state machine
59      STATE_PROC: process(clk, reset)
60      begin
61          if reset = '1' then
62              state_machine <= idle;
63              counter <= 0;
64              read_ok <= '0';
65              tof_int <= 4;
66          elsif rising_edge(clk) then
67              -- change states on conditions
68              if ADC_non_zero = '1' then
69                  state_machine <= reading;
70              elsif pulse_sent = '1' then
71                  state_machine <= expecting;
72              else
73                  state_machine <= idle;
74              end if;
75              -- state specific condition, usually resets for continuity
76              case state_machine is
77                  when idle =>
78                      counter <= 0;
79                      read_ok <= '1';
80                  when expecting =>
81                      tof_int <= 0;
82                      counter <= counter + 1;
83                      read_ok <= '0';
84                  when reading =>
85                      tof_int <= counter;
86                      read_ok <= '0';
87              end case;
88          end if;
89      end process;
90
91      -- detect pulse & latch
92      PULSE_LATCH_PROC: process(pulse, reset, clk)
93      begin
94          if reset = '1' then
95              pulse_sent <= '0';
96          elsif rising_edge(pulse) then
97              pulse_sent <= '1';
98          elsif rising_edge(clk) then
99              if state_machine = reading then
100                 pulse_sent <= '0';
101             end if;
102         end if;
103     end process;
104
105     -- convert to read ADC
106     CONVERT_PROC: process(clk, reset)
107     begin
108         if reset = '1' then
109             value_read <= "10000000";
110             highest_val <= 0;
```

```
            ADC_non_zero <= '0';
            time_out <= 0;
            reset_done <= '0';
            --lag_array_index <= 0;
            --lag_array(0) <= "00000000000000000000000000000000";
            --lag_array(1) <= "00000000000000000000000000000000";
            --lag_array(2) <= "00000000000000000000000000000000";
        elsif falling_edge(clk) then
            -- read adc value
            value_read <= adc_in;
            -- latch reading if needed
            if to_integer(unsigned(value_read))  > 130 or to_integer(unsigned(
    value_read))  < 126 then
                ADC_non_zero <= '1';
                time_out <= 0;
            elsif state_machine = reading then
                if time_out = 51 then
                    ADC_non_zero <= '0';
                else
                    time_out <= time_out + 1;
                end if;
            end if;
            -----------------------------------
            -- Reset highest_val only once when entering expecting state
            if state_machine = expecting then
                if  reset_done = '0' then
                    highest_val <= 0;
                    reset_done <= '1';
                end if;
            else
                reset_done <= '0';
            end if;
            -- check if max and renew if needed only in reading state
            if state_machine = reading then
                if to_integer(unsigned(value_read)) > highest_val then
                    highest_val <= to_integer(unsigned(value_read));
                end if;
            end if;
            -----------------------------------
            -- make lag array and increment
--            lag_array(lag_array_index) <= adc_in;
--            lag_array_index <= lag_array_index + 1;
--            if lag_array_index = 3 then
--                lag_array_index <= 0;
--            end if;
        end if;
     end process;


end Behavioral;
```

Listing A.3: FPGA code that reads from the ADC and outputs it as registers

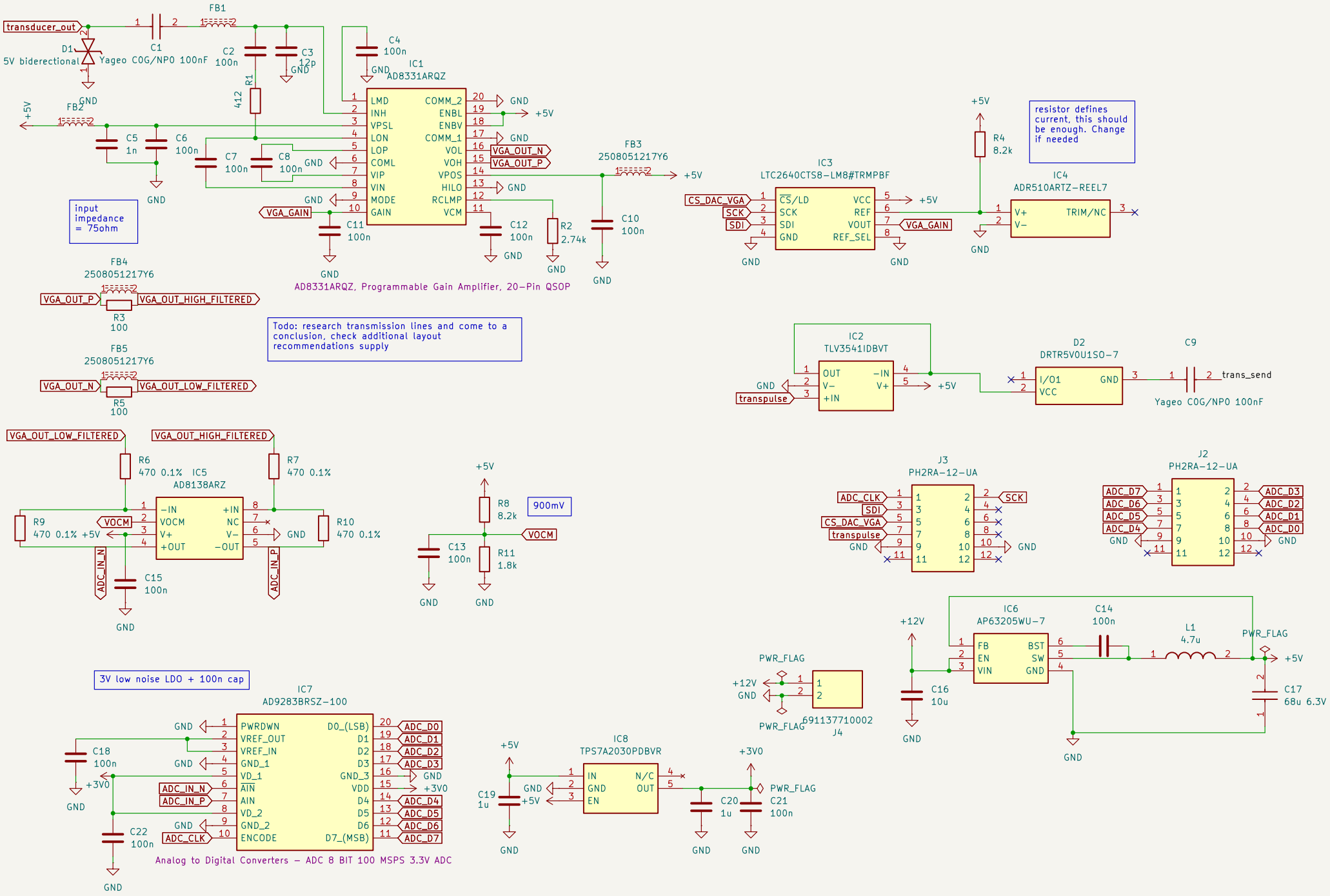
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

```

```
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity pulser is
  Port (
                clk     :   in  std_logic;
                fire    :   in std_logic;
                pulse :   out std_logic
            );
end pulser;

architecture Behavioral of pulser is
  signal fire_sync  : std_logic := '0';   -- Synchronized fire
  signal fire_prev  : std_logic := '0';   -- Previous value for edge detection
  signal count_en   : std_logic := '0';   -- Enable counting
  signal counter    : integer range 0 to 10 := 10; -- 10-cycle counter
begin

  -- Generate pulse: clock gated when count_en is active
  pulse <= clk when count_en = '1' else '0';

  -- Synchronize fire and control counting
  SYNC_PROC: process(clk)
  begin
    if rising_edge(clk) then
      -- Synchronize fire to clock domain
      fire_sync <= fire;
      fire_prev <= fire_sync;

      -- Detect rising edge of fire (and not already counting)
      if (fire_prev = '0' and fire_sync = '1') and count_en = '0' then
        count_en <= '1';  -- Start counting
        counter  <= 0;    -- Reset counter
      end if;

      -- Handle counting
      if count_en = '1' then
        if counter < 9 then  -- Count 0 to 9 (10 cycles)
          counter <= counter + 1;
        else
          count_en <= '0';   -- Stop after 10 cycles
          counter  <= 10;    -- Reset to idle state
        end if;
      end if;
    end if;
  end process;
end Behavioral;
```

Listing A.4: FPGA code creates a 10 pulse pulse train

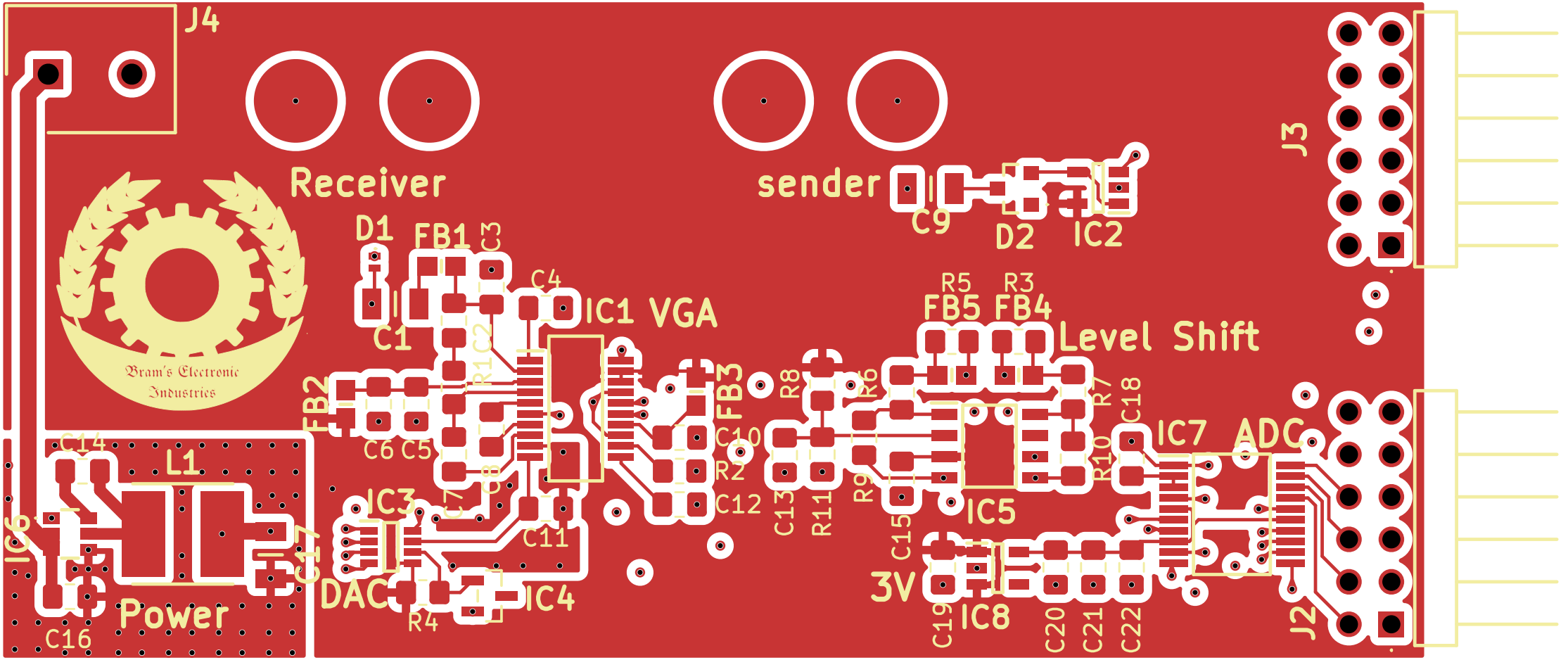
Appendix B

Attachment - Electronic Schematic


transducer_out
D1
5V biderectional
C1
FB1
C2
100n
C3
12p
C4
100n
IC1
AD8331ARQZ
R1
412
FB2
+5V
C5
1n
C6
100n
C7
100n
C8
100n
GND
LMD
INH
VPSL
LON
LOP
COML
VIP
VIN
MODE
GAIN
COMM_2
ENBL
ENBV
COMM_1
VOL
VOH
VPOS
HILO
RCLMP
VCM
VGA_OUT_N
VGA_OUT_P
VGA_GAIN
C11
100n
C12
100n
R2
2.74k
C10
100n
FB3
2508051217Y6
input impedance = 75ohm
AD8331ARQZ, Programmable Gain Amplifier, 20-Pin QSOP
IC3
LTC2640CTS8-LM8#TRMPBF
CS_DAC_VGA
SCK
SDI
CS/LD
SCK
SDI
GND
VCC
REF
VOUT
REF_SEL
R4
8.2k
resistor defines current, this should be enough. Change if needed
IC4
ADR510ARTZ-REEL7
V+
V-
TRIM/NC
FB4
2508051217Y6
VGA_OUT_P
VGA_OUT_HIGH_FILTERED
R3
100
FB5
2508051217Y6
VGA_OUT_N
VGA_OUT_LOW_FILTERED
R5
100
Todo: research transmission lines and come to a conclusion, check additional layout recommendations supply
IC2
TLV3541IDBVT
OUT
V-
+IN
-IN
V+
transpulse
D2
DRTR5V0U1SO-7
I/O1
VCC
C9
trans_send
Yageo C0G/NP0 100nF
VGA_OUT_LOW_FILTERED
VGA_OUT_HIGH_FILTERED
R6
470 0.1%
IC5
AD8138ARZ
R7
470 0.1%
R9
470 0.1%
R10
470 0.1%
VOCM
-IN
VOCM
V+
+OUT
+IN
NC
V-
-OUT
ADC_IN_N
ADC_IN_P
C15
100n
R8
8.2k
900mV
C13
100n
R11
1.8k
J3
PH2RA-12-UA
ADC_CLK
SDI
CS_DAC_VGA
transpulse
SCK
J2
PH2RA-12-UA
ADC_D7
ADC_D6
ADC_D5
ADC_D4
ADC_D3
ADC_D2
ADC_D1
ADC_D0
3V low noise LDO + 100n cap
IC7
AD9283BRSZ-100
PWRDWN
VREF_OUT
VREF_IN
GND_1
VD_1
AIN
AIN
VD_2
GND_2
ENCODE
D0_(LSB)
D1
D2
D3
GND_3
VDD
D4
D5
D6
D7_(MSB)
C18
100n
C22
100n
+3V0
Analog to Digital Converters – ADC 8 BIT 100 MSPS 3.3V ADC
PWR_FLAG
691137710002
J4
IC8
TPS7A2030PDBVR
IN
GND
EN
N/C
OUT
C19
1u
C20
1u
C21
100n
IC6
AP63205WU-7
FB
EN
VIN
BST
SW
GND
C14
100n
L1
4.7u
C16
10u
C17
68u 6.3V
+12V

Appendix C

Attachment - PCB drawing


J4
Receiver
sender
J3
D1
FB1
C3
C4
C1
IC1 VGA
C9
D2
IC2
R5
R3
FB5
FB4
Level Shift
FB2
C6
C5
R1
C2
FB3
R8
R6
C10
R2
C12
C13
R11
R9
C15
IC5
R7
C18
IC7 ADC
R10
C14
L1
IC6
C17
IC3
C7
C8
C11
DAC
IC4
R4
3V
IC8
C19
C20
C21
C22
J2
Power
C16
Bram's Electronic
Industries