

Portable Ultrasound System for Blood Velocity Estimation
Project Report

Master Thesis
April, 2023

Author:
Jeppe Hinrichs
Advisor(s):
Hyunjoo Lee, Xenofon Fafoutis, Tiberiu Gabriel Zsurzsan

Copyright:	Reproduction of this publication in whole, or in part, must include the customary bibliographic citation, including author attribution, report title, etc.
Cover photo:	RawPixel, 2022
Published by (1):	Technical University of Denmark, DTU Elektro, Institut for Elektroteknologi, Ørsteds Plads, Building 348, 2800 Kgs. Lyngby, Denmark www.elektro.dtu.dk
Published by (2):	Korea Advanced Institute of Science and Technology, School of Electrical Engineering, E3-2, Daehak-ro 291, 34141 Daejeon, Republic of Korea www.ee.kaist.ac.kr
Timespan:	Saturday 1 st October, 2022 ~ Wednesday 12 th April, 2023
Credits:	30 ECTS / 9 credits
Degree:	Master of Science
Field:	Electrical Engineering

Approval

This thesis is submitted to the faculty of Korea Advanced Institute of Science and Technology and the Technical University of Denmark in partial fulfillment of the requirements for the double degree of Master of Science in Electrical Engineering (M.S.E.E).

The study was conducted in accordance with the Code of Research Ethics¹.

Daejeon, Republic of South Korea
Jeppe Hinrichs - s163555 / 20214926

.....
Signature

.....
Date

¹Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology and the Technical University of Denmark, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my dissertation/thesis contains honest conclusions based on my own careful research under the guidance of my advisor(s).

Acknowledgements

I would like to express my sincere gratitude to all those who have contributed to the completion of this thesis.

Firstly, I would like to thank my advisors, Lee Hyunjoo, Fontas, and Gabriel, for their guidance and support throughout the entire research process. Their valuable feedback and constructive criticism have been instrumental in shaping my ideas and improving the quality of my work.

Secondly, I would like to show my profound gratitude to Erick and Hyoga, with whom I shared the EE office, for graciously helping me catch up when I was overwhelmed with coursework. They dedicated their time to help me get up to speed, and I am truly thankful.

I would also like to extend my gratitude to all the members of Brain/Bio Medical Microsystems Laboratory, who have inspired me during my academic journey. Their kindness creates a positive atmosphere and their profound knowledge have provided me with a solid foundation for conducting research.

I am deeply grateful to my family and friends back home for their unwavering support and encouragement, both academically and personally. Their love, care, and motivation have kept me going during the challenging times of this project.

Finally, I would like to acknowledge the collaborative experiments done with Sangmok and Taemin, whose willingness to take part in this research has made it possible for me to generate meaningful findings. Their insights and experiences have been invaluable in helping me to answer my research questions.

Contents

Copyright

Approval

Acknowledgements

Contents I

List of Figures III

List of Tables V

List of Source Codes VI

Acronyms VII

Glossary IX

1 Introduction 1

 1.1 Literature review 1
 1.2 Project scope 3

2 Theory 5

 2.1 Ultrasound 5
 2.1.1 Scattering 6
 2.1.2 Attenuation 7
 2.1.3 Transducer 8
 2.1.4 Doppler effect 8
 2.2 Flow Physics 10
 2.2.1 Blood flow 12
 2.3 Devices 14
 2.3.1 Continuous-wave Flowmeter 14
 2.3.2 Pulsed-wave Flowmeter 16
 2.4 Blood Velocity Estimation 18
 2.4.1 Spectral Envelope 18
 2.4.2 One-Dimensional Velocity Estimation 21
 2.4.3 Sonography 22

3 Synthesis 23

 3.1 Control System 23
 3.2 Pulse Generator 25
 3.3 Power Stage 25
 3.4 Transmit/Receive Switch 28
 3.5 Band-pass Filter 30
 3.6 Preamplifier 32
 3.7 Quadrature Demodulator 32
 3.8 Sample and Hold 35

3.9	Active Filter, DC-Coupling	35
3.10	Digital Signal Processor	37
4	Implementation	39
4.1	Control System	39
4.2	Power Stage	41
4.3	Transmit/Receive Switch	45
4.4	Band-pass Filter	45
4.5	Preamplifier	45
4.6	Quadrature Demodulator	45
4.7	Sample and Hold	47
4.8	Active Filter, DC-Coupler	47
4.9	Digital Signal Processor	49
5	Analysis	51
5.1	Validation experiments	51
5.1.1	Control System	51
5.1.2	Power Stage	53
5.1.3	Transmit/Receive Switch	53
5.1.4	Band-pass Filter	53
5.1.5	Preamplifier	55
5.1.6	Demodulator	55
5.1.7	Sample and Hold Amplifier	55
5.1.8	Active Filter, DC Coupler	58
5.2	Pulse Generator and Power Stage	58
5.3	Doppler String Phantom Experiment	58
6	Conclusion	61
6.1	Future Work	61
	Bibliography	63
A	Source Code	68
A.1	Microcontroller Code	68
A.2	Field Programmable Gate Array Code	69
A.3	MATLAB Scripts	76
B	Simulation Models	90
C	Circuit Schematics	92
D	Circuit CAD Assembly Documentation	104
E	Instruments	106

List of Figures

2.1	Particle displacement for a propagating ultrasound wave	5
2.2	Single element ultrasound transducer construction	8
2.3	Transducer types for acquiring B-mode images	9
2.4	Doppler effect diagram	9
2.5	Circulatory system of the human body	11
2.6	Diagram of ultrasound wave transmitted and reaching blood vessel with incident angle θ	13
2.7	Block diagram of continuous-wave flowmeter	14
2.8	Demodulation effects of Doppler signals in time and frequency domain . . .	15
2.9	Block diagram of pulsed-wave flowmeter	16
2.10	Sampling for a gate pulsed wave system with a single range	17
2.11	Simulated RF sampling of a blood vessel	19
2.12	Stationary tissue sampled signal and stationary tissue sample spectrum . .	19
2.13	Single moving scatterer crossing a concave transducer beam	20
2.14	Frequency axis scaling for scatterer with velocity v_z	21
2.15	Arterial sonogram with time-frequency and Doppler shift	22
3.1	Simplified overview of the entire system	23
3.2	XNUCLEO-F411RE development board by Waveshare	24
3.3	Timing diagram of various control signals for an arbitrary n length pulse chain expressed by the second diagram gap	26
3.4	Block diagram of power stage	26
3.5	LTspice simulation output of transmitter	27
3.6	Block diagram of TX/RX switching circuit	28
3.7	Timing diagram of switching interface	29
3.8	Recovery time from transmitting to receiving state with an AC coupled high-voltage pulse and TX810	29
3.9	3D Render of PCB in Altium Designer	30
3.10	Band-Pass Filter insertion loss and specifications	31
3.11	Bode plot of bandpass filter	31
3.12	Block diagram of preamplifier AD8332	32
3.13	LTspice simulation output of preamplifier	33
3.14	Block diagram of demodulator AD8333	33
3.15	LTspice simulation demodulator input variables	34
3.16	LTspice simulation demodulator output variables	34
3.17	AD783 Sample and Hold amplifier block diagram	35
3.18	Sample and Hold function with input function $f(t)$ over time	36
3.20	Small-signal analysis of DC-Coupling filter circuit	37
3.21	Transient analysis of DC-Coupling filter circuit	38
4.1	STM32 Zephyr RTOS pulser output	39
4.2	PYNQ-Z1 development board	40
4.3	PYNQ Z1 PMOD port diagram	40
4.4	finite state machine (<i>FSM</i>) diagram of ultrasound pulser signal controller .	42
4.5	PWM pulser and signal controller as a block diagram with inputs and outputs	42

4.6	Top level block diagram of the FPGA overlay with AXI interconnects and registers	43
4.7	MD1213DB1 High Speed Pulser	43
4.8	Transmit/Receive Switch after assembly	45
4.9	Prototype board of the Band-Pass filter	46
4.10	Demodulator PCB AD8333-EVALZ	46
4.11	Prototype board of the Sample and Hold amplifier	47
4.12	Prototype board of the active filter and direct current (<i>DC</i>)-coupler	48
5.1	Complementary PWM output with Pynq Z1 FPGA and JupyterLab notebook	51
5.2	Captured timing diagram of the control system pulser	52
5.3	Measured input and output of power stage PCB	53
5.4	TX/RX Switch reflection experiment with water tank	54
5.5	Measured transmit and receive on Transmit/Receive Switch PCB	54
5.6	Band-pass filter bode plot	55
5.7	Measured input and output of preamplifier PCB	56
5.8	Measured input of demodulator PCB	56
5.9	Measured output of demodulator PCB	57
5.10	Measured input and output of Sample and Hold amplifier	57
5.11	Measured input and output of Active filter and DC Coupler	58
5.12	Complementary PWM output from the pulse generator and bipolar high power pulses	59
5.13	CIRS Model 043A Doppler String Phantom	60
5.14	Doppler String Phantom experiment diagram (not to scale)	60
A.1	Jupyter Notebook running on PYNQ Z1	69
A.2	ZYNQ Top Level block diagram	70
B.1	LTspice model of transmitter	90
B.2	LTspice model of preamplifier	90
B.3	LTspice model of demodulator	91
B.4	LTspice model of PRF filter	91
C.1	AFE Top Level	93
C.2	AFE Power Stage	94
C.3	AFE BPF	95
C.4	AFE Preamplifier	96
C.5	AFE Demodulator	97
C.6	AFE Sample and Hold Amplifier	98
C.7	AFE Wall Filter	99
C.8	UltrasoundSwitch Schematic A	100
C.9	UltrasoundSwitch Schematic B	101
C.10	MD1213DB1 Transmitter Schematic	102
C.11	AD8332 Preamplifier, AD8333 IQ Demodulator Schematic	103
D.1	UltrasoundSwitch Assembly Information	105

List of Tables

1.1	Comparison of medical imaging modalities	2
1.2	Comparison of papers in literature study	3
1.3	Project specification	4
2.1	Approximate density, sound speed, and acoustic impedance of human tissue types	6
2.2	Approximate attenuation values for human tissue	7
2.3	Typical dimensions and flow in human circulatory system	13
2.4	Measured frequency shifts with a Doppler 3 MHz transducer at various velocities at a 45° incident angle	16
E.1	List of instruments used for solder work	106
E.2	List of instruments used in experiments	106

List of Source Codes

4.1	Constraints on Pulse Generator and Control System	41
4.2	PWM Pulse Generator Descriptor	44
A.1	main.c	68
A.2	Pwm.vhdl	71
A.3	signalcontroller.vhdl	73
A.4	inv_top.vhdl	76
A.5	US Simulation Master Script.m	76
A.6	Analysis.m	81
A.7	demodulator.m	84
A.8	preamplifier.m	86
A.9	sampler.m	86
A.10	transmitter.m	87
A.11	vna.m	87
A.12	bpf datasheet s21.m	88
A.13	switch.m	89

Acronyms

Notation	Description
<i>AC</i>	alternating current
<i>ADC</i>	analogue-to-digital converter
<i>AFE</i>	analogue front end
<i>ASIC</i>	application-specific integrated circuit
<i>BP</i>	band-pass
<i>CMUT</i>	capacitive micromachined ultrasound transducer
<i>CPU</i>	central processing unit
<i>CT</i>	computed tomography
<i>CVD</i>	cardiovascular diseases
<i>CW</i>	continuous-wave
<i>DC</i>	direct current
<i>DCPS</i>	<i>DC</i> power supply
<i>DSP</i>	digital signal processor
<i>DTU</i>	Danmarks Tekniske Universitet (Technical University of Denmark)
<i>FET</i>	field-effect transistor
<i>FFT</i>	fast Fourier transform
<i>FPGA</i>	field-programmable gate array
<i>FSM</i>	finite state machine
<i>GPIO</i>	general purpose input/output (<i>I/O</i>)
<i>HAL</i>	hardware abstraction layer
<i>HP</i>	high-pass
<i>I/O</i>	input/output
<i>IC</i>	integrated circuit
<i>KAIST</i>	Korea Advanced Institute of Science and Technology
<i>LNA</i>	low noise amplifier
<i>low-res</i>	low resolution
<i>LP</i>	low-pass
<i>LSB</i>	least significant bit
<i>MCU</i>	microcontroller unit
<i>MOSFET</i>	metal-oxide-semiconductor field-effect transistor
<i>MRI</i>	magnetic resonance imaging

Notation	Description
PRF	pulse repetition frequency
PSD	power spectral density
PW	pulsed-wave
PWM	pulse-width modulation
PZT	lead circonate titanate
$RTOS$	real-time operating system
SoC	system-on-a-chip
US	ultrasound
VGA	variable gain amplifier
VNA	vector network analyzer

Glossary

Notation	Description
adiabatic	Any process that happens without heat gain or loss is considered adiabatic
bitstream	A bitstream is a file that is created to setup an FPGA. The bitstream describes the hardware logic, routing, and starting values for registers as well as on-chip memory (e.g., LUT). A bitstream, like the ELF format, has its own format for describing its contents
Doppler effect	A change in frequency of a wave in relation to an observer who is moving relative to the wave source
HIGH	Logic high voltage level
LOW	Logic low voltage level
monochromatic wave	A wave which has only a single frequency
open-source	Refers to software whose original source code is publicly available and may be changed and distributed
SPICE	SPICE ("Simulation Program with Integrated Circuit Emphasis") is an open-source IC and board-level circuit simulator
transcutaneous	Applied across the depth of the skin without invasive penetration

1 Introduction

The progress of diagnostic imaging has advanced significantly during the 20th century. As the cost of high-speed computational systems has grown increasingly accessible, so has the use of medical imaging become prominent. Millions of people have potentially been spared painful exploratory surgery through non-invasive diagnostic imaging. Thus, lives can be saved by early diagnosis and intervention through medical imaging. Advancements in scientific visualisation have in turn generated more complex data-sets of increased size and quality. The four major technologies used are ultrasound (*US*), X-ray, computed tomography (*CT*), and magnetic resonance imaging (*MRI*). Each technology has distinct advantages and disadvantages in biomedical imaging, and thus each is still relevant for modern medicine. Table 1.1 contains a comparison and summary of the various fundamental diagnostic imaging modalities.

Between 2004 and 2016, medical imaging has been reported to have been performed more than 5 billion times [16]. Later numbers from 2011 show a general doubling and in particular, a tenfold increase in ultrasound examinations between 2000 and 2011 [38]. Recent data reveal that this trend of doubling has continued throughout the years 2010 to 2020 [56], and reveal that even though patient processes were disrupted during the global SARS-CoV-2 pandemic, the number of medical imaging examinations per 1000 patients still increased. The reasons for this and, particularly, why ultrasound has seen a significant increase in use, can be attributed to its high resolution, cost-effectiveness, portability, and real-time interventional imaging. The downside of ultrasound is its limited penetration, restrictions for use in certain body parts, and inconsistent resolution. When comparing soft tissue examinations, which ultrasound is limited to, both *CT* and *MRI* can image the entire body with consistent resolution and contrast, but are more expensive and have poor portability due to the immense size of their hardware.

Fix end year

The cardiovascular system, which transports oxygen and nutrients to tissue, produces a complex flow pattern that causes velocity fluctuations. Several cardiovascular diseases (*CVD*) are also known to cause abnormal blood flow. In studies published by the Centers for Disease Control, a person dies from CVD every 34 seconds in the United States and complications from CVD cost 229 billion USD between 2017 and 2018 [57]. As mentioned above, ultrasound is a powerful tool for performing non-invasive imaging of the cardiovascular system [26], [32], and has no adverse risk to patients. Determining power spectral density (*PSD*) of a received signal is a common way to estimate blood velocity. A processed image of *PSD* over time is commonly known as a sonogram, where changes in blood velocity over time can be seen.

Check over repetitive

1.1 Literature review

A systematic review was conducted using PubMed, Google Scholar, Elsevier, DTU FindIt and IEEE Xplore with the search terms “pulsed-wave Doppler ultrasound”, “blood velocity estimation”, and “ultrasound flow-meter”. The search was limited to English-language articles. The literature search yielded more than 50 papers, of which 37 were studied for the purpose of learning from the contents [1]–[4], [6]–[14], [17], [19], [21]–[24], [27]–[31], [35], [37], [40], [43], [44], [47], [49]–[51], [54], [56], [58], [61]. In addition, textbooks [32], [38], [39] were used in the preparation and study of the theoretical principles of biomedical imaging and ultrasound.

Table 1.1: Comparison of medical imaging modalities [38]

Modality	Ultrasound	X-ray	CT	MRI
Topic	Longitudinal, shear, mechanical properties	Mean X-ray tissue absorption	Local tissue X-ray absorbtion	Biochemistry (T_1 and T_2)
Access	Small windows adequate	2 sides needed	Circumferential around body	Circumferential around body
Spatial resolution	0.2 mm to 3 mm ^a	~ 1 mm	~ 1 mm	~ 1 mm
Penetration	3 cm to 25 cm ^b	Excellent	Excellent	Excellent
Safety	Excellent	Ionizing radiation	Ionizing radiation	Very good
Speed	Real-time	Minutes	20 minutes	Varies [†]
Cost	\$	\$	\$\$	\$\$\$
Portability	Excellent	Good	Poor	Poor
Volume coverage	Real-time 3D volumes, improving	2D	Large 3D volume	Large 3D volume
Contrast	Increasing (shear)	Limited	Limited	Slightly flexible
Intervention	Real-time 3D increasing	No ^c	No	Yes, limited
Functional	Functional ultrasound	No	No	fMRI

^a Frequency and axially dependent.^b Frequency dependent.^c Fluoroscopy limited.† Typical: 45 minutes, fastest: Real-time (*low-res*).

Table 1.2: Comparison of papers in literature study

	Huang <i>et al.</i> [30]	Jana <i>et al.</i> [51]	Ding <i>et al.</i> [54]
Architecture	PW 10 MHz	CW 8 MHz	PW 3.7 MHz
Type	Smartphone	Portable device	Computer
Power	12 V	12 V	Details not available
Components	PRF timer, bipolar pulser, quadrature demodulation, SHA	RF amplifier, envelope detection, LP filter, FPGA, preamplifier, ADC	AFG, RF amplifier, quadrature demodulation, SHA, BP filter
DSP	512 pt FFT	512 pt FFT	FFT size not mentioned
Metrics	Doppler spectrogram	Haemodynamic parameters	Doppler spectrogram
Output	Aux microphone signal	Bluetooth	DAQ input signal
Validation	In-vivo animal experiment	ML evaluation on humans	Physiological simulator

Among these works are some of the earliest papers that outlined the field as it was emerging. Other articles study the possibilities of improvements in algorithms and experimental parameters. Overall, the results indicate that the Doppler flow meter is a reliable method for estimating blood flow velocity in various parts of the body. Studies include experiments using physiological simulators and in-vivo on humans and animals alike. Some of the review articles have compared Doppler flow-meters to other imaging techniques for this application, such as magnetic resonance imaging and computed tomography angiography, and have shown that the Doppler flow-meter is a cost-effective, portable and non-invasive choice.

Of the selected papers studied in this project, three papers are distinctly relevant for the design and implementation of a blood velocity estimation system. A comparison between these three papers can be seen in table 1.2. Based on the literature review, a gap is identified in the acquisition method of the signal chain. A number of articles studied and developed the algorithms for blood flow estimation and imaging, but do did have an analogue front end (*AFE*) and used offline data acquisition methods which are not usable for clinicians. A few of the articles feature online data acquisition but use field-programmable gate array (*FPGA*) or application-specific integrated circuit (*ASIC*) technology in the design, which are practical but less cost-effective. This high cost of parts resembles a limitation to its use. To improve the current state of ultrasound Doppler imaging there is potential to decrease cost and retain effectiveness using modern low-cost electronics.

1.2 Project scope

A list of project goals is provided in table 1.3. The project is conducted under the guidance of advisors from the affiliated institutions Danmarks Tekniske Universitet (Technical University of Denmark) (*DTU*), Department of Electrical Engineering, Department of Applied Mathematics and Computer Science, and Korea Advanced Institute of Science and Technology (*KAIST*) at the Brain/Bio Medical Microsystems Laboratory. The report is divided into five chapters, and the first part is an introduction to the project. The second chapter will focus on explaining the theory of the topic of the project. The third chapter focuses on the synthesis of a system model for experimental testing. The fourth chapter explains the method of implementation during the assembly of the system. The fifth chapter will explain the testing methodology performed on the hardware. Finally, additional documentation of testing, code, circuit diagrams, and laboratory setups can be found in the appendix.

Omskriv til
uni

Table 1.3: Project specification

Project specification
Study and research ultrasound and its principles and applications
Design and implement a device for ultrasound blood velocity estimation
Investigate and test the device in an experimental setting
Validate results with commercial equipment
Make quantifiable performance measurements on the system
Write a technical report documenting the project work

2 Theory

This chapter explains the overall theory that forms the fundamental principles of this project. Initially, the characteristics of ultrasound will be explained from an acoustics standpoint. Then, a brief overview of systemic circulation is explained *in vivo*. Lastly, the various types of flow meters are outlined with their strengths and weaknesses.

2.1 Ultrasound

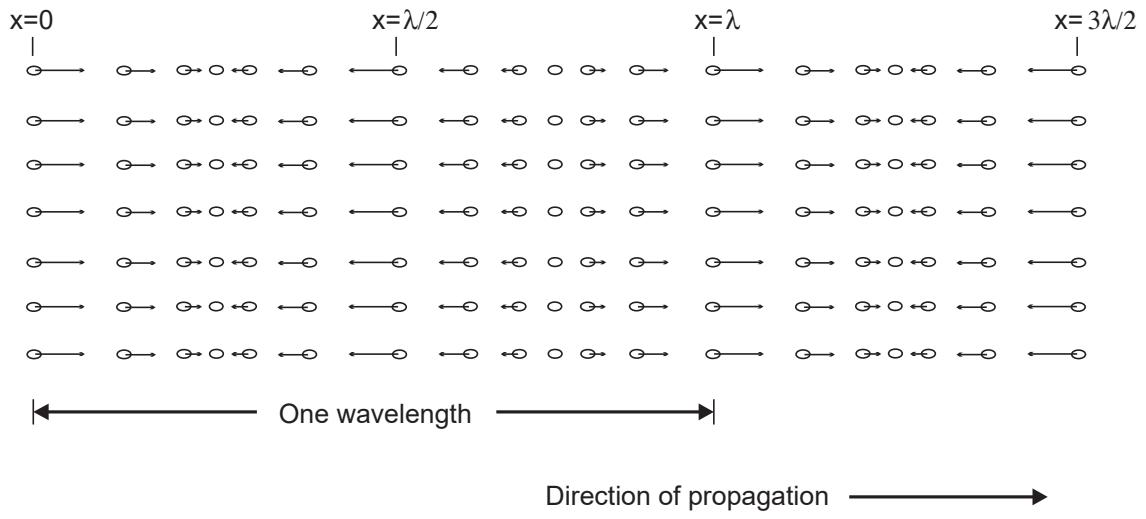


Figure 2.1: Particle displacement for a propagating ultrasound wave [32]

Ultrasound is a technology that transmits sound wave with frequencies above the audible range (20 to 20 000 Hz) to mechanically vibrate matter. The particles in the medium would be at rest and distributed uniformly before any disturbance. The wave propagates as a disturbance and the particles oscillate around their mean position due to the presence of the ultrasonic wave. Typically, the *US* frequency-band used in clinical settings are from 1 to 15 MHz [38]. Certain specialist applications, such as eye, skin, and small animal imaging may use ultrasound frequencies at or above 20 MHz [31]. The trade-off with very high-frequency ultrasound balances increased resolution with reduction in penetration depth. For blood velocity estimation or echocardiography, moderate penetration is more important than resolution. Thus, the frequency used is most often at 10 MHz or less. Figure 2.1 visualizes the propagation of a plane wave in matter. The oscillation occurs parallel to the wave's direction, making it longitudinal, and the disturbance will propagate with the variable c , which is determined by the medium and is given by eq. (2.1).

$$c = \sqrt{\frac{1}{\rho_0 \kappa_S}} \quad (2.1)$$

where ρ_0 is the mean density (kgm^{-3}) and κ_S is the adiabatic compressibility (m^2N^{-1}). Since in the majority of cases, the propagation of ultrasound is linear, the linear propagation is assumed in this work. The acoustic pressure of the harmonic plane wave is expressed by eq. (2.2) and propagates along the z -axis.

$$p(t, z) = p_0 e^{j(\omega t - kz)} \quad (2.2)$$

where ω is the angular frequency, k is the wave number and is expressed by $k = \omega/c = 2\pi/\lambda$, and p_0 is the acoustic pressure amplitude. A spherical wave is expressed by eq. (2.3).

$$p(t, r) = p_0 e^{j(\omega t - kr)} \quad (2.3)$$

where r is radial distance and is defined in a polar coordinate system. For each time instance, the acoustic pressure $p(t, r)$ is constant over a fixed radial position. In this scenario, the pressure amplitude is given by $p_0(r) = k_p/r$, where k_p is a constant since the energy of the outgoing wave must be constant. Particle speed u is dependent on the pressure caused by a wave expressed by eq. (2.4).

$$u = \frac{p}{Z} \quad (2.4)$$

where Z is the characteristic acoustic impedance, defined as the ratio of acoustic pressure to particle speed at a given position in the medium and is expressed by eq. (2.5).

$$Z = \rho_0 c \quad (2.5)$$

Characteristic acoustic impedance Z is one of the most significant variables in the characterization of propagating plane waves. Reference values for density, speed of sound, and characteristic acoustic impedance can be seen in table 2.1.

Table 2.1: Approximate density, sound speed, and acoustic impedance of human tissue types [32]

Medium	Density (ρ_0) kg/m ³	Speed of sound (c) m/s	Acoustic impedance (Z) kg/(m ² s)
Air	1.2	333	0.4×10^3
Blood	1.06×10^3	1566	1.66×10^6
Bone	$1.38\text{--}1.81 \times 10^3$	2070–5350	$3.75\text{--}7.38 \times 10^6$
Brain	1.03×10^3	1505–1612	$1.55\text{--}1.66 \times 10^6$
Fat	0.92×10^3	1446	1.33×10^6
Kidney	1.04×10^3	1567	1.62×10^6
Lung	0.4×10^3	650	0.26×10^6
Liver	1.06×10^3	1566	1.66×10^6
Muscle	1.07×10^3	1542–1626	$1.65\text{--}1.74 \times 10^6$
Spleen	1.06×10^3	1566	1.66×10^6
DI	1×10^3	1480	1.48×10^6

In the following sections, various acoustic wave phenomena will be briefly described.

2.1.1 Scattering

A wave propagating through a medium continues in the same direction until it encounters a new medium. When this occurs, a portion of the wave is transmitted into the new medium with a change in direction. Because the scattered wave is the result of several contributors, it is necessary to define it statistically. The amplitude distribution is Gaussian [32] and can thus be fully described by its mean and variance. The mean value is zero because the dispersed signal is caused by variances in the acoustic characteristics in the tissue. The

correlation between multiple data is what allows ultrasound to determine blood velocities. Because minor movements have a significant correlation, it is feasible to discover alterations in location by comparing sequential measurements of moving structures, such as blood cells. In medical ultrasound, only one transducer is used to transmit and receive, and only the backscattered signal is analysed. The power of the scattered signal is defined by the scattering cross-section, which in small cases means a uniform intensity I_i , and is expressed by eq. (2.6).

$$P_s = I_i \sigma_{sc} \quad (2.6)$$

where σ_{sc} is the scattering cross-section in square meters. The backscattering cross section is material dependant and determines the intensity of the scattering. If the dispersed energy is evenly emitted in all directions, the scattered intensity is given by eq. (2.7).

$$I_s = \frac{P_s}{4\pi R^2} = \frac{\sigma_{sc}}{4\pi R^2} \cdot I_i \quad (2.7)$$

where R is the distance to the scattering region [32]. This results in a spherical wave. A transducer with radius r gives the power P_r , presuming the attenuation and focus are neglected, and is expressed by eq. (2.8).

$$P_r = I_s \pi r^2 = \sigma_{sc} \frac{r^2}{4R^2} \cdot I_i \quad (2.8)$$

The backscattering coefficient, which defines the characteristics from its volume, is another measure of scattering strength. It is defined as the average received power per steradian volume of scatterers when flooded with plane waves of unit amplitude and the unit is 1/cm³s. Back-scattering coefficients in the blood are significantly lower than the backscattering coefficients from various tissue types. This poses a challenge when estimating blood flow close to tissue vessel walls [5], [32].

2.1.2 Attenuation

The ultrasonic wave will be reduced as it propagates through the tissue due to absorption and scattering. The attenuation in tissue is frequency dependent, with greater attenuation with increasing frequency. Because of absorption and dispersion, the ultrasonic wave will be attenuated as it travels through the tissue. The relationship between attenuation, distance travelled, and frequency is often linear. Attenuation in the tissue occurs as a result of both dispersion, which spreads energy in all directions, and absorption, which turns it into thermal energy. A table of approximate attenuation values can be seen in table 2.2.

Table 2.2: Approximate attenuation values for human tissue [32]

Tissue	Attenuation dB/(MHz · cm)
Liver	0.6–0.9
Kidney	0.8–1
Spleen	0.5–1
Fat	1–2
Blood	0.17–0.24
Plasma	0.01
Bone	16–23

Equation (2.9) expresses the exponential decrease of pressure of a wave propagating in z -direction.

$$p(z) = p(z = 0)e^{-\alpha z} \quad (2.9)$$

In eq. (2.9), $p(z = 0)$ is the pressure in the point of origin and α is the attenuation coefficient. The attenuation coefficient unit is Npcm^{-1} and, alternatively, dBcm^{-1} with the relationship described in eq. (2.10).

$$\alpha = \frac{1}{z} \ln \frac{p(z = 0)}{p(z)} \quad (2.10a)$$

$$\alpha(\text{dBcm}^{-1}) = 20(\log_{10} e)\alpha(\text{Npcm}^{-1}) = 8.68\alpha(\text{Npcm}^{-1}) \quad (2.10b)$$

The significance of absorption and scattering in ultrasonic attenuation in biological tissues is a point of contention. Scattering adds just a few per cent to attenuation in most soft tissues. As a result, it is fair to conclude that absorption is the primary mechanism of ultrasonic attenuation in biological tissues [39].

2.1.3 Transducer

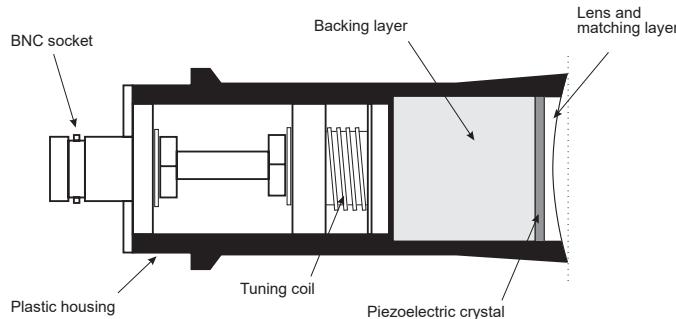


Figure 2.2: Single element ultrasound transducer construction [32]

A layperson knows transducers as speakers and microphones in the context of PA systems. In the case of medical US it is the device that generates the acoustic pressure field, which is emitted into the tissue. A common type of US transducer is a lead circonate titanate (*PZT*) type, which has a piezoelectric crystal inside the housing. When excited, this crystal emits ultrasound waves toward flowing blood. The red blood cells will reflect a fraction of the emitted waves. These reflected waves are of a different frequency than the transmitted wave. If the red blood cells move away from the transducer, the frequency will be lower. If the red blood cells are moving towards the transducer, the frequency will be higher. This is caused by the Doppler effect. The reflected ultrasonic waves return to the crystal and are converted back into electrical signals. The single-element transducer shown in fig. 2.2 has a minimal imaging window and has to be mechanically manipulated to obtain a wide window, which is unfeasible for responsive high-frequency imaging. Thus, usually, a transducer array is used. Various types of US transducer exist with different strengths and weaknesses, shown in fig. 2.3.

2.1.4 Doppler effect

The Doppler effect is a phenomenon in which an observer perceives a shift in the frequency of sound emitted from a source when either the source or the observer is moving, or both are moving. The reason for the perceived change in frequency is visualised in fig. 2.4. In diagram (a), the source S_p is stationary and produces a spherical distribution pattern of

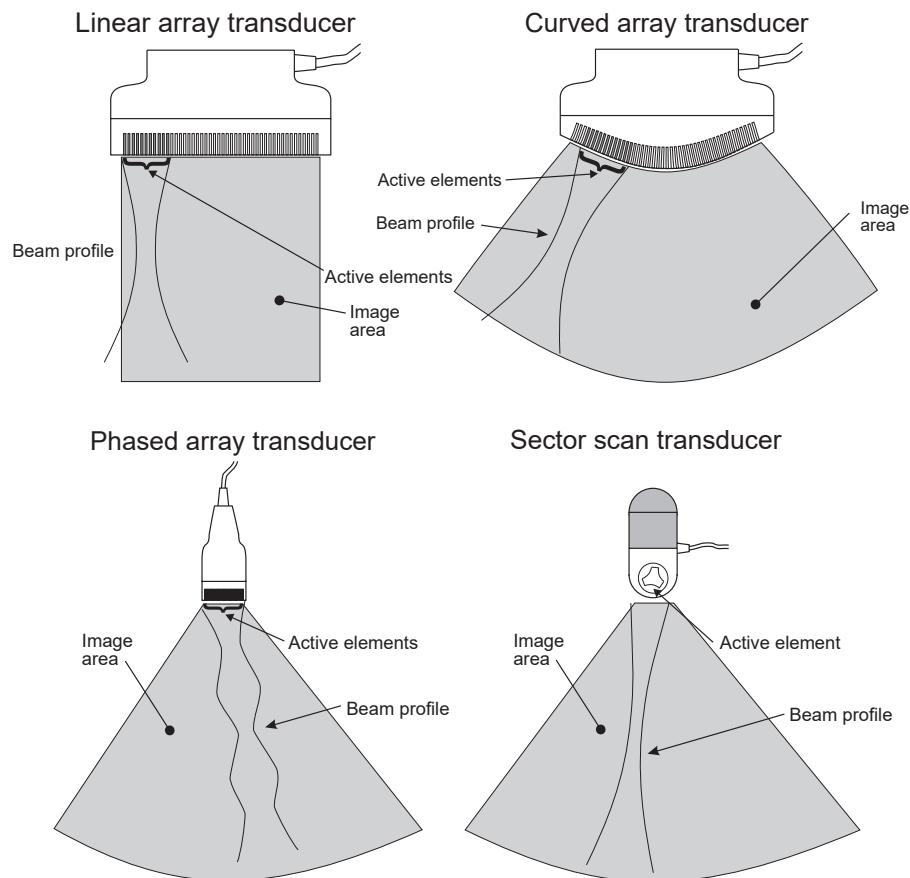


Figure 2.3: Transducer types for acquiring B-mode images [32]

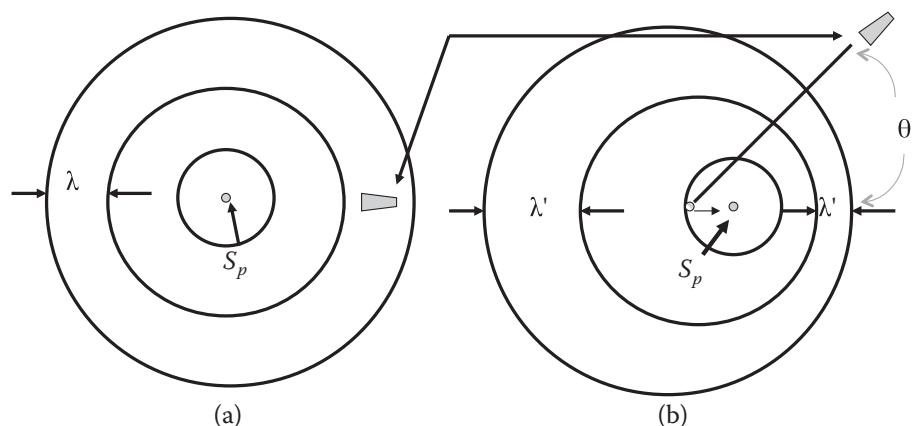


Figure 2.4: Doppler effect diagram. A stationary observer perceives a change in the frequency of a wave generated by a moving source toward the observer as a result of a wavelength shift from λ to λ' . In (a), the source is still. In (b), the source is moving at a velocity v . [39]

the wave with the perceived frequency of the observer is given by $f = c/\lambda$, where c is the velocity of the wave in the medium and λ is the wavelength. In diagram (b), the sound source is moving towards the right with a velocity v . The locomotion of the source changes the distribution pattern and causes a longer wavelength on the left, indicating a lower perceived frequency, and a shorter wavelength on the right, indicating a higher perceived frequency, both denoted as λ' in the diagram. In the case of the observer on the right side, the perceived frequency becomes eq. (2.11).

$$f' = \frac{c}{\lambda} = \frac{c}{\lambda - vT} = \frac{c}{(c - v)T} = \frac{c}{c - v} \cdot f_0 \quad (2.11)$$

And vice versa, on the left side, the perceived frequency becomes eq. (2.12).

$$f' = \frac{c}{c + v} \cdot f_0 \quad (2.12)$$

This perceived difference between the frequency that is transmitted from the source f_0 , and the perceived frequency f' is also called the Doppler frequency, f_d . When these connections are combined, the Doppler frequency for a source moving with velocity v and an observer travelling with velocity v' is given by eq. (2.13).

$$f_d = f' - f = \left(\frac{c + v'}{c - v} - 1 \right) \quad (2.13)$$

If both source and observer are moving with the same velocity, v , assuming $c \gg v$, the v cancels out and the expression is reduced to eq. (2.14).

$$f_d = \frac{2vf}{c} \quad (2.14)$$

If the velocity of the moving source is traveling with an incident angle θ , the v in eq. (2.14) is replaced with $v(\cos \theta)$. This results in the expression found in eq. (2.15) and forms the basis for applied Doppler effect measurements.

$$f_d = \frac{2v(\cos \theta)f}{c} \quad (2.15)$$

The Doppler effect is used in ultrasonic Doppler devices used to image blood flow transcutaneously. An ultrasonic transducer in these devices sends ultrasonic waves into a blood artery, and the scattered radiation from moving red cells is measured by either the same transducer or a second transducer. The Doppler frequency, which is determined by the velocity of red blood cells, is extracted using modern electronic demodulation techniques which will be explored later.

2.2 Flow Physics

The flow physics of the human circulatory system are sophisticated, and numerous non-stationary flow patterns are observed. The human circulatory system takes care of transporting oxygen and nutrients to organs, as well as disposing of waste products produced by metabolism. It is possible because the blood within the circulatory system contains several smaller sub-components, such as plasma and formed cellular elements that

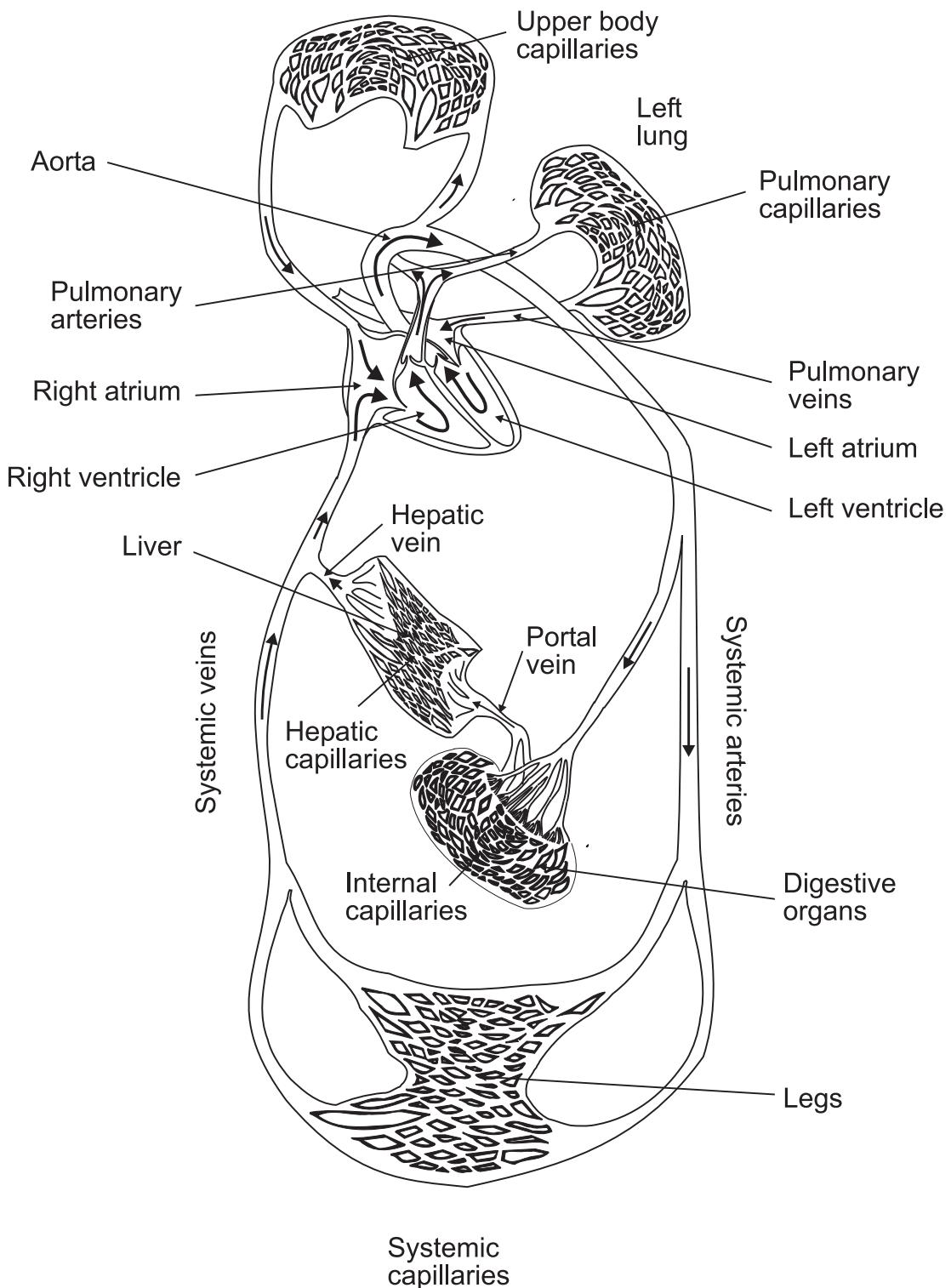


Figure 2.5: Circulatory system of the human body [32]

perform these vital functions. Initially, blood is discharged from the left ventricle of the heart through the aorta and travels to all areas of the body through multiple branches of the arterial tree. When blood flows through the arteries, it enters smaller channels known as arterioles. These arterioles lead to a network of tiny capillaries through which nutrients and waste materials are exchanged between the blood and the organs. The capillaries connect to form a network of venules, which supply the veins and deliver blood back to the heart. This system, in its entirety, is called systemic circulation. A diagram of the circulatory system as described above can be seen in fig. 2.5. In summary, when examining the elements that comprise the circulatory system, it consists of several components:

- Heart, the primary organ of the circulatory system that maintains blood pressure and controls blood velocity.
- Blood, and its sub-components
 - Plasma, which forms the primary volume, contains nutrients and formed cellular elements.
 - Red and white blood cells, which carry oxygen and fight off infections, respectively.
 - Platelets, which are also known as thrombocytes, have the function of clotting during blood vessel injury.
- Blood vessels
 - Arteries (and arterioles), transport oxygenated blood to organs and tissues at high pressure and velocity.
 - Capillaries are thin but wide-ranging blood vessels that perform the exchange of matter between the circulatory system and tissue.
 - Veins (and venules) carry blood back to the heart at low pressure and velocity.

2.2.1 Blood flow

Blood flow is the amount of blood that goes through a blood vessel in a particular period of time, and has a complicated flow pattern due to its pulsing flow. Advanced analysis of haemodynamics is not within the scope of this report, so the explanation will be brief. The primary forces that determine the blood flow F are the pressure difference across a blood vessel and vascular resistance. It is determined by Ohm's law as in eq. (2.16).

$$F = \frac{\Delta P}{R} \quad (2.16)$$

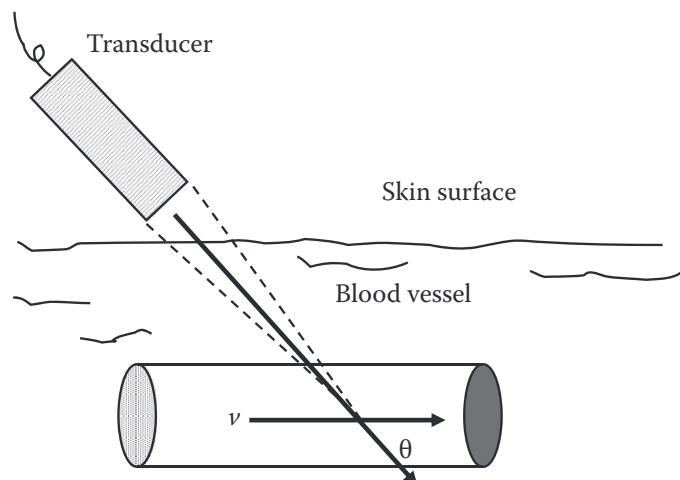
where ΔP is the pressure difference across the blood vessel and R is the vascular resistance. The pressure difference ΔP is calculated with eq. (2.17).

$$\Delta P = P_1 - P_2 \quad (2.17)$$

Where P_1 and P_2 are the blood pressures measured at each end of the blood vessel. Pressure has significant importance on blood flow because an increase in arterial pressure not only increases the force that pushes blood through the capillaries but also expands the vessels, lowering vascular resistance. Selected dimensions and flow characteristics can be seen in table 2.3.

Table 2.3: Typical dimensions and flow in human circulatory system [32]

Vessel	Internal diameter (cm)	Wall thickness (cm)	Length (cm)	Young's modulus (N/m ² ·10 ⁵)	Peak velocity (cms ⁻¹)	Mean velocity (cms ⁻¹)	Reynolds number (peak)	Pulse propagation velocity (cms ⁻¹)
Ascending aorta	1–2.4	0.05–0.08	5	3–6	20–290	10–40	4500	400–600
Descending aorta	0.8–1.8	0.05–0.08	20	3–6	25–250	10–40	3400	400–600
Abdominal aorta	0.5–1.2	0.04–0.06	15	9–11	50–60	8–20	1250	600–700
Femoral artery	0.2–0.8	0.02–0.06	10	9–12	100–120	10–15	1000	800–1030
Carotid artery	0.2–0.8	0.02–0.04	10–20	7–11				600–1100
Arteriole	0.001–0.008	0.002	0.1–0.2		0.5–1		0.09	
Capillary	0.0004–0.0008	0.0001	0.02–0.1		0.02–0.17		0.001	
Inferior vena cava	0.6–1.5	0.01–0.02	20–40	0.4–1	15–40		700	100–700

Figure 2.6: Diagram of US wave transmitted and reaching blood vessel with incident angle θ [39]

2.3 Devices

A device that measures the flowing of blood is called a flowmeter. Flowmeters may be used both inside and outside of vessels. One of the flowmeters that may be used outside the vessel to monitor flow is *US*. Figure 2.6 depicts an ultrasonic wave of frequency f insonifying a blood artery, resulting in an angle of θ relative to velocity v . For simplicity, it is assumed that blood flows in a vessel at a constant velocity v . The echoes returned are shifted in frequency as described in eq. (2.15) earlier in the chapter. The echoes scattered by blood after being insonified by an ultrasonic wave convey information about the velocity of blood flow. Blood flow measurements are often used in clinical settings to determine the status of blood vessels and organ functioning. The two commonly used fundamental techniques for ultrasound Doppler flow measurements are continuous-wave (*CW*) and pulsed-wave (*PW*). Both will be explained.

2.3.1 Continuous-wave Flowmeter

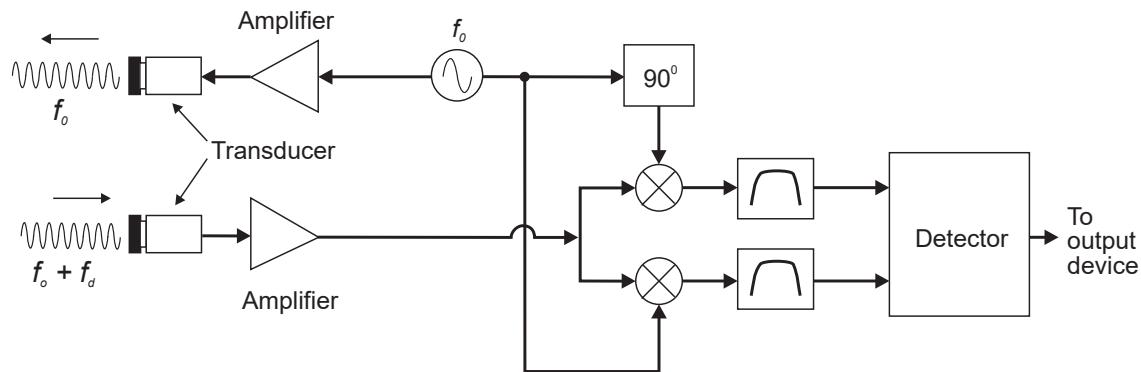


Figure 2.7: Block diagram of *CW* flowmeter [32]

The earliest non-invasive cardiovascular diagnostic technologies relied heavily on *CW* Doppler flowmeters. To continuously transmit waves and receive signals from moving reflectors, the *CW* flowmeter uses two transducers. *CW* flowmeters use less sophisticated electronics than *PW* flowmeters. A drawback to the *CW* flowmeter is the lacking depth discrimination due to the continuous characteristic of this device type. A block diagram of a typical *CW* flowmeter can be seen in fig. 2.7. The basic principles of the device are previously explained in section 2.1.4, and the measurement of the device is described in eq. (2.11). The device continuously emits an ultrasonic wave in the first transducer expressed as a function of time by eq. (2.18) [32].

$$e(t) = \cos(2\pi f_0 t) \quad (2.18)$$

While receiving the backscattered signal on the second transducer expressed by eq. (2.19) [32].

$$r_s(t) = a \cos(2\pi f_0 \alpha(t - t_0)) \quad (2.19)$$

$$\alpha \approx 1 - \frac{2v_z}{c} \quad (2.20)$$

$$\alpha t_0 \approx \frac{2d_0}{c} \quad (2.21)$$

Where v_z indicates the velocity in the z direction. Applying the Fourier transform, the expression yields eq. (2.22).

$$r_s(t) \cdot e^{j2\pi f_0 t} \iff R_s(f - f_0) \quad (2.22)$$

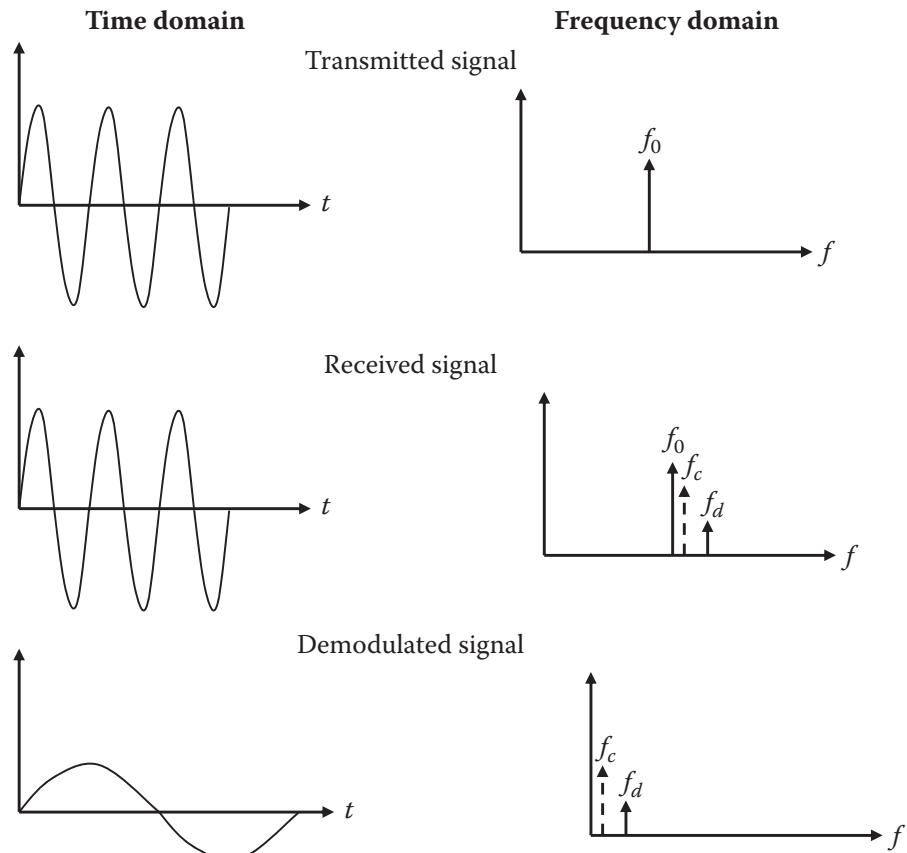
Where $R_s(f - f_0)$ is the Fourier transform of $r_s(t)$. The received signal is then multiplied with a quadrature signal of frequency f_0 to find the Doppler frequency in eq. (2.23).

$$m(t) = a [\cos(2\pi f_0 t) + j \sin(2\pi f_0 t)] \cos(2\pi f_0 \alpha(t - t_0)) \quad (2.23)$$

$$= \frac{a}{2} \left\{ \cos(2\pi f_0[(1 - \alpha)t - \alpha t_0]) + \cos(2\pi f_0[(1 - \alpha)t - \alpha t_0]) \right. \quad (2.24)$$

$$\left. + j \sin(2\pi f_0[(1 - \alpha)t - \alpha t_0]) + j \sin(2\pi f_0[(1 - \alpha)t - \alpha t_0]) \right\}$$

As is general for quadrature demodulation, the resulting signal contains the frequency components of the sum and difference of the emitted and received signals' frequencies shown in fig. 2.8, where the signals are shown in time and frequency domains.



f_c = Frequency of clutter signal e.g., from blood vessel wall

Figure 2.8: Demodulation effects of Doppler signals in time and frequency domain [39]

Generally, a band-pass (*BP*) filter is used on the demodulated signal to remove the high-frequency summed signal at twice the frequency of f_0 . The filtered signal after the *BP* filter is expressed by eq. (2.25) and contains the Doppler shift of the emitted signal.

$$m_f(t) \approx \frac{a}{2} e^{(j2\pi f_0 \frac{2v_z}{c} t)} e^{(-j2\pi f_0 \alpha t_0)} \quad (2.25)$$

Where the second exponential term is the delay proportional to the time between transmission and receiving of the signal. The selected cutoff frequency is chosen to be much lower than the carrier frequency to remove the carrier wave. One issue with ultrasonic Doppler blood flow monitoring is that the blood vessels that generate large reflected echoes are

also moving with a low velocity. These big, slow-moving echoes are referred to as clutter signals in Doppler nomenclature. The band pass filter's low-end cutoff frequency must be designed to minimize interference from these clutter signals. The design of this band pass filter in the low-frequency region, which serves the function of high pass, also known as a clutter rejection filter, has proven troublesome since the magnitude of clutter signals is many orders greater than that of blood and may obfuscate those from slow-moving blood.

Table 2.4: Measured frequency shifts with a Doppler 3 MHz transducer at various velocities at a 45° incident angle [32]

Velocity (v) m/s	Doppler frequency (f_d) Hz
0.01	28
0.1	276
0.5	1377
1	2755
2	5510
5	13 770

Seen in table 2.4 is an example of measured Doppler frequencies using a 3 MHz transducer using the method shown in fig. 2.6. Note that the measured frequencies are all within the audible range.

2.3.2 Pulsed-wave Flowmeter

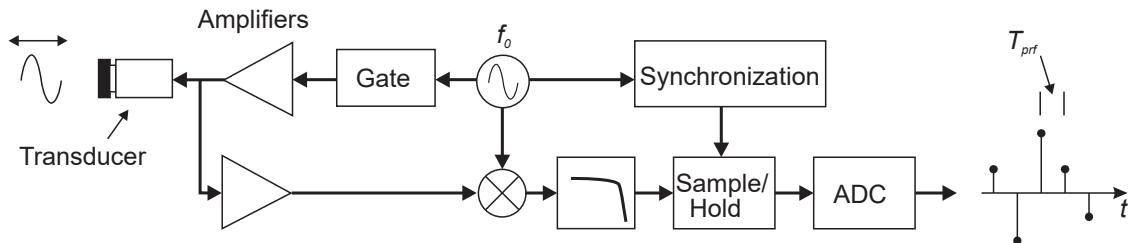


Figure 2.9: Block diagram of *PW* flowmeter [32]

The concept of a pulsed-wave flowmeter was proposed in [2] and other related articles. This type of flowmeter is periodically changing from a transmitter to a receiver. In the transmit mode, the transducer emits a series of pulses. When in the receiving mode, the transducer is listening for the backscattered signal. A simplified block diagram can be seen in fig. 2.9. The movement of particles within the blood causes a displacement in the backscattered signal. These systems are commonly referred to as “Doppler systems” even though it is somewhat misleading. The effects of attenuation are also causing a shift in frequency of a higher magnitude than the velocity of particles in the blood. This is because the conventional Doppler effect is not the straightforward methodology that is applied to the analysis of the back-scattered signal. It is, in fact, an artefact. It is the shift in the location of the scatters that is observed, not the shift in the transmitted frequency. Figure 2.10 shows the received signal after demodulation and filtering; the depth in tissue is fixed here, and the signals displayed on the left side of the figure are the result of a pulse sequence. Each line represents a single pulse, and each pulse is emitted at a pulse repetition frequency, f_{prf} . Instead, on the right, the dotted line shows the sampled signal

formed by taking into account the amplitude of each pulse after a specified time period. To depict the signals on the graph, a single pulse is emitted for each line, and the signals are displaced in amplitude. The sampled signal is displayed on the right. In the pulsed-wave

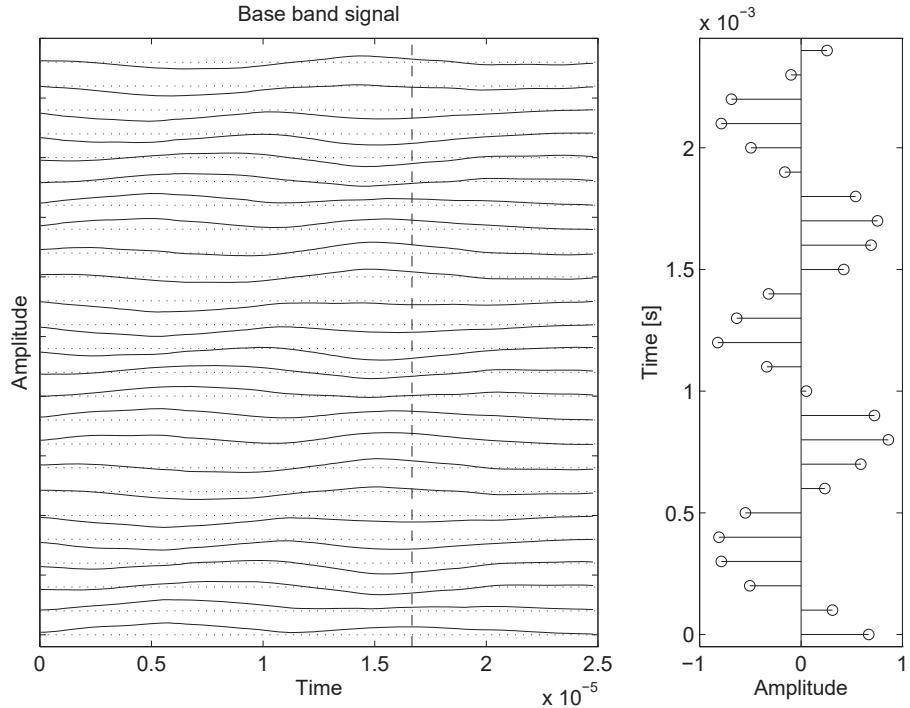


Figure 2.10: Sampling for a gate pulsed wave system with a single range [32]

flowmeter, the received signal is described by eqs. (2.26a) to (2.26d) [32].

$$r_s(t) = a \cdot e^{(\alpha(t-t_0))} \quad (2.26a)$$

$$\alpha = \left(1 - \frac{2v_z}{c} \right) \quad (2.26b)$$

$$\alpha t_0 = \frac{2d_0}{c} \quad (2.26c)$$

$$v_z = |\vec{v}| \cos \theta \quad (2.26d)$$

Equation (2.27) provides the calculation for the time shift t_s of the RF signal between two emissions, which is determined by the distance that the scatterer moves in the direction of the ultrasound beam proportional to the velocity component v_z .

$$t_s = \frac{2v_z}{c} T_{\text{prf}} \quad (2.27)$$

Where c is the speed of sound in the medium, T_{prf} is the interval between each pulse emission. To measure the movement of the scatterer, the signal can be recorded at a certain depth. By selecting one sample at that specific depth for each line, a sampled signal with a frequency that corresponds to the scatter velocity can be obtained. Hypothetically, if the velocity of stationary scatterers in blood was measured, a constant amplitude would be measured. A change in the sample value is observed when there is movement. Between two pulses, the scatterer movement is proportional to the velocity v_z in the direction of the ultrasound beam. Taking one sample from each line at a certain depth yields a sampled signal with a frequency proportional to the scatter velocity. After the back-scattered signal is received it is multiplied by the centre frequency of the emitted pulse and filtered to

remove the sum frequency [32]. A analogue-to-digital converter (*ADC*) quantifies the signal for further signal processing. Referring to displacement fig. 2.10 again, the dashed vertical line represents the sample of each pulse that is taken. If sampling is done T_s after pulse emission, the measurement depth is expressed by eq. (2.28).

$$d_0 = \frac{T_s c}{2} \quad (2.28)$$

Thus, if a sample is taken at the same depth for each line, resulting in a sinusoidal signal proportional in frequency to the scatter velocity [15]. This technique improved the accuracy of the investigations of blood vessels and facilitated the display of velocity profiles. Equation (2.29a) describes the relationship between the received sampled signal for a single scatter and the sinusoidal pulse emitted by the transducer during the i^{th} pulse emission.

$$r(i) = a(i) \sin(2\pi f_p i T_{\text{prf}} + \theta) \quad (2.29a)$$

$$f_p = \frac{2v_z}{c} f_0 \quad (2.29b)$$

Where f_p is expressed by eq. (2.29b), $a(i)$ is the amplitude for the i^{th} pulse, f_0 is the emitted frequency, and θ is the phase factor proportional to the depth of interest.

CW og PW
on

2.4 Blood Velocity Estimation

This section will explain the methodology of velocity estimation, assuming a pulsed-wave system signal is obtained from a back-scattered signal. There are variations of methods that can be applied, but only the implementation that will be used is discussed in this report.

2.4.1 Spectral Envelope

Figure 2.11 displays a signal where the scatterers in the sampled depth d_0 move away from the transducer, and the shift in the pulsed signal is present. The measurement is created by taking one sample at a specific depth (d_0) for each RF line. The sampling is performed at a frequency of f_{prf} , as indicated by the dotted line in the figure. Due to the slow movement of the received signals past the sample volume, there is a gradual change in the sampled signal over time. As a result of the slow movement of the signals, the frequency of the sampled signal is much lower than that of the RF signal. It's worth noting that the received signal not only shifts in time from pulse to pulse, but its shape also changes. This is because the signal is constructed by adding up responses from many scatterers that move at different speeds. A stationary structure will result in an identical sample value for all segmented RF lines. The spectrum of this stationary signal is shown in eq. (2.30).

$$|H_s(f)| = \left| a \frac{\sin(\pi f N T_{\text{prf}})}{\sin(\pi f T_{\text{prf}})} \right| \quad (2.30)$$

which for $f = 0$ is equal to aN and has its first zero at f_{prf}/N . The signal spectrum based on eq. (2.30) is shown in fig. 2.12.

When dealing with small blood vessels, the stationary echoes can be significantly stronger (up to 40 dB) than the blood signal itself. To properly visualize blood flow details, the stationary signal must be eliminated from the sonogram. This requires setting the cutoff frequency to at least $f_{\text{prf}} = N$, in order to remove the main component. In some cases, a higher cutoff frequency may be necessary, particularly if the stationary echo is strong or the vessel wall is in motion. The speed and pulse repetition frequency of a single moving

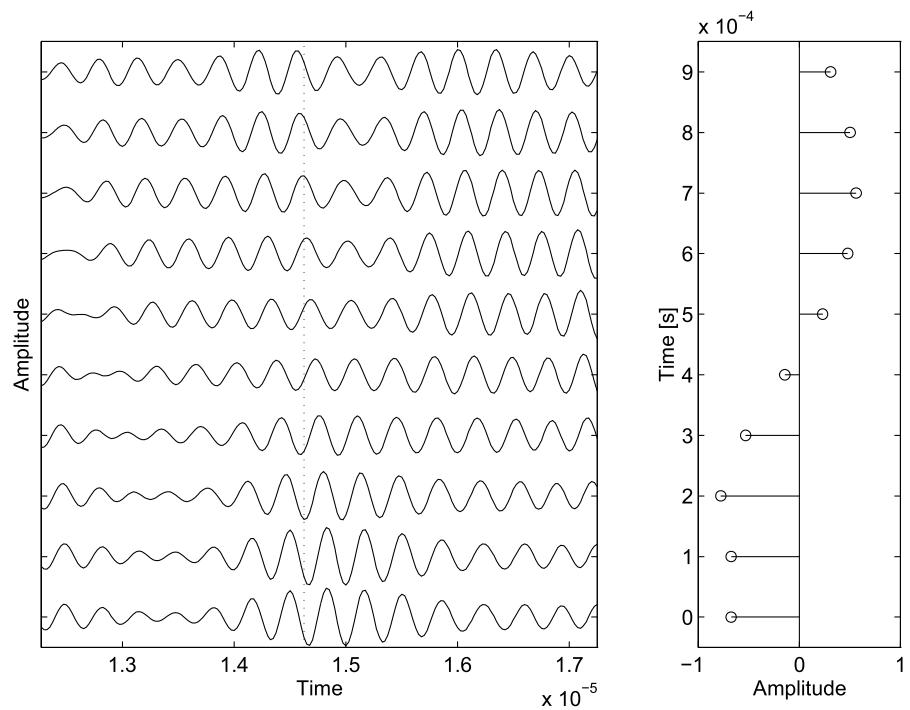


Figure 2.11: Simulated RF sampling of a blood vessel with segmented receiver line with a sample interval denoted by a dotted line (left) and the resulting amplitude of the sample (right) [32]

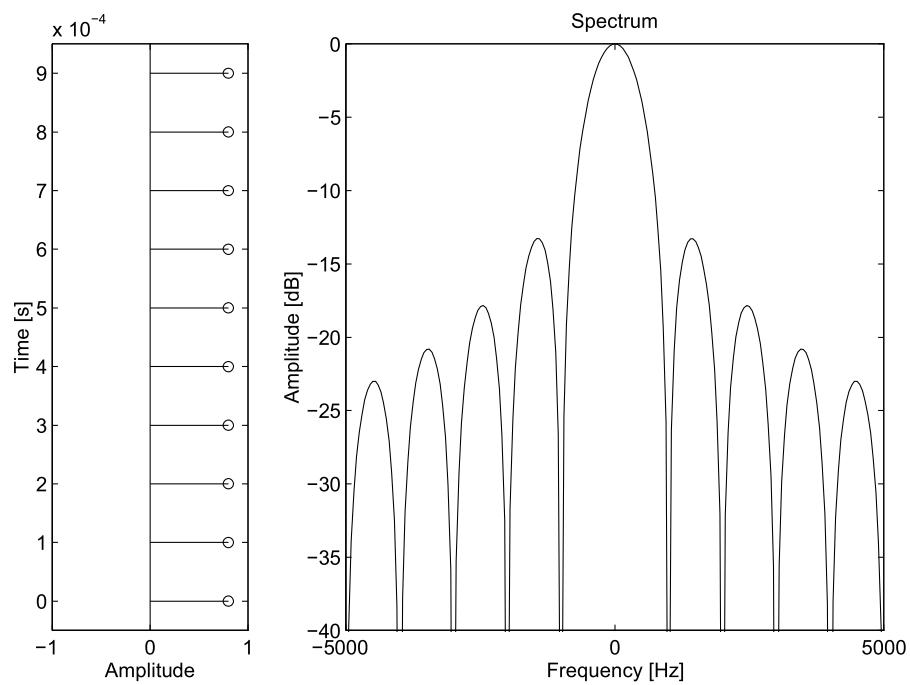


Figure 2.12: Stationary tissue sampled signal (left) and stationary tissue sample spectrum (right) [32]

scatterer affects the signal received. The signal will look like the pulse, but its time scale and frequency will differ from the RF pulse because the pulse waveform slowly moves past the point where it is measured. Figure 2.13 shows this, where a beam from a concave transducer is crossed by a single scatterer. The time shift between each line is determined by the expression in eq. (2.31).

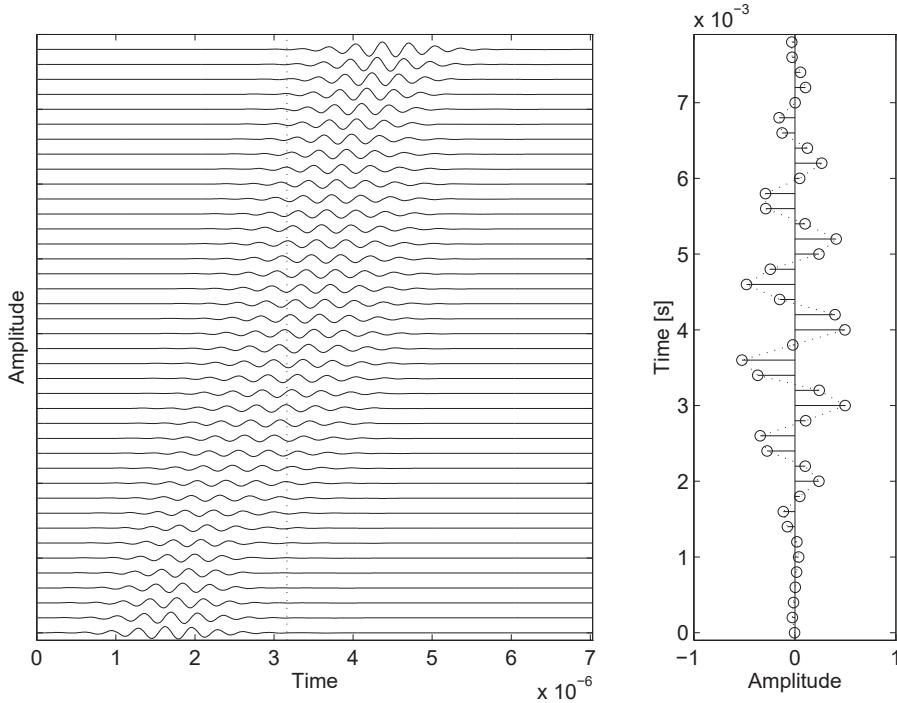


Figure 2.13: Single moving scatterer crossing a concave transducer beam [32]

$$t_s = \frac{2v_z}{c} \cdot T_{\text{prf}} \quad (2.31)$$

Which increases linearly by line i . Individual sinusoidal components are expressed by eq. (2.32).

$$y(i) = \sin \left(2\pi \frac{2v_z}{c} \cdot f_0 i T_{\text{prf}} \right) \quad (2.32)$$

Where the variable iT_{prf} corresponds to time and f_0 corresponds to the fundamental frequency. The received signal frequency is expressed by $2v_z/c \cdot f_0$, where the frequency axis is scaled by factor $2v_z/c$ as seen in fig. 2.14. The center frequency transforms to $f_p = \frac{2v_z}{c} f_0$. Every other spectral element is transformed by factor $\frac{2v_z}{c}$ and the received signal has the shape of a scaled frequency pulse. A sufficient number of lines must be captured, or only a part of the pulse is obtained. In practical terms, it corresponds to weighing the pulse with a rectangular window, which broadens the spectrum. The spectrum of the window is expressed by eq. (2.33) and is convolved with the spectrum of the pulse. A narrow spectrum is seen for slow velocities and the resulting spectrum is nearly determined solely by the spectrum of the window.

$$W(f) = \frac{\sin(\pi f N T_{\text{prf}})}{\sin(\pi f T_{\text{prf}})} e^{(-j\pi f N T_{\text{prf}})} \quad (2.33)$$

This window limitation does not apply, as long as the whole pulse is sampled. If $N T_{\text{prf}} = cM/2v_z f_0$, the number of lines acquired and the width of the rectangular pulse are matched.

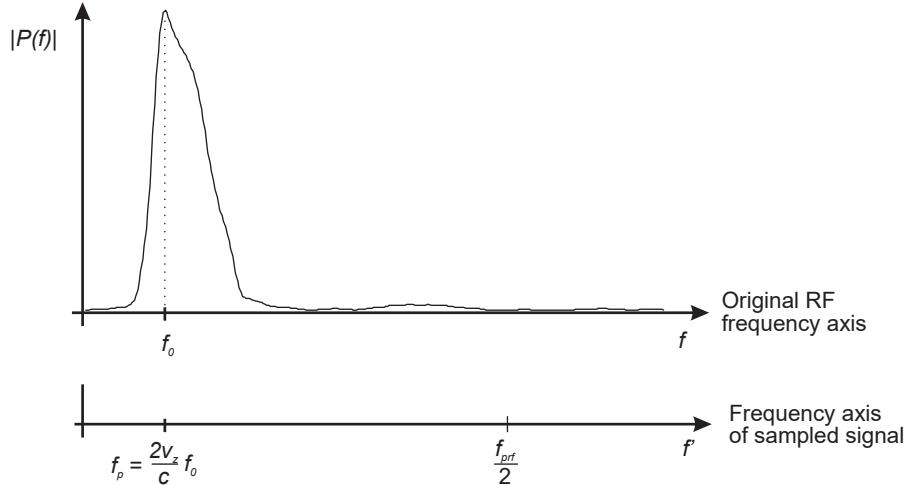


Figure 2.14: Frequency axis scaling for scatterer with velocity v_z [32]

A limitation on the lowest possible velocity and frequency is found through eqs. (2.34a) to (2.34c).

$$NT_{\text{prf}} = \frac{1}{\frac{2v_{\min}}{c} f_0} \quad (2.34a)$$

$$v_{\min} = \frac{c}{2} \cdot \frac{f_{\text{prf}}}{N f_0} \quad (2.34b)$$

$$f_{\min} = \frac{f_{\text{prf}}}{N} \quad (2.34c)$$

And conversely, the maximum velocity is determined by the pulse repetition frequency f_{prf} , as aliasing occurs at frequencies $f_{\text{prf}}/2$ or above. The relation of maximum velocity is expressed by eqs. (2.35a) and (2.35b).

$$\frac{f_{\text{prf}}}{2} \leq \frac{2v_{\max}}{c} f_0 \quad (2.35a)$$

$$v_{\max} = \frac{c}{2} \cdot \frac{f_{\text{prf}}}{2f_0} \quad (2.35b)$$

However, this does not consider the pulse bandwidth. In practice, the maximum velocity v_{\max} will be slightly lower.

2.4.2 One-Dimensional Velocity Estimation

If a pulsed sinusoidal signal, i.e. $p(t) = \cos(2\pi f_0 t)$, is emitted into an ultrasound field a number of times, the returned signal is sampled at a depth of interest, d_0 . The sampled signal of a monochromatic wave yields eq. (2.36).

$$r(k, l) = \cos \left(2\pi \left(\frac{f_0}{f_s} k - \frac{2v_z}{c} f_0 l T_{\text{prf}} \right) \right) \quad (2.36)$$

Where c is the speed of sound, v_z is the blood velocity component along the axial beam, f_0 is the carrier frequency, l is the emission times, k is the sample depth, T_{prf} is the time between each pulse burst, f_s is the sampling frequency, and $\varphi = 2\pi f_0 / f_s$ is the phase factor for the depth of interest. The returned signal frequency is given by eq. (2.37) and is proportional to the axial blood velocity component.

$$\psi = -2\pi \frac{2v_z f_0}{f_{\text{prfc}}} \quad (2.37)$$

To obtain positive and negative velocities, a one-sided spectrum should be used in the signal, which can be found with a Hilbert transform of the signal [33]. The sampled signal is expressed by eqs. (2.38a) and (2.38b).

$$r_q(k, l) = e^{j(\Phi k + \phi l)} \quad (2.38a)$$

$$\Phi = 2\pi \frac{f_0}{f_s} \quad (2.38b)$$

2.4.3 Sonography

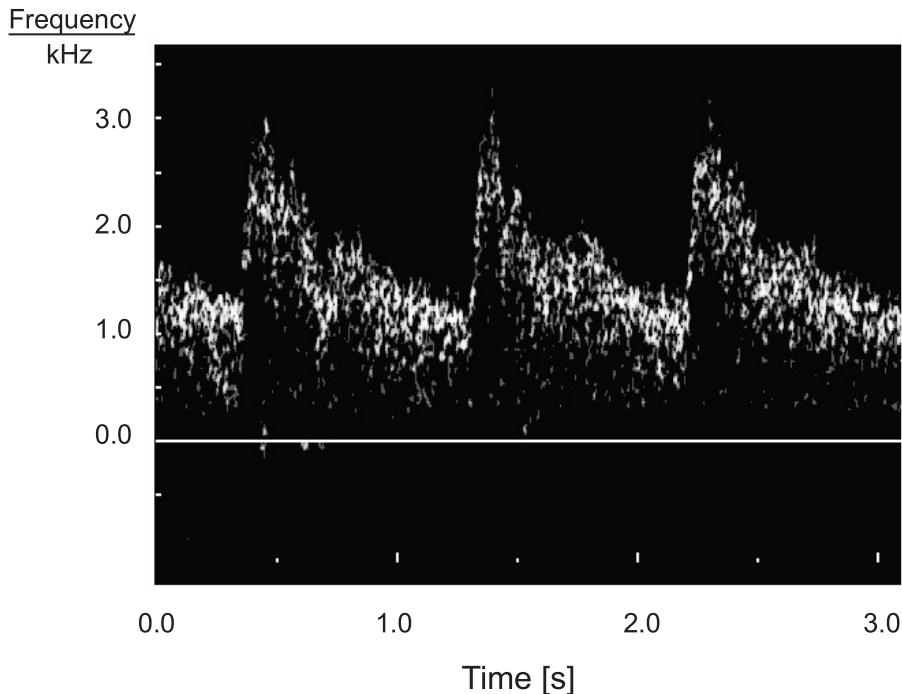


Figure 2.15: Arterial sonogram with time-frequency and Doppler shift [32]

Given that the frequency volume of the received signal is similar to the blood's velocity distribution, the Fourier transform of the received signal can be used to obtain velocity. The spectrogram, usually erroneously known as the Doppler spectrum, can be created by saving the *PSD* together. The *PSD* is calculated for each of the components that make up the received signal in order to accomplish this. A quadrature-demodulated signal is used to display both positive and negative frequencies. When these spectra are shown side by side, the evolution of the velocity distribution can then be seen. Sonography of an artery is displayed in fig. 2.15.

3 Synthesis

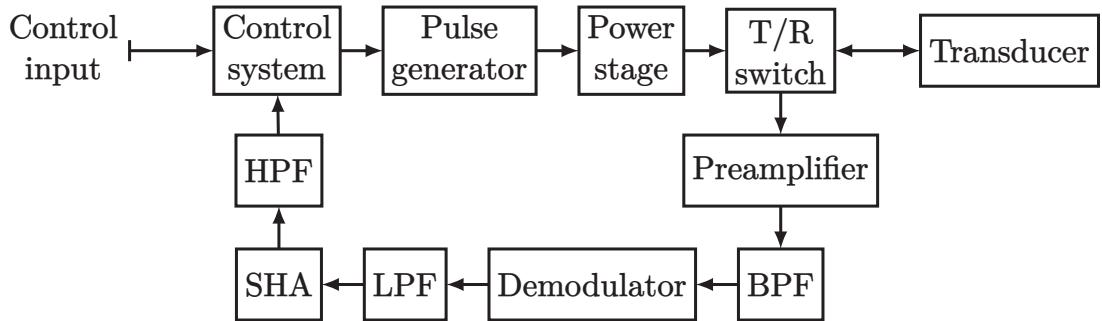


Figure 3.1: Simplified overview of the entire system

A simplified overview of the entire system can be seen in fig. 3.1. Each of the various modules will be explained during this chapter of the report. Initially, the control system will be briefly explained and the reasons for its design choice. Secondly, the signal chain in the transmitter will be outlined and how the transducer is driven by the power stage with the added protective switching circuit. Finally, the analogue front-end will be further explained with its various subcircuits for filtering, amplifying, demodulating, and sampling the signal. Lastly, the design of the digital signal processor (*DSP*) within the control system will be explained.

3.1 Control System

The choice of platform for controls and data acquisition is an embedded system. A microcontroller is a small computer that is built into a single integrated circuit (*IC*) chip. It includes a central processing unit (*CPU*), memory, and *I/O* peripherals, and it is designed to perform a specific set of tasks. Microcontrollers are used in a wide range of electronic devices, including appliances, automobiles, industrial control systems, and consumer electronics. Microcontrollers are often used in applications where a small, low-power device is needed to perform simple tasks, such as controlling a motor or reading a sensor. They are usually programmed in a high-level language, such as C or C++, and they can be programmed to perform a variety of tasks, depending on the specific application. The chosen microcontroller unit (*MCU*) for this project is XNUCLEO-F411RE, visible in fig. 3.2, because it is sufficient for the application and sourcing limitations within the *IC* supply chain. For implementing the control system, a real-time operating system (*RTOS*) can offer multiple benefits for the embedded system development. A RTOS is an operating system that is designed to handle real-time applications. Real-time applications are those that require timely processing of data in order to function correctly. This can include tasks such as controlling industrial machinery, monitoring and controlling processes. Real-time operating systems are designed to prioritize certain tasks and ensure that they are completed within a specific timeframe. They do this by allocating a certain amount of processing resources to each task, and by interrupting the execution of lower-priority tasks as needed to ensure that high-priority tasks are completed on time. RTOSs typically include features such as preemptive scheduling, real-time communication, and support for multiple processors and hardware architectures. Alternatively, a vendor-locked baremetal

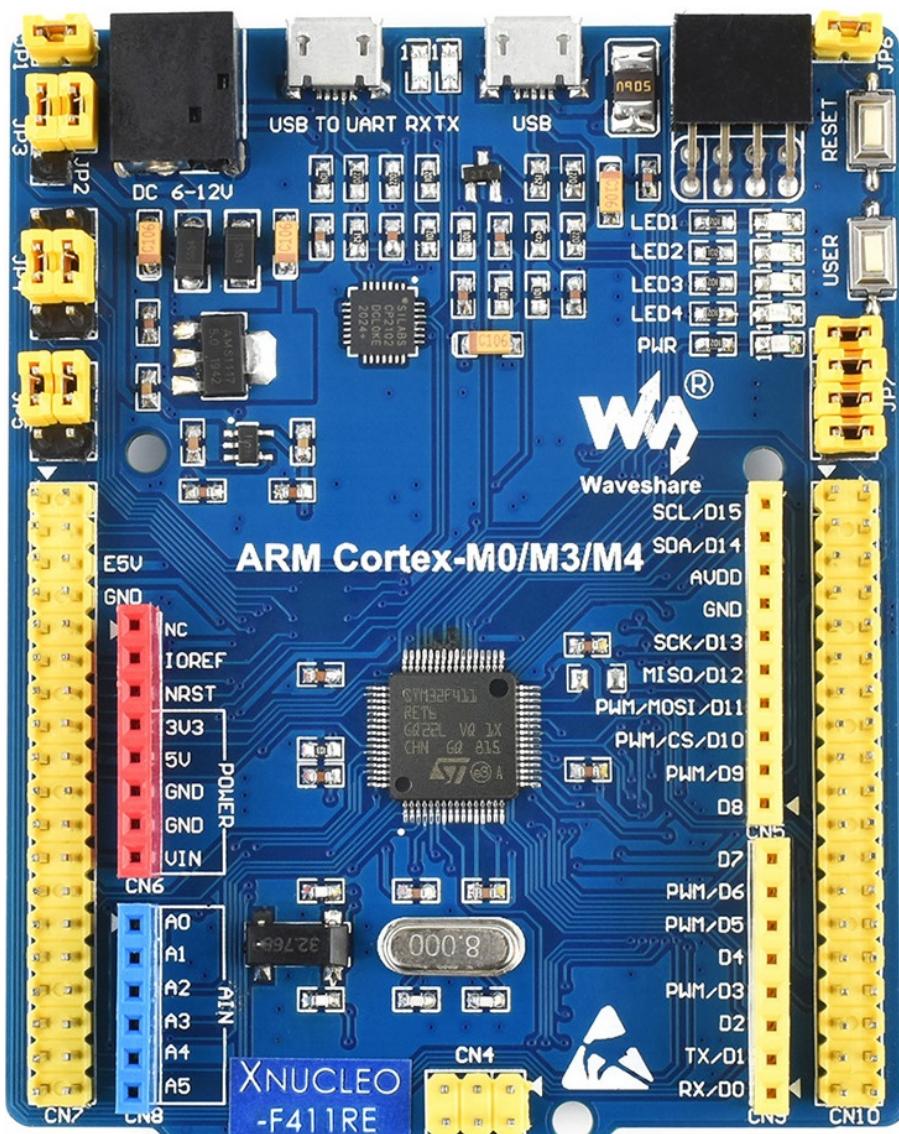


Figure 3.2: XNUCLEO-F411RE development board by Waveshare

implementation is an option, in this case, STM32 HAL. Notable differences between the two approaches are, but not limited to:

- Multitasking: RTOS allows for parallel execution that enable more complex applications.
- Portability: Standard modules mean that the same code can be easily ported to other devices and even other platforms without modifications.
- Reduced development time: Especially for rapid prototype development, using pre-existing APIs significantly reduces development time by providing many of the low-level tasks such as scheduling, resource management, and timing by the operating system.

3.2 Pulse Generator

Initially, a pulse generator was designed by using a programmable synthesizer circuit, but due to constraints within generating complementary PWM with dead-time when driving the half-bridge, a timer based PWM generation in the microcontroller is preferable. In a half-bridge power stage, dead-time refers to the amount of time that elapses between the moment when one of the switches in the half-bridge (either the high-side or the low-side switch) turns off and the moment when the other switch turns on. During the dead-time, both switches in the half-bridge are off, which means that there is no current flowing through either switch in the half-bridge. A scenario where both switches are on, can cause problems if the output of the half-bridge is connected to a load, as it may cause the load to behave erratically or even be damaged. To avoid these problems, it is important to carefully consider the amount of dead-time in a half-bridge power stage. In general, a longer dead-time will reduce the risk of damage to the load, but it will also reduce the efficiency of the power stage, as energy will be lost during the dead-time. Therefore, it is important to carefully balance the trade-off between efficiency and safety in order to determine the optimal amount of dead-time for a given half-bridge power stage. Based on discussions during design review, it was decided to change approach and generate the two complementary PWM signals by configuring the PWM module of the controller with the functionality though with a trade-off in resolution. However, for this application there is no need to amplitude modulate the output signal. In the context of a PW Doppler system, it is desired to generate 4 signals:

- 5 MHz complementary signal PWM with dead-time for the pulsed burst during transmit mode.
- 5 MHz complementary signal PWMN with dead-time and opposite phase from PWM.
- 10 kHz signal PRF for the timing control of the transmit/receive switch.
- 20 MHz clock signal CLK for the demodulation circuit in the receiver.
- PULSE and GATE controlled by PRF for S/H control with pulse length $t_{pulse} = N_{pulse} \times T_{prd}$, where $T_{prd} = 1/f_{pwm}$. GATE is delayed from pulse by sample depth.

3.3 Power Stage

Several MOSFET drivers were considered, e.g. ISL55111[55], EL7104[20], and MD1213[45]. The MD1213 has an advantage over the ISL55111 or EL7104 for ultrasound MOSFET drivers since it is specifically designed for high-voltage P-channel and N-channel MOSFETs

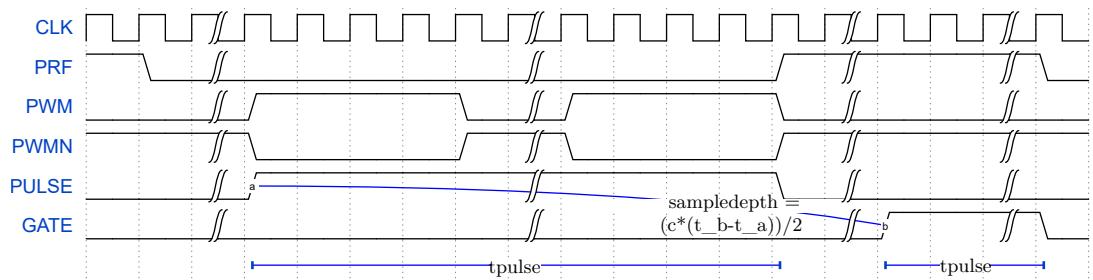


Figure 3.3: Timing diagram of various control signals for an arbitrary n length pulse chain expressed by the second diagram gap

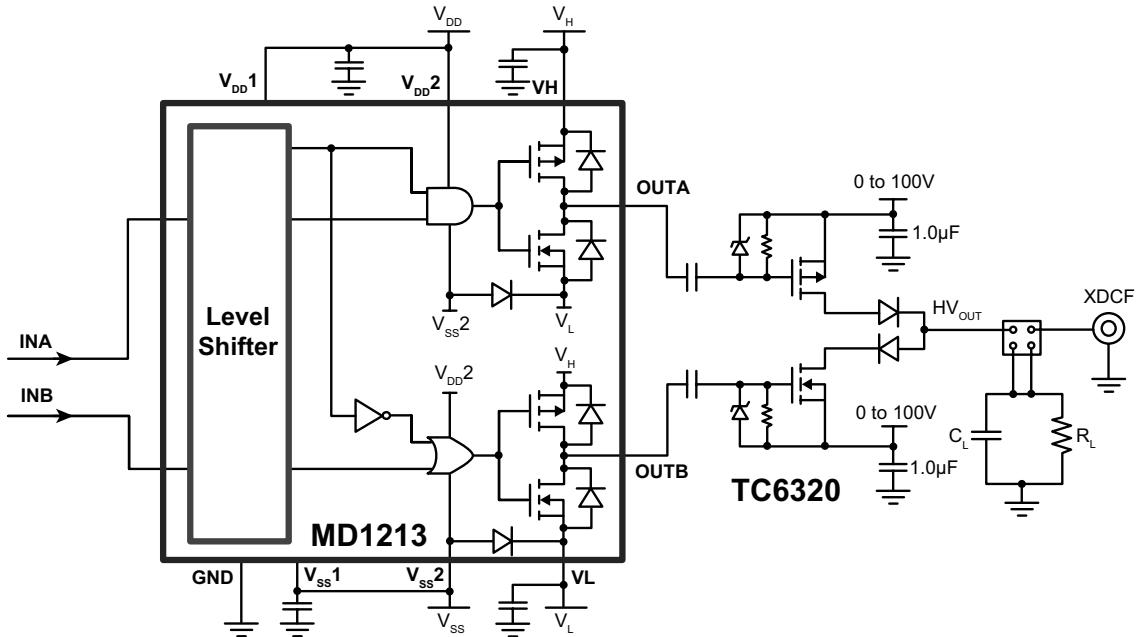


Figure 3.4: Block diagram of power stage [36]

in medical ultrasound and other applications needing a high output current for a capacitive load. It has a high-speed input stage with a logic interface that can function from 1.8 V to 5 V and an ideal operating input signal range of 1.8 V to 3.3 V. The DC-coupled adaptive threshold circuit sets the level translator switch threshold to the average of the input logic LOW and logic HIGH levels. Consequentially, the MD1213 is designed primarily for driving MOSFETs in medical ultrasound applications, whereas the ISL55111 and EL7104 are more general-purpose drivers that may not perform as well in ultrasound applications. The MD1213's output stage has a distinguishing feature in that the LOW and HIGH levels of the output signal may be set independently of the rest of the circuit's supply voltages. The input logic levels, for example, might be 0 V and 1.8 V, whereas the control logic is powered by +5 V to -5 V. The output LOW and HIGH values, on the other hand, may be changed between -5 V to +5 V. This gives you greater flexibility in adjusting the output signal levels to meet individual needs. The output stage may also provide peak currents of up to 2 A, depending on the load capacitance and supply voltages employed. Seen in section 3.3 is the circuit diagram of the power stage with the gate driver on the left side and the half-bridge on the right side. Using a SPICE macro model, an LTspice simulation

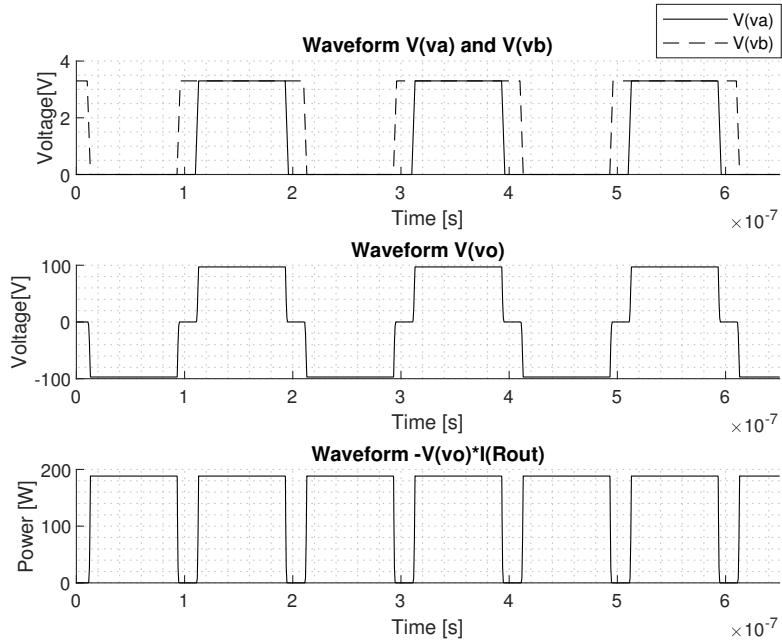


Figure 3.5: LTspice simulation output of transmitter with level shifter and half-bridge power stage from fig. B.1

of the power stage was implemented where the full model can be seen in fig. B.1. The resulting waveforms are seen in fig. 3.5. In the top subplot, the input voltages V_{INA} and V_{INB} are seen with their dead-time visible on each overlapped on period. Since V_{INB} is driving an N-channel metal-oxide-semiconductor field-effect transistor (*MOSFET*), the driving pulse train should be thought of as having the opposite polarity. When looking at the middle subplot, it is noted that dead time is visible as the time when the output voltage is zero. Thus, during that time neither field-effect transistor (*FET*) are allowing a current to pass, and therefore the voltage across the load is equal to zero. The lower subplot shows the maximum ideal power delivery using the peak pulse voltage, assuming the load is equal to $50\ \Omega$. In reality, due to the laboratory instruments available for experiments, the pulse peak voltage will be less than $\pm 100\text{ V}$.

3.4 Transmit/Receive Switch

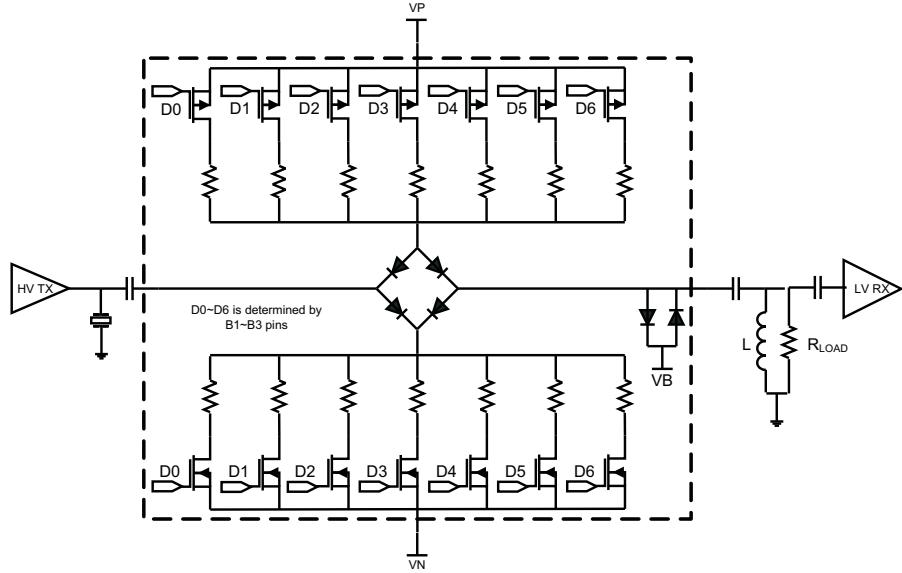


Figure 3.6: Block diagram of TX/RX switching circuit where the inputs D1 through D6 are binary decoded from inputs B_1, B_2, B_3 [25]

Among the design considerations for the transmit and receive switch were the TX810[25] and MD0101[52]. Both ICs are acceptable choices, however, the MD0101 is a newer and generally better choice since it has a lower insertion loss, which means that less of the ultrasound signal is lost as it passes through the switch. This results in a higher-quality image with a better signal-to-noise ratio. Additionally, MD0101 has a wider bandwidth, which means that it can transmit and receive ultrasound signals over a broader range of frequencies. However, since the TX810 is in stock and is also acceptable, it was chosen for the design. TX810 is an IC from Texas Instruments that can be used to switch transmit and receive paths of an ultrasound system. The IC fundamentally works by having a 3-bit programmable pin interface that will open and close the switch with a variable bias current. See fig. 3.7 for a visualisation of the switching operation, where INPUT is the incoming Doppler waveform being picked up from the transducer, $B_3/B_2/B_1$ is the switching signal closing the switch and thereby going in receive mode, and OUTPUT is the received signal seen in the AFE. When high-voltage transmitter signals are applied to the input, the internal diodes limit the output voltage. While in receive mode, the TX810's insertion loss is minimized. The TX810 features a 3-bit interface that may be used to program bias current from 7 mA to 0 mA for varying performance and power requirements, unlike conventional T/R switches. The device is put up in power-down mode when the TX810 bias current is set to 0 mA (high-impedance mode). The TX810 does not put a significant load on high-voltage transmitters when operating in the high-impedance mode.

Seen in fig. 3.8 is the recovery time between transient states in the TX810 diode bridge where the red curve is the input and blue is the output signal. According to the specified output curve, it takes approximately 15 μ s to recover from the transmitting pulse. Thus, back-calculating to find the minimum distance to reliably receive an undisturbed signal is

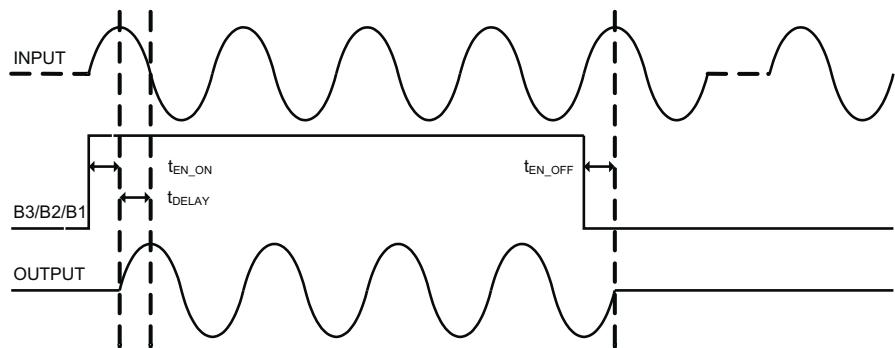


Figure 3.7: Timing diagram of switching interface where $t_{EN_ON} = 0.6 \mu s$, $t_{EN_OFF} = 2.4 \mu s$, and $t_{DELAY} = 1.3 \text{ ns}$ for the condition $B_1 = B_2 = B_3$ [25]

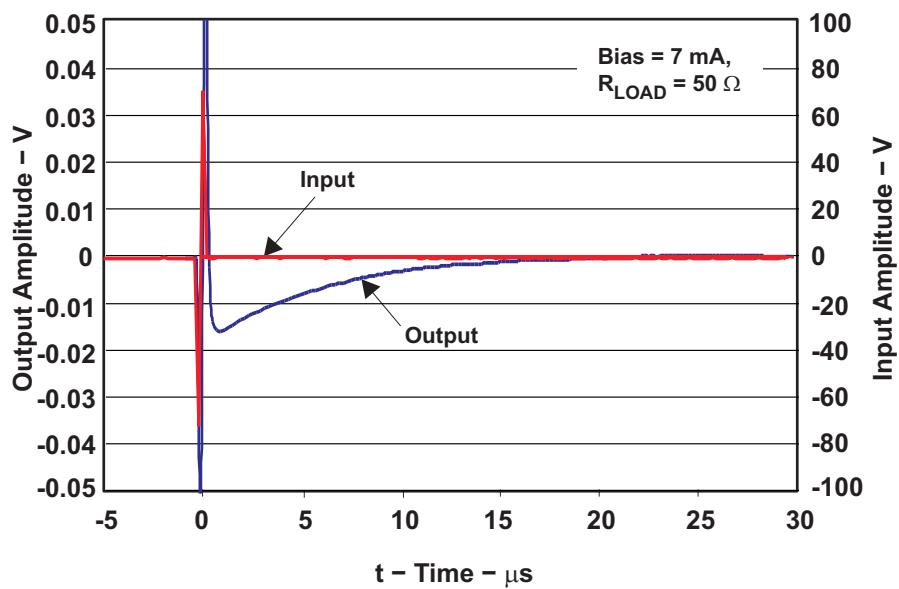


Figure 3.8: Recovery time from transmitting to receiving state with an AC coupled high-voltage pulse and TX810 [25]

proportional to eq. (3.1).

$$d = \frac{c \cdot t}{2} \Rightarrow \quad (3.1a)$$

$$d = \frac{1480 \text{ ms}^{-1} \cdot 15 \mu\text{s}}{2} = 11.55 \text{ mm} \quad (3.1b)$$

Where d is equal to the travel distance from the transducer to the scatterer, c is the speed of sound in water (assumed to be 1480 ms^{-1}), and t is the recovery time as specified in fig. 3.8. The reason for dividing the distance by half is because the travel time is double the distance since the acoustic wave has to travel the distance from the transducer and the reflected wave has to travel back the equal distance. This means that a distance of less than 11.55 mm to a scatterer is likely to produce an unreliable measurement. The ultrasound switch is designed to switch the transmit and receive paths at specific times, as determined by the input signals. A PCB design was implemented in Altium Designer [59] utilising three channels of the maximum eight available channels in the IC. Seen in fig. 3.9 is a 3D render of the designed PCB.

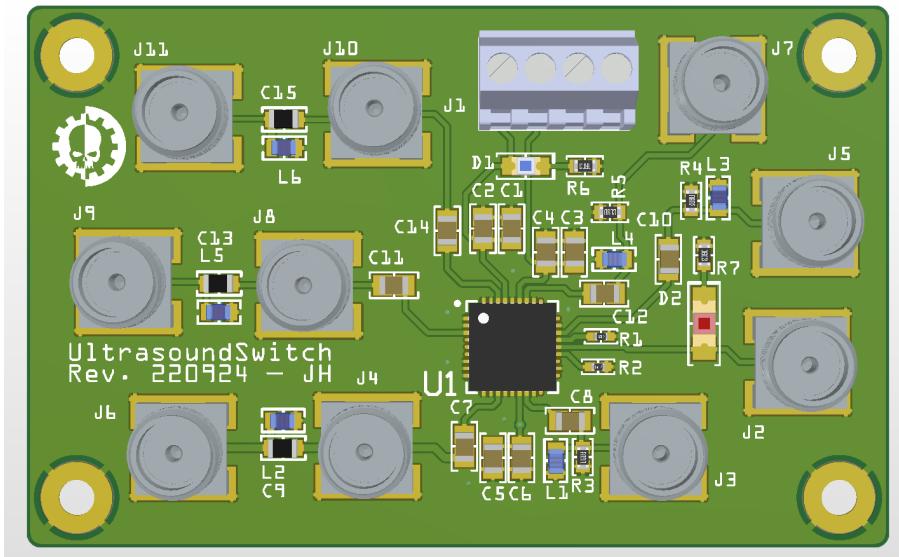


Figure 3.9: 3D Render of PCB in Altium Designer

The module is designed with three usable channels, either three separate transducers for multi-angle sonography, or a capacitive micromachined ultrasound transducer (*CMUT*) with three channels in a single angle. However, in the following experiments with the TX/RX switch, only one channel will be used for simplifying the data acquisition experiments.

3.5 Band-pass Filter

After the signal is received, it is filtered with a *BP* filter to remove unwanted noise and interference from the received signal. The presence of these unwanted frequency components can distort the received signal and reduce the quality of the resulting imaging. The specs from the datasheet of used modular component BPF-C4R5+ [48] are seen in fig. 3.10. Filtering the received signal through this device means that any signals produced by the transducer at frequencies outside the range of interest will be attenuated. An ultrasound receiver requires a band-pass filter to enable only the frequencies within a predetermined range to pass through while blocking out frequencies outside that range. Using the specs and S21 forward transmission coefficient data, a bode plot can be visualised in fig. 3.11.

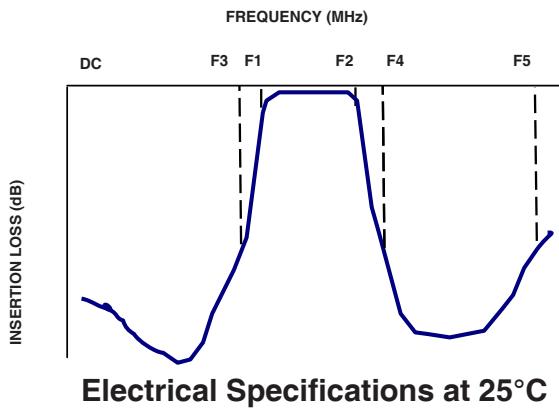


Figure 3.10: Band-Pass Filter with (above) insertion loss showing a pass band of 2 MHz to 7 MHz of 0.5 dB insertion loss and (below) electrical specifications showing the minimum stopband attenuation of 20 dB [48]

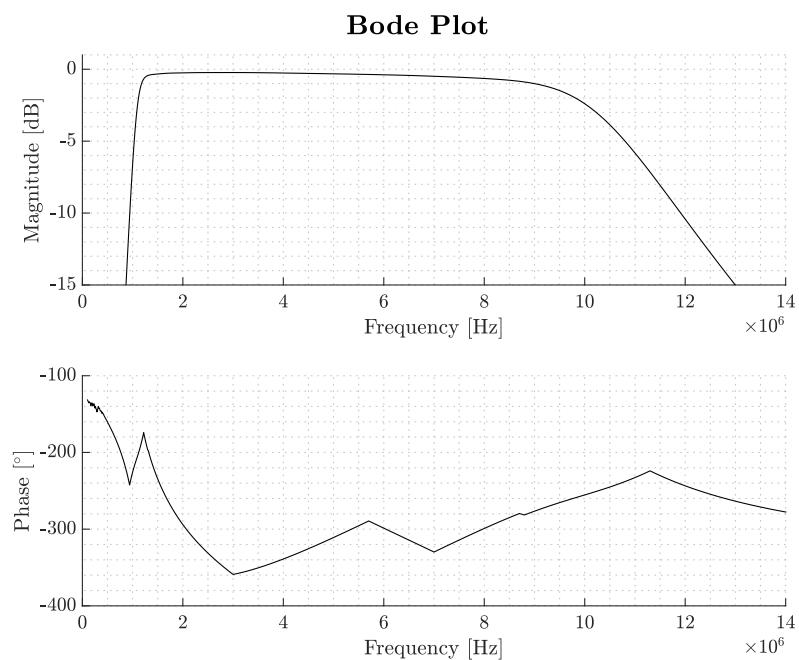


Figure 3.11: Bode plot of BPF-C4R5+ S21 forward transmission coefficient of the active filter as specified by the manufacturer

3.6 Preamplifier

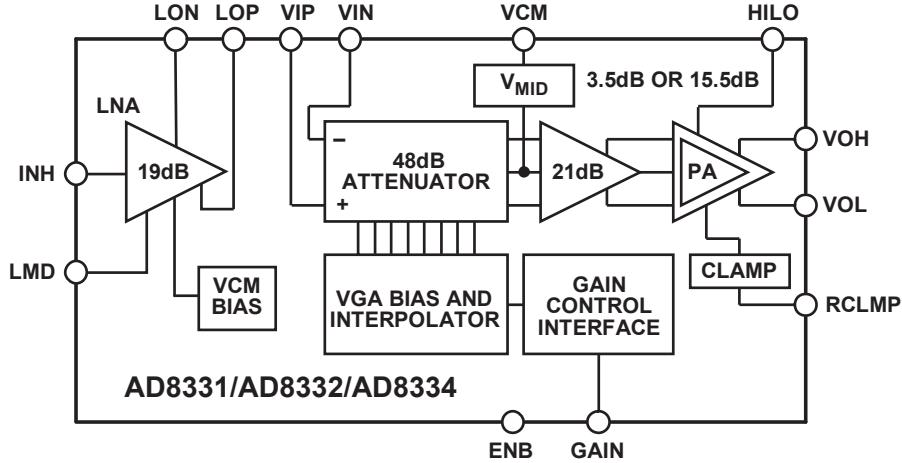


Figure 3.12: Block diagram of preamplifier AD8332 [41]

The isolated signal is still rather weak to be measured using digital circuits, and therefore the amplitude must be increased with the preamplifier circuit. This circuit is based on the integrated circuit from Analog Devices AD8332 [41], which is a device that combines a dual-channel low noise amplifier (*LNA*) and variable gain amplifier (*VGA*), designed specifically for ultrasound systems. A diagram of its internal functional blocks can be seen in fig. 3.12. The AD8332 functions at frequencies up to 120 MHz. Each channel includes an ultralow noise preamp (*LNA*), a *VGA* with 48 dB of gain range, and a selectable gain post amp with adjustable output limiting. The LNA gain is 19 dB with a single-ended input and differential outputs. To match the signal source without sacrificing noise performance, the LNA input impedance can be adjusted using a single resistor. The *VGA* has low output-referred noise, which is useful in driving high-speed differential ADCs. The gain of the post amp can be pin-selected to 3.5 dB or 15.5 dB, depending on the converter requirements. The output can be limited to a user-defined clamping level to avoid input overload to a subsequent *ADC*, with the clamping level adjusted using an external resistor. A SPICE macro model is provided by the vendor and the preamplification is successfully simulated using LTspice with the full LTspice model found in fig. B.2, and the probed inputs and outputs seen in fig. 3.13.

3.7 Quadrature Demodulator

After the preamplifier, the amplified signal must be demodulated to prepare it for sampling. The device used for quadrature demodulation is an integrated circuit from Analog Devices AD8333 [42] I/Q demodulator. A diagram of the internal functional blocks can be seen in fig. 3.14 where the primary inputs are RFIP and RFIN, which are the two differential RF signals from the preamplifier. The RF inputs connect directly to the outputs of the LNA of the preamplifier. The internal 0° and 90° phases of the local oscillator (LO) are generated by a divide-by-4 circuit that drives the mixers of a matched I/Q demodulator pair. The I and Q outputs are presented as currents, making summation possible. The summed current outputs are then converted to voltages by a high dynamic range, current-to-voltage (I-V) converter, such as the AD8021 [18], which functions as a trans-impedance amplifier. A SPICE macro model is provided by the vendor and the I/Q demodulation is successfully simulated using LTspice using the LTspice model found in appendix fig. B.3, with the probed inputs and outputs seen in figs. 3.15 and 3.16.

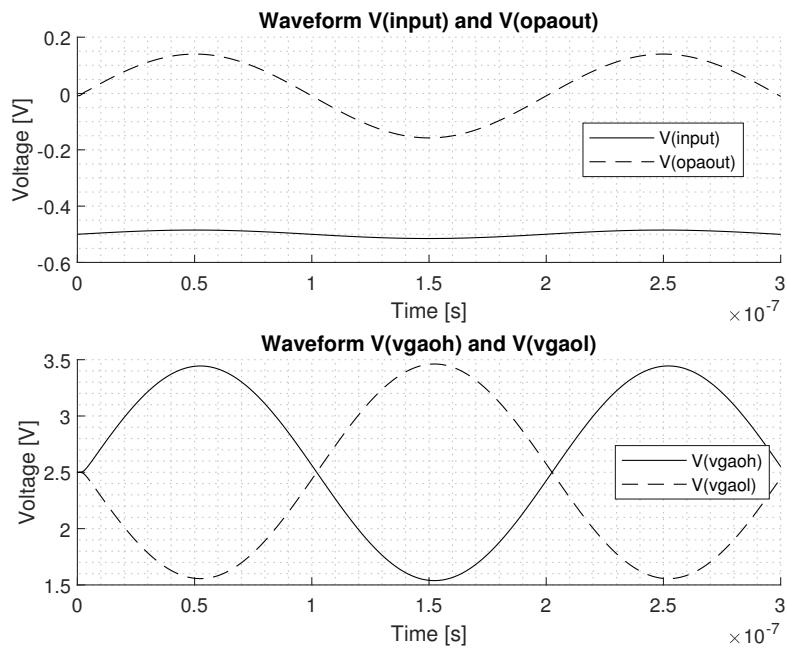


Figure 3.13: LTspice simulation output of preamplifier *LNA* and *VGA* from fig. B.2

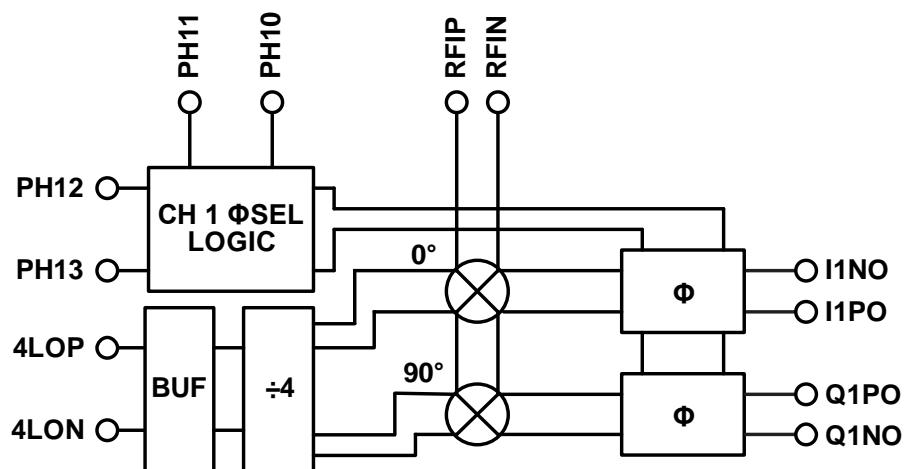


Figure 3.14: Block diagram of demodulator AD8333 [42]

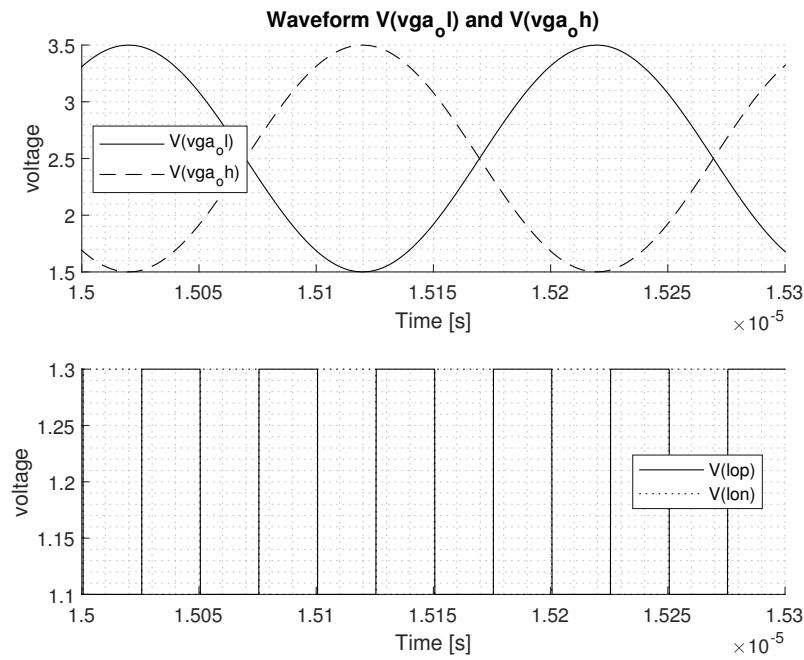


Figure 3.15: LTspice simulation demodulator input variables from fig. B.3

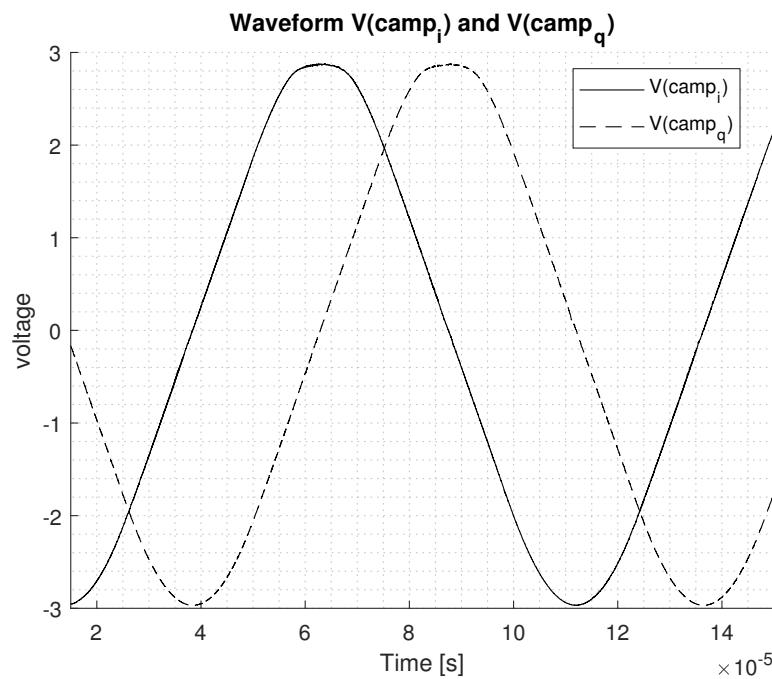


Figure 3.16: LTspice simulation demodulator output variables Q and I voltages from fig. B.3

3.8 Sample and Hold

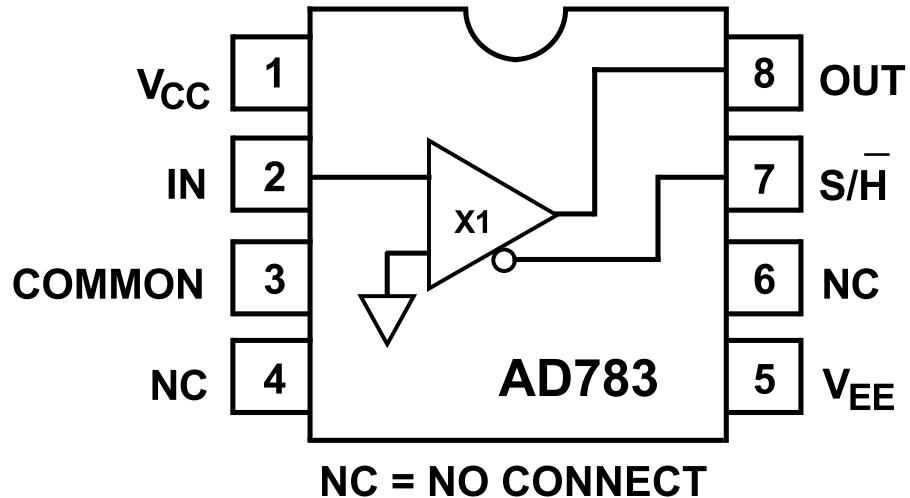


Figure 3.17: AD783 Sample and Hold Amplifier functional block diagram [34]

In this system, the sample-and-hold amplifier is necessary to keep values between each sample line. In chapter 2, it was described how the pulsed-wave flow-meter measures the movement of scatterers by sampling the back-scattered signal at a specific depth. Generally, the intended use for this part is in general data acquisition systems such as an *ADC*. In that application, the sample-and-hold amplifier captures an analogue signal and retains it during certain operations, usually analogue-to-digital conversion. Through a S/H input, two possible modes are selected, *sample* or *hold*. During the sample mode of operation, the output of the sample-and-hold amplifier follows the input. During the hold mode of operation, the output may not change by more than 1 least significant bit (*LSB*). The typical usage of a SHA is to keep the ADC input constant throughout the conversion process. With some types of ADCs, but not all, the input cannot change by more than 1 *LSB* during the conversion, or else the process will be compromised. This can either impose very low-frequency limits on such ADCs or necessitate their use with a SHA to hold the input during each conversion. An internal capacitor forms the key component of the sample-and-hold amplifier, which serves as the energy storage device. The input amplifier buffers the input signal by presenting a high impedance to the signal source while providing current gain to charge the hold capacitor. In the sample mode, the voltage on the hold capacitor follows the input signal, albeit with some delay and bandwidth limitations. In the hold mode, the switch is opened, and the capacitor retains the voltage present before being disconnected from the input buffer. The output buffer prevents the held voltage from discharging too soon by offering a high impedance to the hold capacitor. The switching circuit and its driver work together to enable the SHA to alternate between sample and hold modes.

In the pulsed-wave flowmeter, the sample-and-hold amplifier is used to keep each sample value between each gate pulse. This is done for both the I and Q signals in parallel. A diagram of a sample-and-hold operation can be seen in fig. 3.18.

3.9 Active Filter, DC-Coupling

Finally, the signal should be DC coupled for sampling and filtered using an active *HP* filter to remove unwanted PRF frequency and wall frequency. For generating a DC bias voltage,

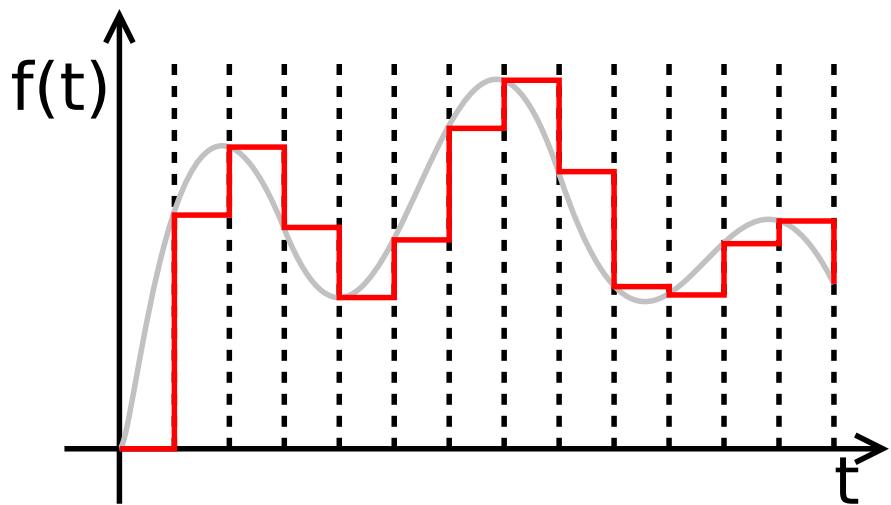
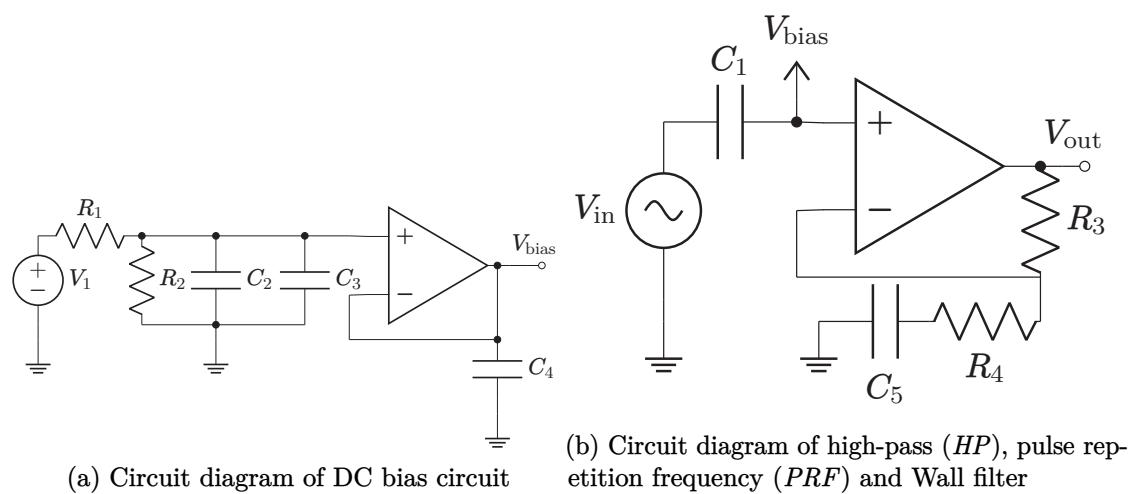


Figure 3.18: Sample and Hold function with input function $f(t)$ over time [62]



the circuit in fig. 3.19a is used. Here, a voltage divider is coupled with some stabilising capacitors and a voltage follower to output a stable output. In the circuit diagram in fig. 3.19b, a *HP* filter is used as a combined *PRF* and Wall-filter in conjunction with a 2 dB gain amplification to maximize the dynamic range of the *ADC* measurements from 0 V to 3.3 V. In the diagram, $V_{bias} = 1.65$ V, which is half of the *ADC* dynamic range of 0 V to 3.3 V, and v_{in} is the input signal to the filter from the output of the sample-and-hold amplifier. To get a 1.65 V DC-bias, a voltage-follower op-amp configuration is used with a voltage divider input on the supply voltage of 5 V, outlined in eq. (3.2).

$$V_{dc} = V_{cc} \cdot \frac{R_2}{R_1 + R_2} = 5 \text{ V} \cdot \frac{5 \text{ k}\Omega}{10 \text{ k}\Omega + 5 \text{ k}\Omega} = 1.66 \text{ V} \quad (3.2)$$

For the 2 dB gain, a non-inverting amplifier configuration is used, expressed by eq. (3.3).

$$G_{\text{dB}} = \frac{V_{\text{out}}}{V_{\text{in}}} = 20 \cdot \log_{10} \left(1 + \frac{R_3}{R_4} \right) = 20 \cdot \log_{10} \left(1 + \frac{40 \text{ k}\Omega}{150 \text{ k}\Omega} \right) = 2.05 \text{ dB} \quad (3.3)$$

A SPICE simulation was implemented and can be seen in fig. B.4. From this simulation model, a small signal analysis as well as a transient analysis was conducted. The resulting small signal analysis can be seen in fig. 3.20 and the transient analysis can be seen in fig. 3.21 and confirms the expected result.

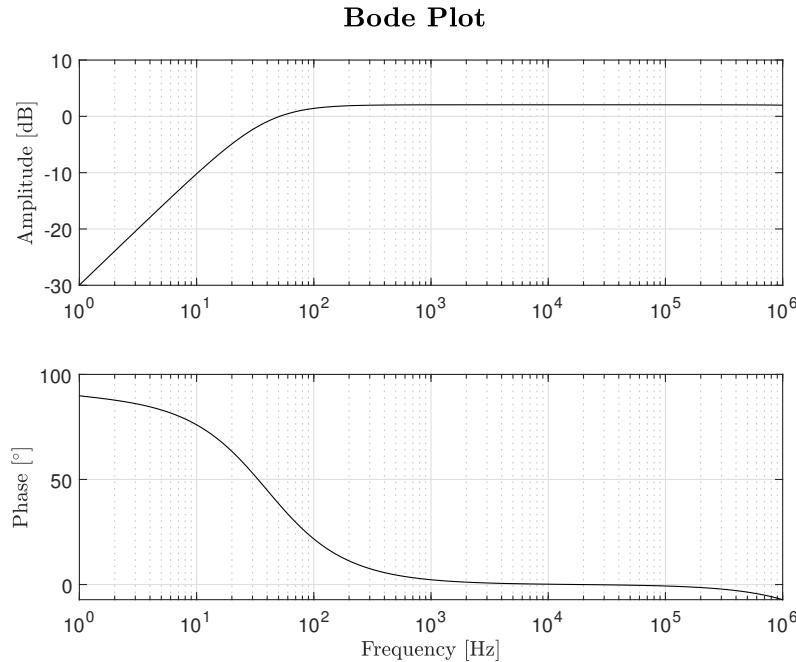


Figure 3.20: Small-signal analysis of DC-Coupling filter circuit

3.10 Digital Signal Processor

In the *DSP* system, the function is to turn a waveform into a humanly readable velocity through performing an fast Fourier transform (*FFT*) of the input signal captured from the output of the active filter of the previous section. Since *DSP* devices are programmable *DC* devices of typically 3.3 V to 5 V, it is vital that the input signal is *DC*-coupled. Since the input signal is relatively low frequency, most *ADC* interfaces should be sufficient to capture a frequency window of less than 10 kHz. cleardoublepage

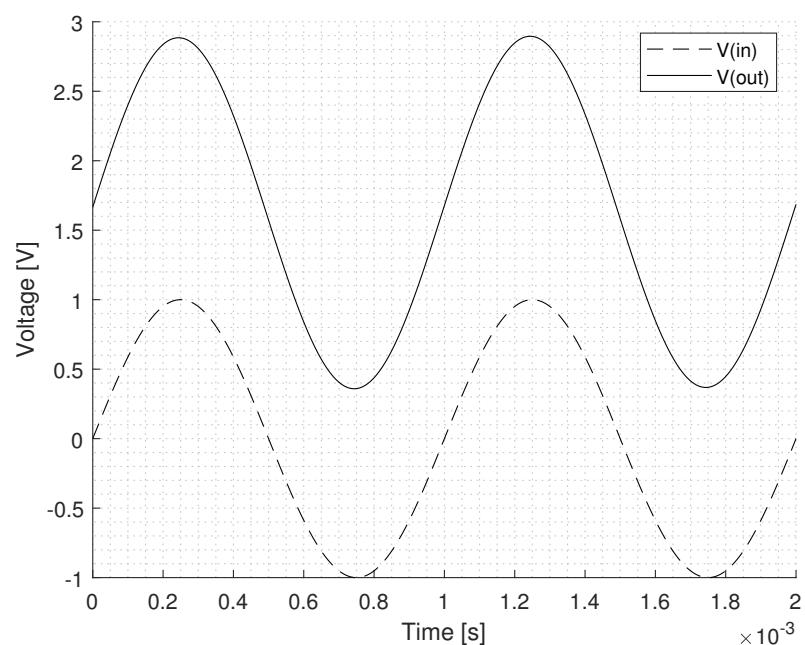


Figure 3.21: Transient analysis of DC-Coupling filter circuit

4 Implementation

In this chapter, the steps involved in turning a theoretical design into a tangible system will be outlined. Since the synthesis chapter dealt with an explanation of the functions of each module and simulations, with a subsequent evaluation of the outcomes, this chapter will focus on the creation of physical hardware implementations and reproducing the expected results.

4.1 Control System

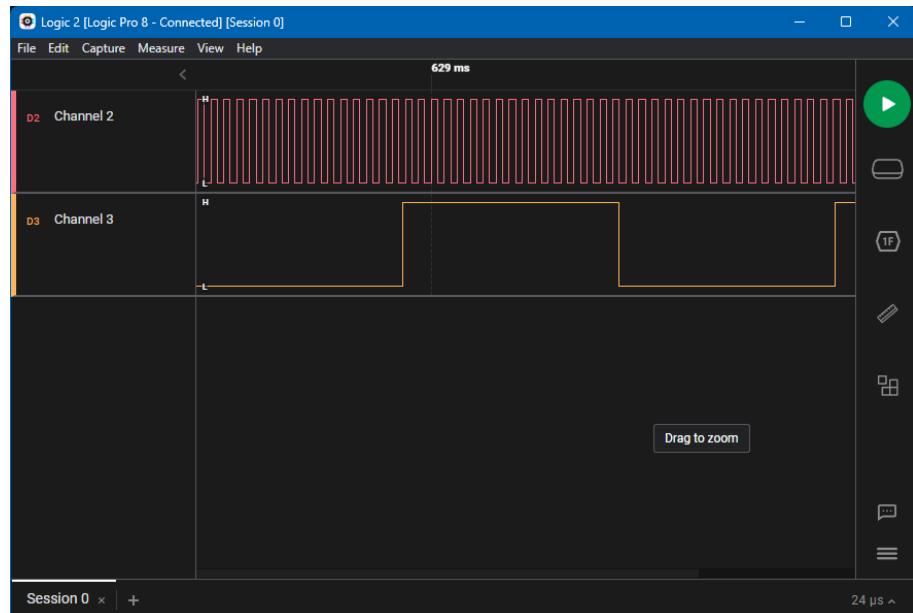


Figure 4.1: STM32 Zephyr RTOS pulser output

During the initial implementation stage of the control system, a bring-up of the STM32F411RE board and several pulse signals were successfully generated. In fig. 4.1, two pulse signals can be seen. The experimental microcontroller PWM generator code can be seen in listing A.1. In Channel 2, 5 MHz ultrasound pulse can be seen. In Channel 3, the 10 kHz PRF signal can be seen. Unfortunately, soon thereafter it was discovered a limitation of the API in Zephyr is not mature enough developed for power systems such as the half-bridge in the transmitter circuit. In more practical terms, it was not possible to generate two complementary signals with dead-time using the existing Zephyr PWM API. To continue with that solution, a new PWM driver would have to be written from scratch, which is no trivial task. Alternative solutions were investigated. Another option was to use the hardware abstraction layer (*HAL*) provided by the manufacturer of the microcontroller, but this would also mean an increased amount of development time for the control system since the *HAL* is rudimentary in implementation and has little abstraction. However, after finding inspiration [53], it was decided to try the alternative system PYNQ-Z1, which is a development board by Digilent. On the PYNQ Z1 board is a Zynq 7000 system-on-a-chip (*SoC*). Inside the Zynq 7000 SoC there are both an *FPGA* and Arm based processor. PYNQ is an open-source framework that runs on Xilinx compute platforms where higher levels of abstraction enable faster productivity.

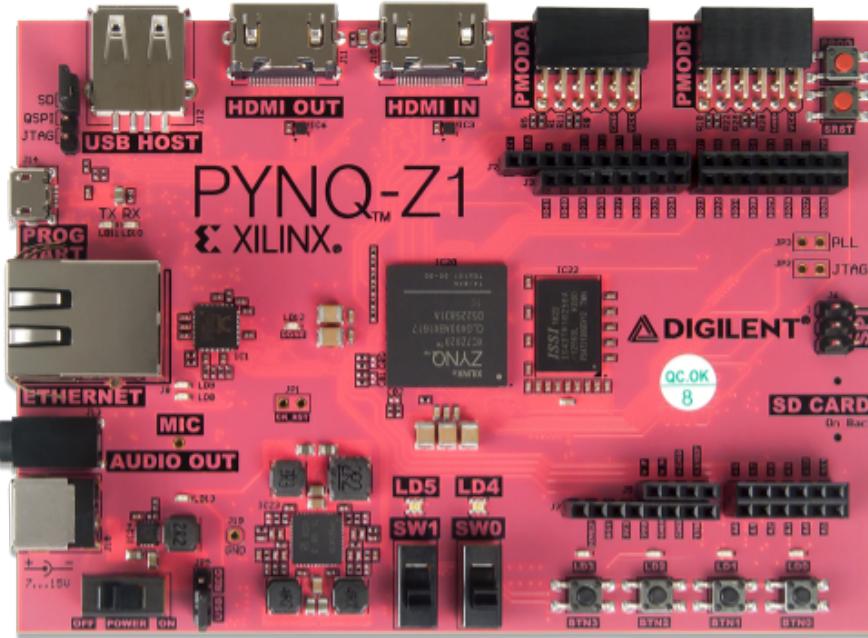


Figure 4.2: PYNQ-Z1 development board

Seen in fig. 4.2 is the development board with its peripherals. The development of a prototype pulser system will be done by implementing an FPGA project in Vivado and then generating the bitstream. After this, the FPGA artifacts are generated as a **.bit** and **.hwh** file. These two files are used in the JupyterLab environment as an overlay to configure the logic of the FPGA and output signals on the PMOD-A connector.

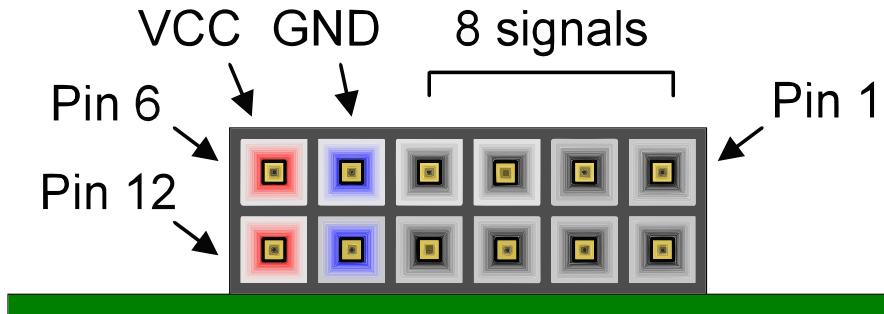


Figure 4.3: PYNQ Z1 PMOD port diagram [46]

On the PYNQ-Z1 board, the PMOD ports are 12-pin female connectors with 0.1 inch spacing that connect to normal 12-pin headers. As illustrated in fig. 4.3, each 12-pin PMOD port offers two 3.3V VCC signals (pins 6 and 12), two GND signals (pins 5 and 11), and eight logic signals. Each PMOD port on a PYNQ board is classified as normal, MIO linked, XADC, or high-speed. The PYNQ-Z1 features two PMOD ports, both of which are high-speed. For maximal switching rates, the High-speed PMOD ports route their data signals as impedance matched differential pairs. For further protection, they feature pads for loading resistors, however the PYNQ-Z1 ships with these loaded as 0Ω shunts. With the series resistors shunted, these PMOD ports provide no short-circuit protection but allow for substantially quicker switching rates. Pins 1 and 2, pins 3 and 4, pins 7 and 8, and pins 9 and 10 are coupled to neighboring signals in the same row. Traces are routed in a 100Ω

($\pm 10\%$) differential configuration. Coupled pairs may produce crosstalk if pins on this port are utilized as single-ended signals. Because it is being used as a single-ended signal in this application, one of the pins is grounded and its pair is utilized for the single-ended signal.

```

1 set_property IOSTANDARD LVCMOS33 [get_ports PULSE]
2 set_property IOSTANDARD LVCMOS33 [get_ports PWM]
3 set_property IOSTANDARD LVCMOS33 [get_ports PWMN]
4 set_property IOSTANDARD LVCMOS33 [get_ports GATE]
5 set_property IOSTANDARD LVCMOS33 [get_ports PRF]
6 set_property IOSTANDARD LVCMOS33 [get_ports CLK]
7 set_property PACKAGE_PIN Y16 [get_ports PULSE]
8 set_property PACKAGE_PIN Y18 [get_ports PWM]
9 set_property PACKAGE_PIN Y19 [get_ports PWMN]
10 set_property PACKAGE_PIN Y17 [get_ports GATE]
11 set_property PACKAGE_PIN U18 [get_ports PRF]
12 set_property PACKAGE_PIN U19 [get_ports CLK]
13
14 set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[3]}]
15 set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[2]}]
16 set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[1]}]
17 set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[0]}]
18 set_property PACKAGE_PIN R14 [get_ports {LEDs[0]}]
19 set_property PACKAGE_PIN P14 [get_ports {LEDs[1]}]
20 set_property PACKAGE_PIN N16 [get_ports {LEDs[2]}]
21 set_property PACKAGE_PIN M14 [get_ports {LEDs[3]}]
```

Code 4.1: Constraints on Pulse Generator and Control System

When programming an FPGA with software like as Xilinx's Vivado, it becomes necessary to inform the system which physical pins on the FPGA correspond to the FPGA ports defined in the VHDL code. This is quite similar to putting a register high or low on a microcontroller to turn an LED on or off, operate a clock, or function as a data line. However, with a microcontroller, many of these pins are “hard-wired” in the sense that they cannot be relocated to a physically different pin on the microcontroller. In general, it's just not an option. This is not the case with an FPGA; instead, the hardware interface is established in VHDL and then the appropriate inputs and outputs on that interface are constrained to whichever pins on the FPGA are required, making FPGAs incredibly versatile for complicated and bespoke designs. In listing 4.1, the constraints can be seen setting the port names to a certain IOSTANDARD and voltage level and the pin name PACKAGE_PIN. All the described pins are *I/O* available on the PMOD port of the PYNQ-Z1.

Based on inspiration found on tutorials for PWM generators on PYNQ platforms [53], [60], a prototype of an ultrasound pulser is developed by implementing a pulse-width modulation (*PWM*) generator for the complementary PWM output and a signal controller to control pulse timings. A block diagram of the ultrasound pulser system can be seen in fig. 4.5. After that, an interface is developed to enable the control system to take input from a programmable Arm processor.

4.2 Power Stage

A picture of the power stage PCB can be seen in fig. 4.7. An experiment is conducted to validate the function of the power stage. Using the jumpers, the PCB is configured without its onboard load, and a *PZT* transducer is attached with a splitter adapter to connect the other side to an oscilloscope for data acquisition.

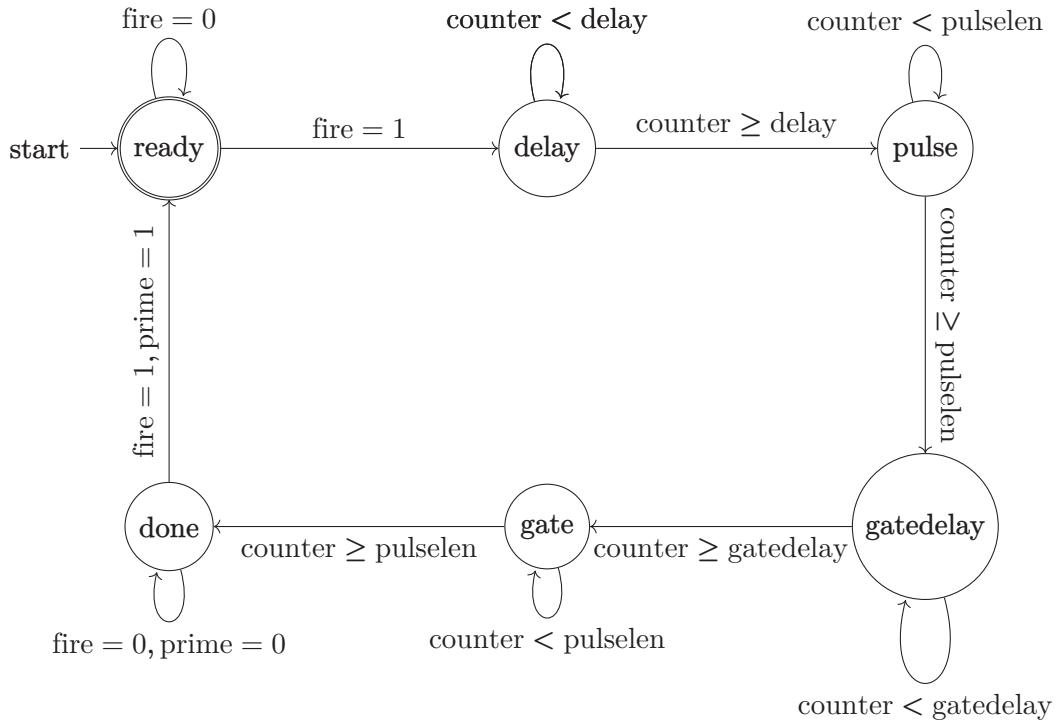
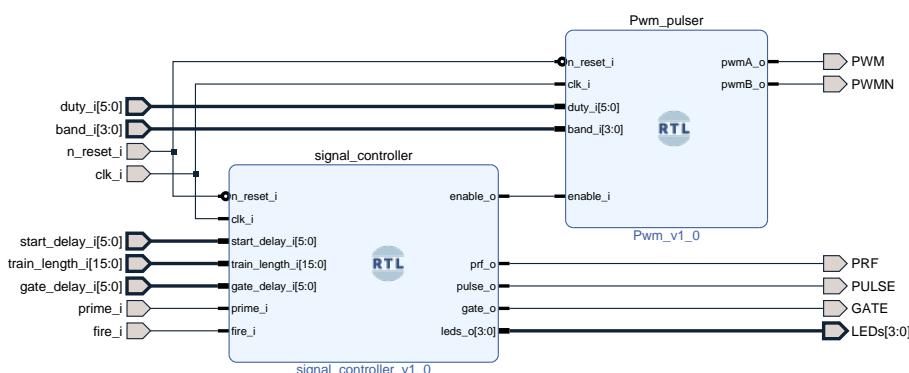
Figure 4.4: *FSM* diagram of ultrasound pulser signal controller

Figure 4.5: PWM pulser and signal controller as a block diagram with inputs and outputs

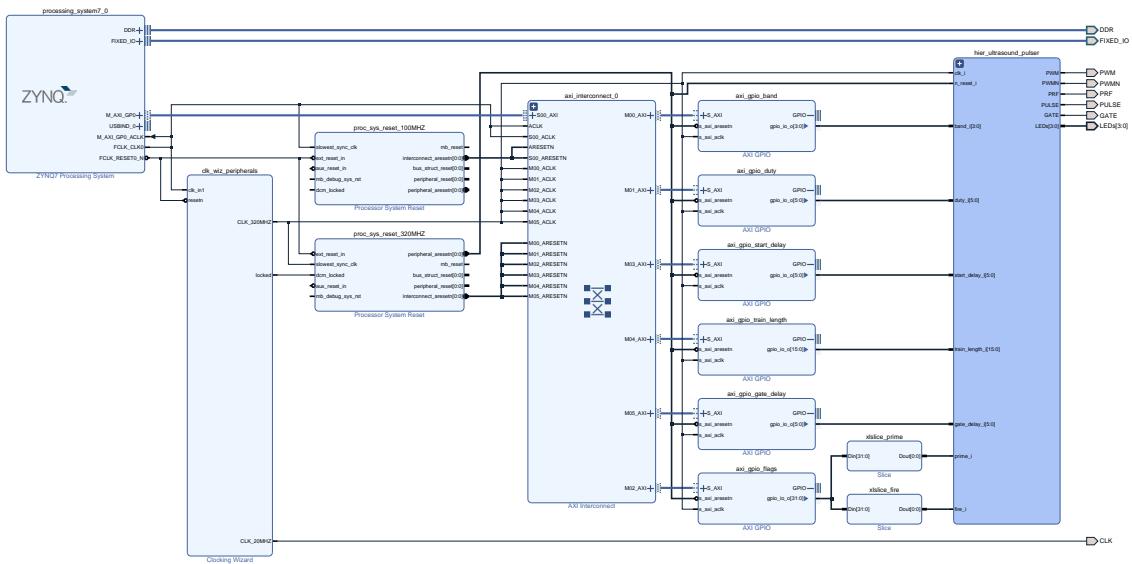


Figure 4.6: Top level block diagram of the FPGA overlay with AXI interconnects and registers

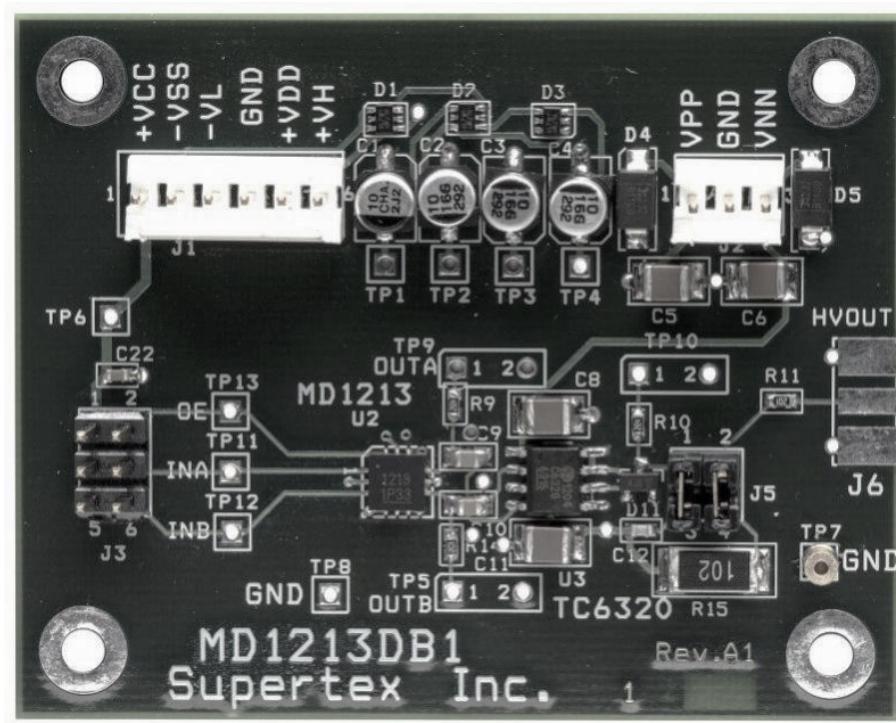


Figure 4.7: MD1213DB1 High Speed Pulser

```

1 architecture arch of Pwm is
2     signal timer_r      : natural range 0 to 2**duty_i'length-1;
3 begin
4     clocked: process(clk_i)
5     begin
6         if rising_edge(clk_i) then
7
8             -- sync reset
9             if n_reset_i = '0' then
10                 pwmA_o    <= LO;
11                 pwmB_o    <= LO;
12                 timer_r <= 0;
13             else
14
15                 -- timer
16                 timer_r <= timer_r + 1;
17                 pwmA_o    <= LO;
18                 pwmB_o    <= LO;
19                 if enable_i = '0' then
20                     pwmB_o    <= HI;
21
22                     timer_r <= 0;
23                 else
24                     -- output a
25                     if timer_r <= unsigned(duty_i) and timer_r
26                         >= unsigned(band_i)  then
27                         pwmA_o <= HI;
28
29                     end if;
30                     -- output b
31                     if timer_r > to_integer(unsigned(band_i)) +
32                         to_integer(unsigned(duty_i)) then
33                         pwmB_o <= HI;
34                     end if;
35
36                     end if; -- enable
37             end if; -- sync reset
38         end if; -- rising_edge
39     end process clocked;
40 end architecture;

```

Code 4.2: PWM Pulse Generator Descriptor

4.3 Transmit/Receive Switch

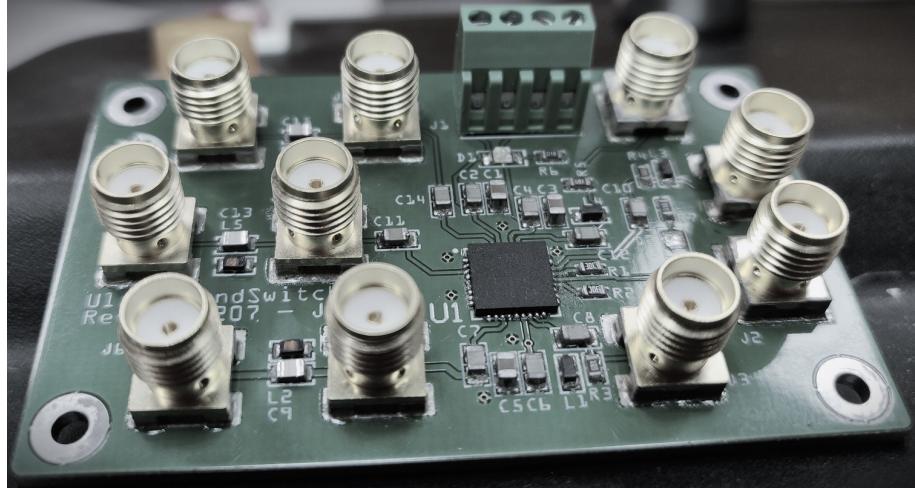


Figure 4.8: Transmit/Receive Switch after assembly

The entire schematic of the transmit/receive switch can be found in the appendix in figs. C.8 and C.9. As mentioned in the previous chapter, a PCB layout was made and a batch of 5 was ordered with an accompanying stencil for fast assembly. After the PCBs arrived, the stencil was mounted in the stencil frame and the PCB was aligned for solder paste application. After the solder paste application is completed, all the components are placed on their corresponding footprints and the PCB is placed in the reflow oven. The equipment used in this process is listed in table E.1. The finished assembly can be seen in fig. 4.8.

4.4 Band-pass Filter

Since the *BP* filter is comprised of a module component in a bespoke form factor, a circuit is implemented on a prototyping board. With the BPF-C4R5+ mounted in the center, input and output connectors are placed on either side with SMA connectors. Since the filter is passive, no power connectors are needed. The prototype board of the *BP* filter can be seen in fig. 4.9.

4.5 Preamplifier

Before the signal can be demodulated, it must be DC-biased and amplified. This is what the preamplifier is for. For the preamplifier, the circuit is validated using an experiment where a function generator is transmitting a low amplitude sine with ac-coupling and measure the amplified dc-coupled output. Using the AD8332 built-in *LNA* and bypassing the *VGA* in the circuit, it feeds the amplified output to the next subcircuit Quadrature Demodulator. The preamplifier is part of the same board as the quadrature demodulator in section 4.6.

4.6 Quadrature Demodulator

As described in the previous chapter, the demodulator use an I/Q quadrature demodulation scheme to take two differential RF signals and a quadruple frequency signal, in this case, 5 MHz and 20 MHz, respectively, and determines the frequency difference between the

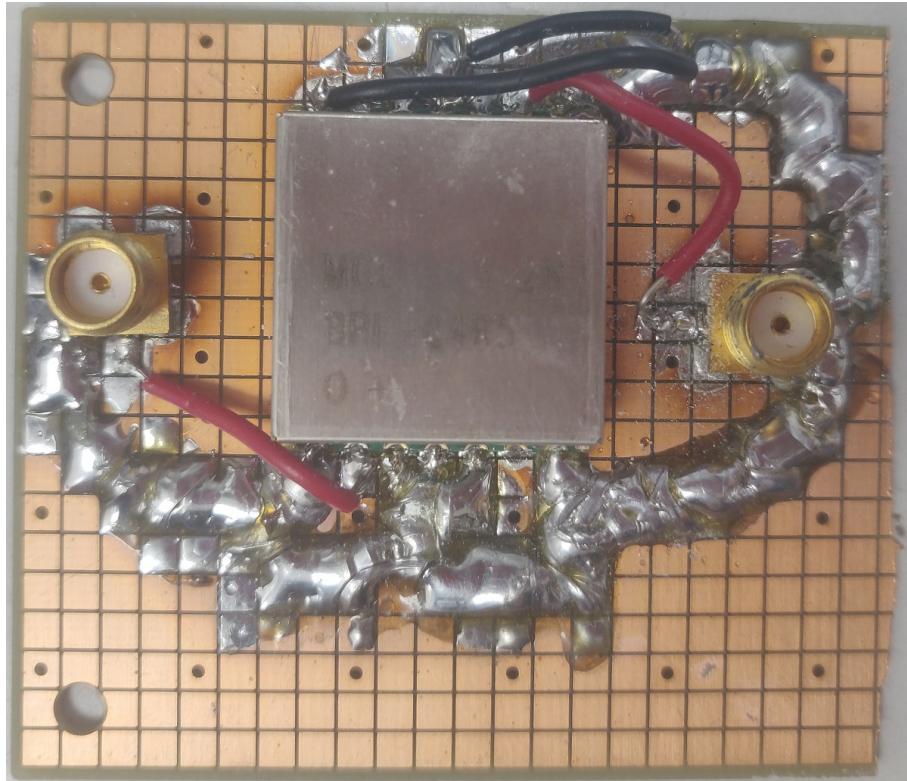


Figure 4.9: Prototype board of the Band-Pass filter

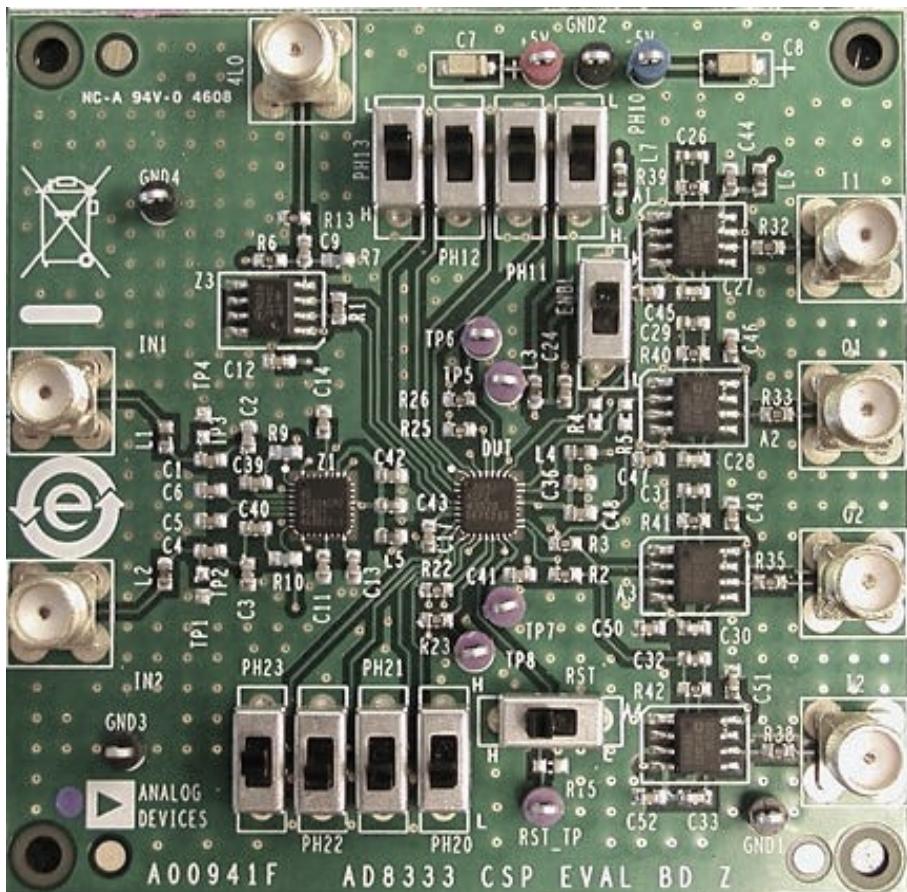


Figure 4.10: Demodulator PCB AD8333-EVALZ

fundamental frequency and the Doppler frequency on the output. After the differential signal from the preamplifier is demodulated, it is output as a current. Next, it is converted using the AD8021 current-to-voltage amplifier coupled with an active low-pass (*LP*) filter to remove the summed demodulated frequency component. What remains are the low-frequency *I* and *Q* signals in the kHz range. The entire schematic of the demodulator can be found in the appendix in fig. C.11.

4.7 Sample and Hold

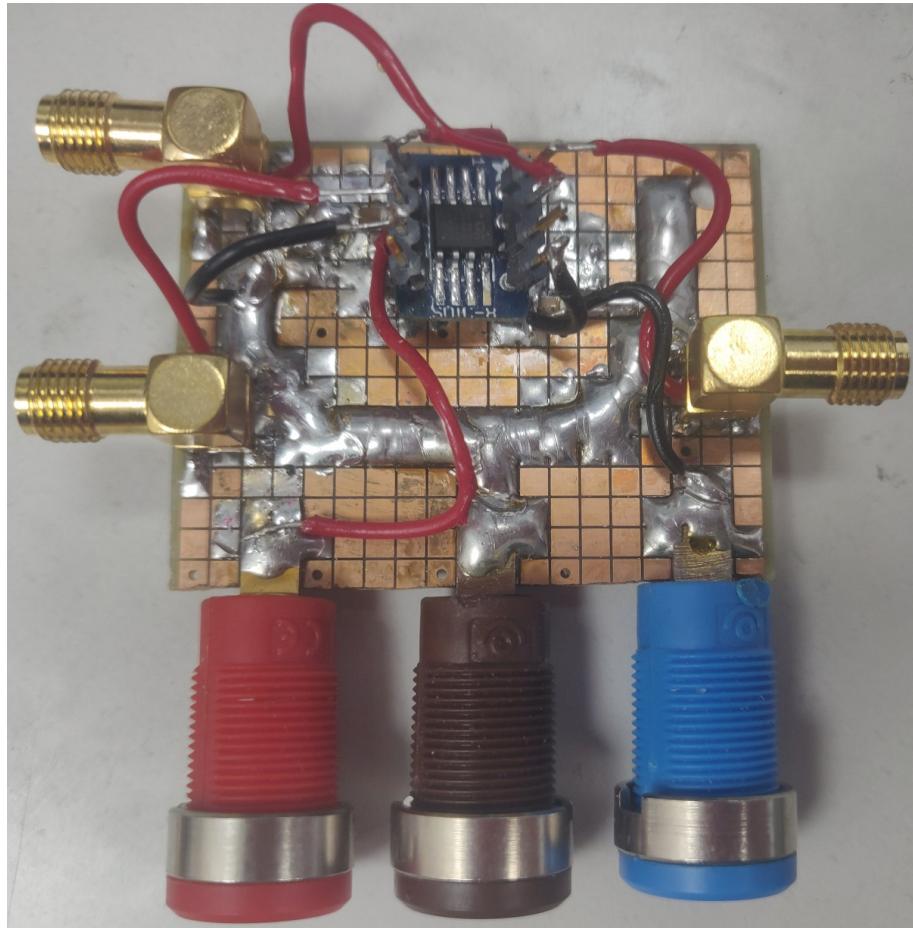


Figure 4.11: Prototype board of the Sample and Hold amplifier

After each demodulated burst is sampled, between each sample line pulse repetition it is desired to hold the voltage, so the analogue-to-digital conversion that may be running asynchronously does not sample zero-values between the bursts. After the low-frequency signals are output from the quadrature demodulator, they are sampled using the GATE signal output from the control system. This GATE signal is a delayed pulse equal to the length of the pulse train and determines the sampling depth of the AFE. The prototype board of the sample and hold amplifier can be seen in fig. 4.11.

4.8 Active Filter, DC-Coupler

Finally, before the signal is digitally quantized by the *ADC*, it has to pass through a *HP* filter to remove undesired *PRF* or Wall frequency components. The implementation is done on a prototyping board and can be seen in fig. 4.12.

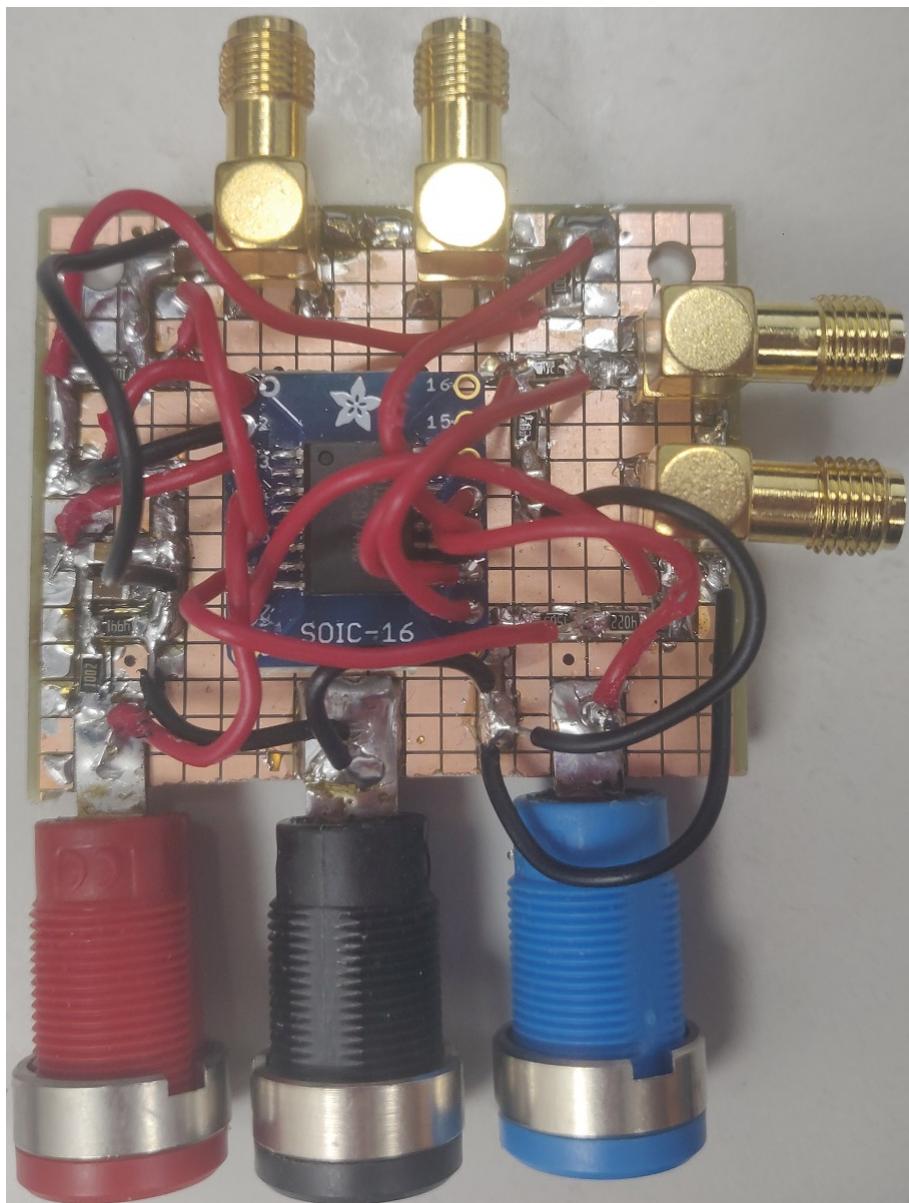


Figure 4.12: Prototype board of the active filter and *DC*-coupler

4.9 Digital Signal Processor

5 Analysis

In this chapter every subcomponent will be part of an experiment in isolation to validate their function. Also, various test setups will be discussed for the purpose of making multi-module experiments in the AFE and control system to evaluate their performance and compatibility. As such, each module will be tested independently to validate its function before a larger, more comprehensive experiment is conducted. All figures in this chapter show measurements performed on physical hardware.

5.1 Validation experiments

5.1.1 Control System

First, a bitstream was generated with a Pwm generator and AXI interconnect registers. An AXI interconnect register is a method to create GPIO interfaces between the logical processor and the Arm processor on the Zynq 7000 SoC using registers that temporarily store GPIO data in the bitstream overlay. A block diagram of the overlay can be seen in fig. A.2 and the JupyterLab notebook can be seen in fig. A.1.

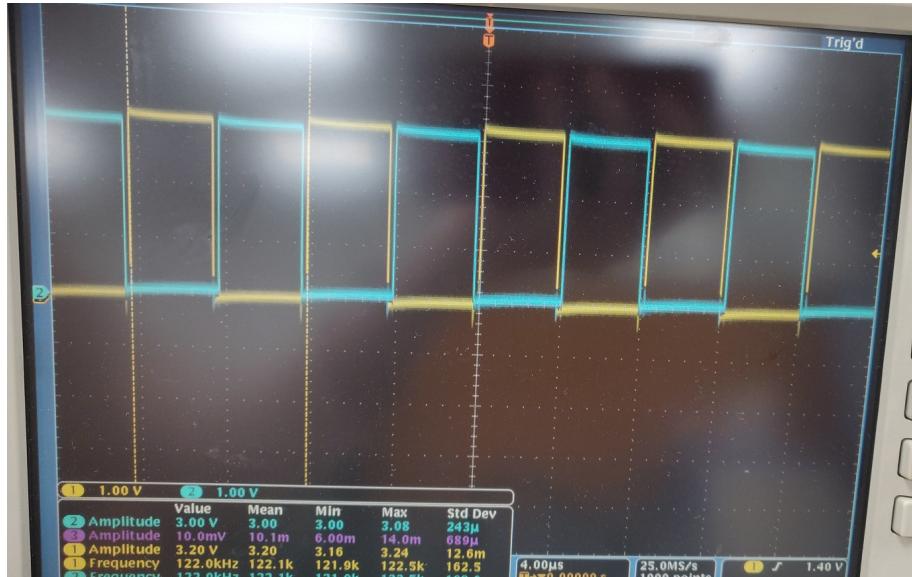


Figure 5.1: Complementary PWM output with Pynq Z1 FPGA and JupyterLab notebook

A preliminary implementation was implemented in Vivado and JupyterLab for a continuous complementary PWM controller. The resulting measurement can be seen in fig. 5.1 before a clock configuration, which means the frequency corresponded to an arbitrary temporary pulse frequency of 122 kHz. As the initial test proved successful, the continued development and maturation of the pulser enabled a complete signal generator for the ultrasound pulse generator.

Seen in fig. 5.2 are the measured signals from the pulse generator showing the various timings of each signal captured with a Salae Logic Analyzer Pro 8.

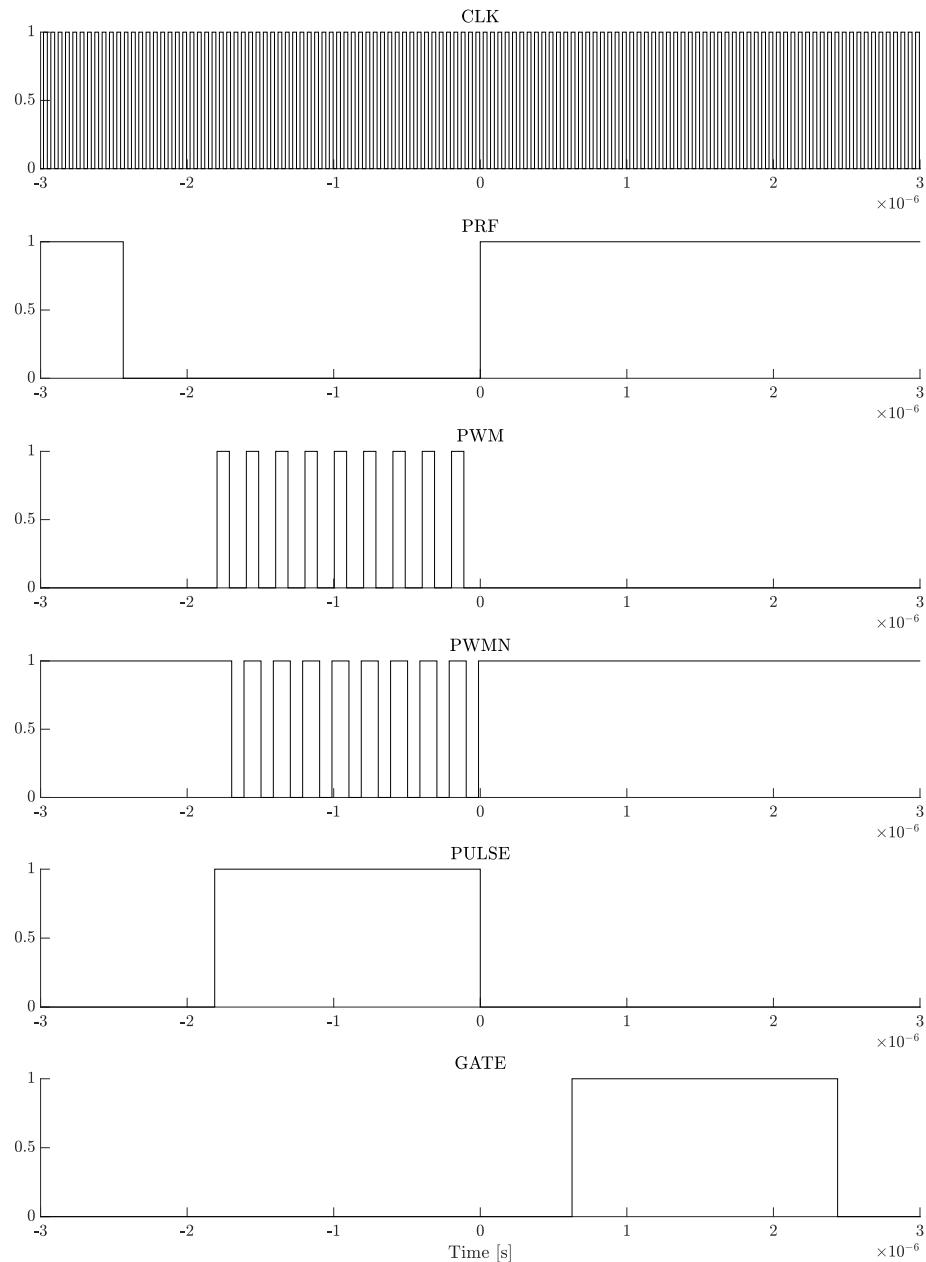


Figure 5.2: Captured timing diagram of the control system pulser

5.1.2 Power Stage

Seen in fig. 5.3 are actual measured inputs and outputs of the power stage. On the input, there are two complementary 5 MHz signals with varying duty cycle to generate the desired dead time. On the output, we see the rail-to-rail push-pull operation of the *MOSFET* half-bridge. The schematic of the transmitter can be found in the appendix in fig. C.10. Noticeable noise is observed in the input signal top and base but is negligible for successful operation. Possibly, the noise is due to a cable and adapter setup.

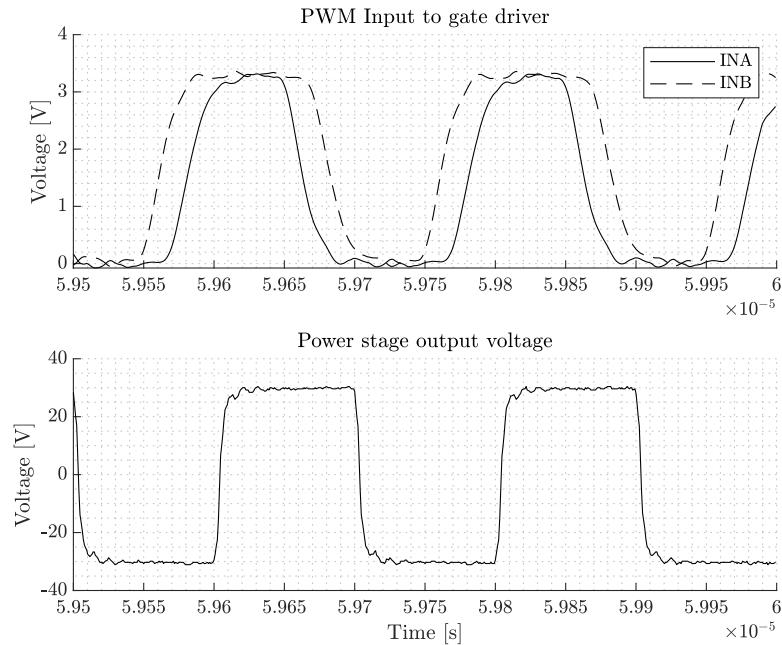


Figure 5.3: Measured input and output of power stage PCB. (Above) Input to gate driver with dead-time (Below) Output of MOSFET half-bridge and the voltage across the load

5.1.3 Transmit/Receive Switch

For validating the TX/RX switch, an experiment is conducted with a *PZT* transducer, water tank, function generator and an oscilloscope. Using two input signals, $f_{\text{prf}} = 10 \text{ kHz}$ switch signal, and $f_0 = 5 \text{ MHz}$ burst mode transmit signal, the switch is configured to transmit and receive. A picture of the submerged transducer with a reflector can be seen in fig. 5.4. After submerging the transducer in distilled water and measuring on the receiver side of the TX/RX switch, a reflected signal from the tank can be observed in fig. 5.5.

5.1.4 Band-pass Filter

it is desired to validate its frequency response to determine if it functions as desired. To obtain the frequency response, a bode plot of the magnitude and phase is measured from 300 kHz until 20 MHz using a vector network analyzer (*VNA*) in a S21 configuration, meaning a measurement of the output in respect to the input. This measurement determines the difference in magnitude and phase of the output in comparison with the input signal. Observed in fig. 5.6 is the frequency response of the band-pass filter measured on a *VNA*. It is noted that the pass band frequencies are mostly as expected with -0.5 dB frequencies at 1.5 MHz and 7 MHz. Though, the roll-off in the higher stop band appears somewhat lower than in the lower stop band. That would mean that it is plausible that higher frequency noise components are retained in the output than in the lower stop band. For the phase, it



Figure 5.4: TX/RX Switch reflection experiment with water tank

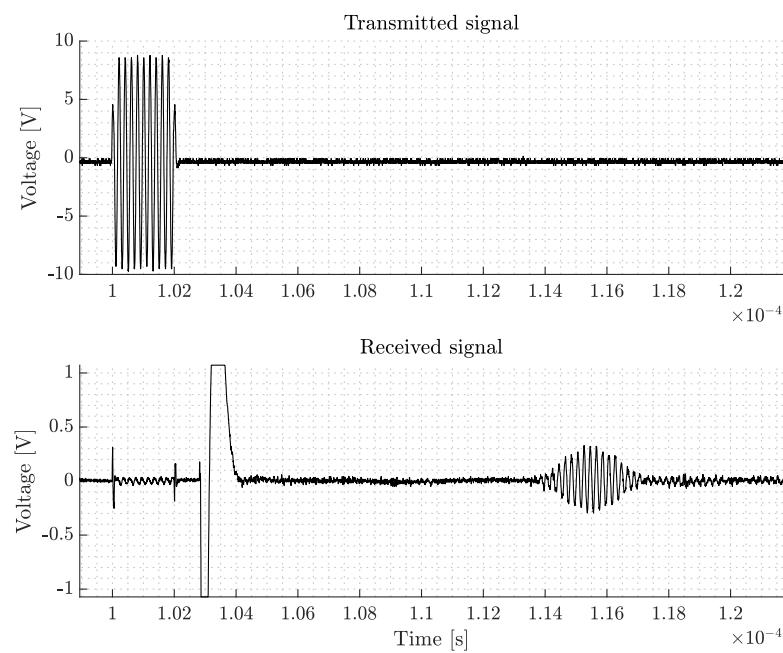


Figure 5.5: Measured transmit and receive on Transmit/Receive Switch PCB (Above)
Measured transmit voltage (Below) Received reflected signal off water tank

seems to have a significant phase delay, going from around 100° to 250° from the start to the end of the pass band.

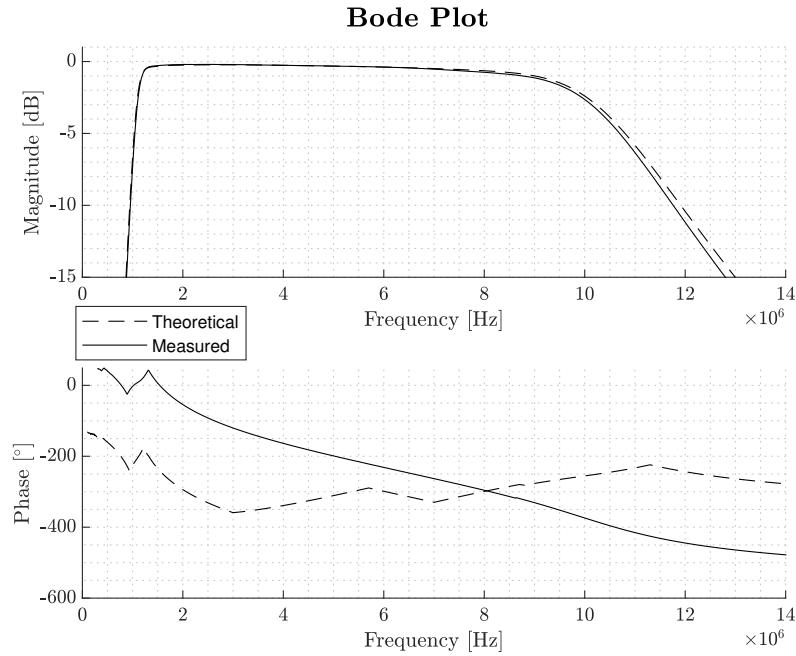


Figure 5.6: Band-pass Filter bode plot from 0.3 MHz to 14 MHz with (above) magnitude and (below) phase

5.1.5 Preamplifier

Seen in fig. 5.7 are measurements of the preamplifier circuits showing a 70 mV input signal and a 300 mV output signal with a 2.5 V DC bias. In this application, however, only the *LNA* is used, and the *VGA* is bypassed in the hardware preamplifier configuration. The schematic of the preamplifier circuit is part of the demodulation schematic and can be found in the appendix in fig. C.11.

5.1.6 Demodulator

An experiment is conducted with the sample-and-hold amplifier to verify the functionality. A low-frequency I-Q simulated signal is created from the function generator with a sample gating pulse train to control the sample-and-hold function. Seen in fig. 5.8 are the input signals, differential signals of 5.001 MHz and 20 MHz local oscillator signal. Seen in fig. 5.9 are the differential input signals *A* and *B* and the demodulated output signals *I* and *Q*, where the phase between *I* and *Q* denotes the Doppler shift direction, or rather, the direction of flow of the scatterer. It is noted that the differential signal is so high frequency compared to the timescale so there has to be a zoomed in subplot where the waveform is visible to show the waveform. This highlights the observation of the Doppler shift being pushed down in frequency, which is one of the key functions of the demodulator in this system.

5.1.7 Sample and Hold Amplifier

Seen in fig. 5.10 is the measured inputs and outputs of the circuit during the experiment. Above is the I-Q input and in the middle is the sample gating, and below is the output signal. On the output signal, it is noted the corresponding voltage transients for every pulse in the gate input.

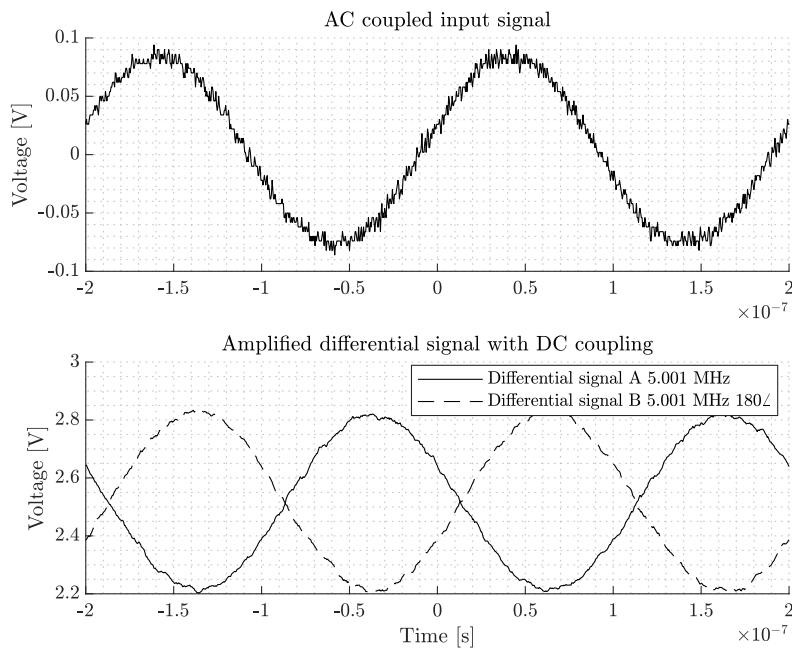


Figure 5.7: Measured input of preamplifier PCB, (Above) AC coupled input signal with amplitude 1 V (Below Measured output of preamplifier PCB, Differential signal with DC coupling and $\times 19$ dB amplification)

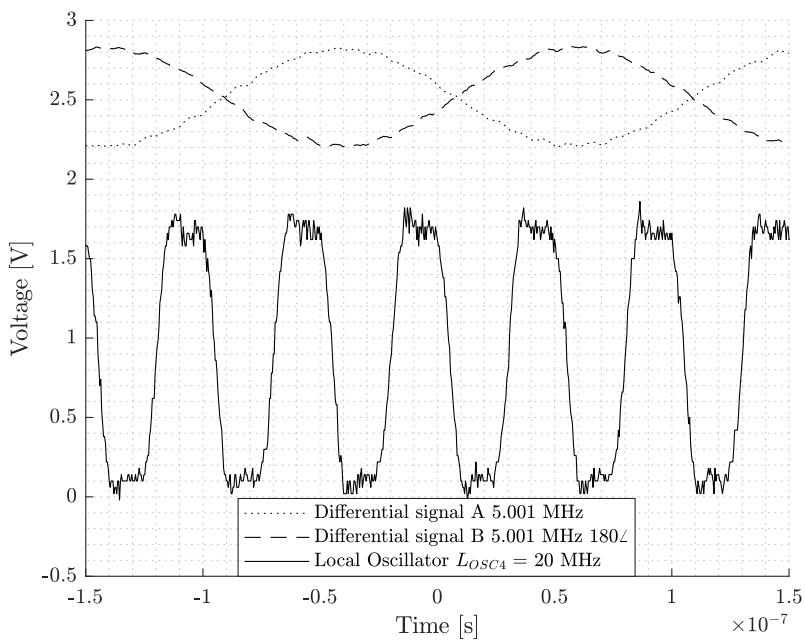


Figure 5.8: Measured input of demodulator PCB (Above) Input from received signal (Below) Input from local oscillator ($f_0 \cdot 4$)

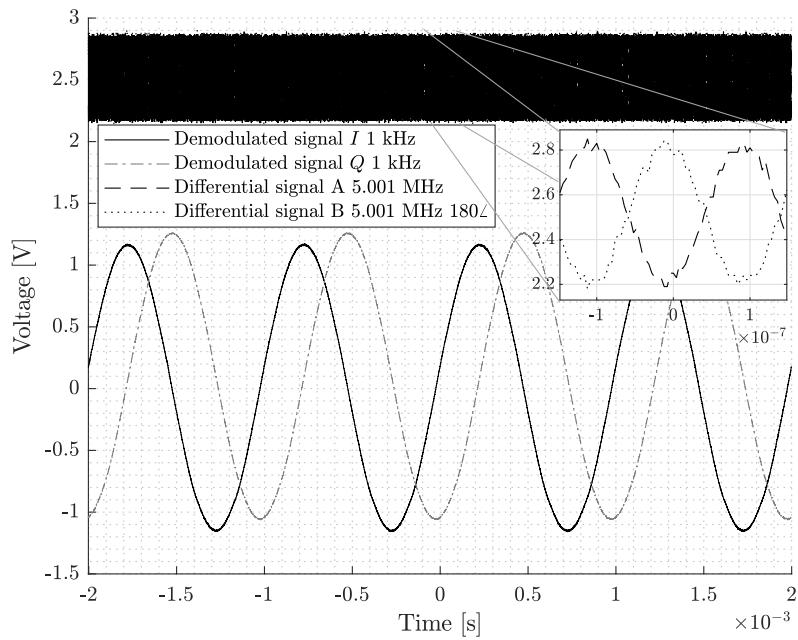


Figure 5.9: Measured output of demodulator PCB

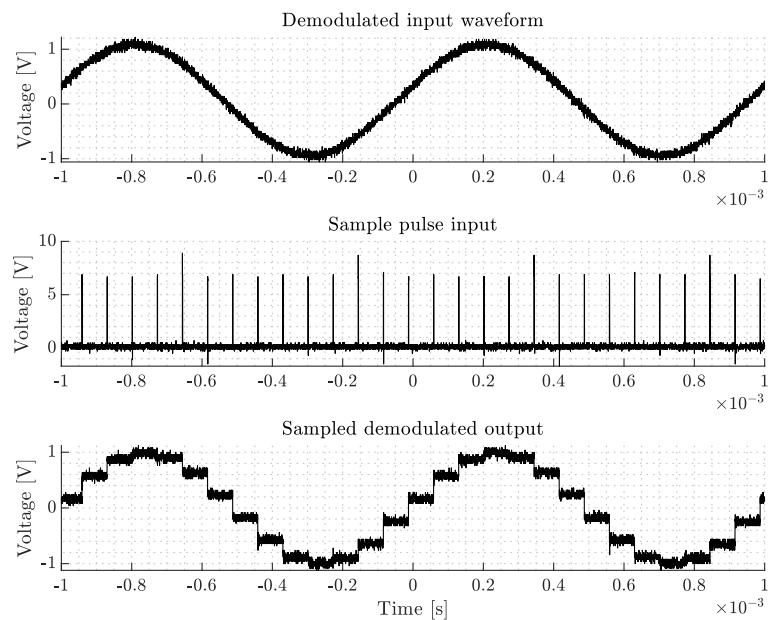


Figure 5.10: Measured input and output of Sample and Hold amplifier

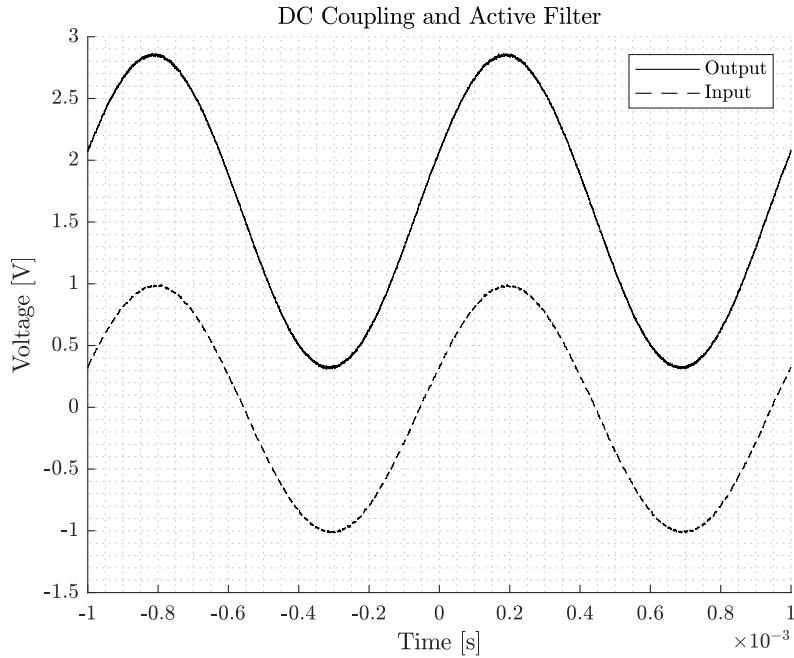


Figure 5.11: Measured input and output of Active filter and DC Coupler

5.1.8 Active Filter, DC Coupler

Seen in fig. 5.11 is the measured input and output of the DC coupler and active filter. The dashed line is the alternating current (*AC*) coupled input signal with 1 V amplitude and the solid line is the amplified *DC* coupled output signal.

5.2 Pulse Generator and Power Stage

Using the pulse generator and the power stage, an experiment is performed where the complementary PWM signals are generated and output to the gate driver of the power stage. In turn, the gate driver will drive the MOSFET pair and output a high-power output than the pulse generator can supply on its own through its general purpose *I/O* (*GPIO*).

Seen in fig. 5.12 are the measurements obtained from the pulse generator and power stage experiment. The measurements show the ability of the Pynq Z1 as a pulse generator is functioning as expected. As the power stage half-bridge is rail-to-rail, the output pulse voltages depend on the power supply voltage. In this case, the experiment is using a 30 V maximum *DC* power supply (*DCPS*), and the output peak pulse voltages are ± 30 V. However, the power stage itself is specified for operating voltages up to ± 100 V.

5.3 Doppler String Phantom Experiment

The CIRS A043 Doppler String Phantom is a device used to evaluate the performance of Doppler ultrasound imaging systems. The phantom consists of a set of strings that move in a controlled manner when exposed to ultrasound waves. The strings are made of nylon monofilament and are arranged in a parallel array.

When the phantom is scanned with a Doppler ultrasound probe, the strings vibrate in response to the sound waves, producing a Doppler signal. The phantom includes a set of control strings that vibrate at a known frequency, allowing the user to calibrate the system

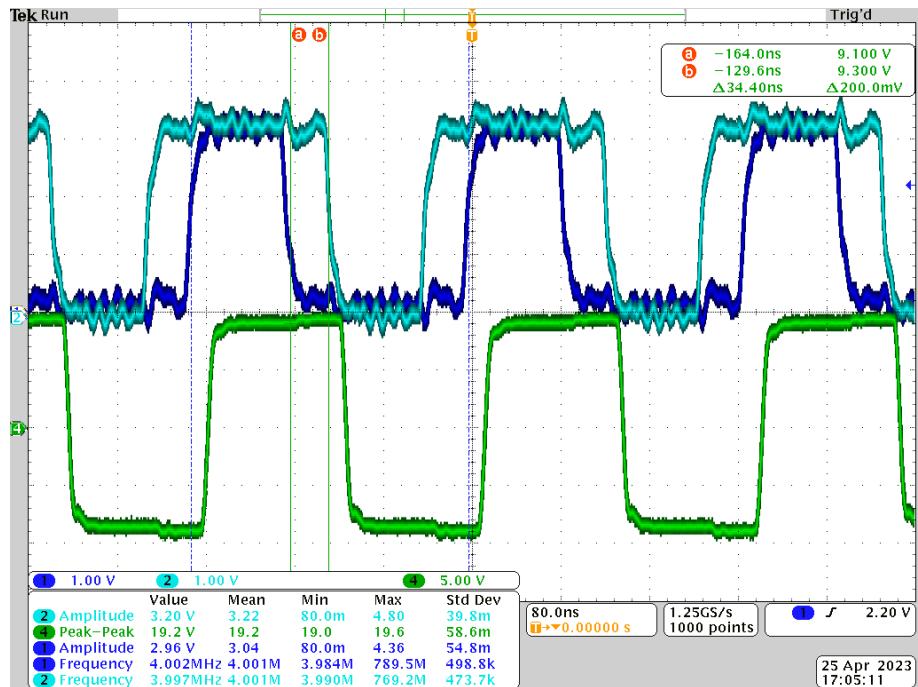


Figure 5.12: Complementary PWM output from the pulse generator and bipolar high power pulses

and determine its accuracy. The string is driven by a motor that can be programmed to move at different speeds and directions, and it is also designed to create turbulence, which mimics the flow of blood in diseased vessels.

The phantom also includes a set of strings that move in a complex, non-linear pattern, simulating blood flow in vessels with turbulent flow. This allows the user to test the system's ability to accurately detect and measure turbulent blood flow, which is an important diagnostic feature for conditions such as stenosis or arterial occlusion.

By evaluating the system's performance on the known frequencies of the control strings and the complex patterns of the turbulence strings, the user can assess the accuracy and reliability of the Doppler ultrasound system. The CIRS A043 Doppler String Phantom is widely used in research, development, and quality assurance of Doppler ultrasound equipment.



Figure 5.13: CIRS Model 043A Doppler String Phantom with (left) motor controller and (right) tank and string loop

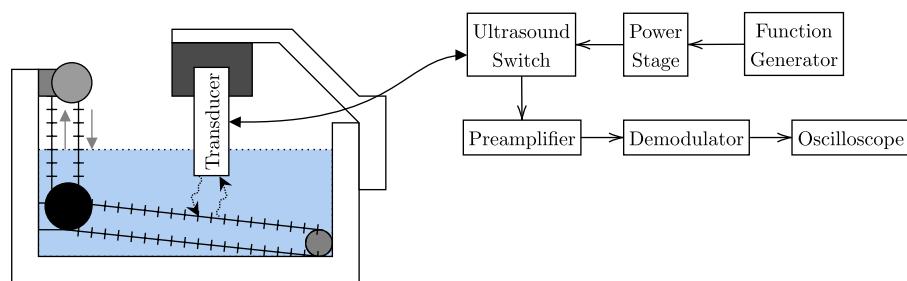


Figure 5.14: Doppler String Phantom experiment diagram (not to scale)

6 Conclusion

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

6.1 Future Work

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Bibliography

- [1] S. Satomura, T. Yoshida, M. Mori, Y. Nimura, G.-i. Hikita, S. Takagishi, and K. Nakanishi, “Analysis of heart motion with ultrasonic Doppler method and its clinical application”, *American Heart Journal*, vol. 61, pp. 61–75, 1 January 1961, ISSN: 00028703. DOI: 10.1016/0002-8703(61)90517-8. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/0002870361905178>.
- [2] D. Baker, “Pulsed Ultrasonic Doppler Blood-Flow Sensing”, *IEEE Transactions on Sonics and Ultrasonics*, vol. 17, pp. 170–184, 3 Jul. 1970, ISSN: 0018-9537. DOI: 10.1109/t-su.1970.29558. [Online]. Available: <http://ieeexplore.ieee.org/document/1538563/>.
- [3] K. K. Shung, R. A. Sigelmann, and J. M. Reid, “Scattering of Ultrasound by Blood”, *IEEE Transactions on Biomedical Engineering*, vol. Bme-23, pp. 460–467, 6 November 1976, ISSN: 0018-9294. DOI: 10.1109/tbme.1976.324604. [Online]. Available: <http://ieeexplore.ieee.org/document/4121084/>.
- [4] F. S. Schlindwein, M. J. Smith, and D. H. Evans, “Spectral analysis of Doppler signals and computation of the normalised first moment in real time using a digital signal processor”, *Medical & Biological Engineering & Computing*, vol. 26, pp. 228–232, 2 March 1988, ISSN: 0140-0118. DOI: 10.1007/bf02442271. [Online]. Available: <http://link.springer.com/10.1007/BF02442271>.
- [5] K. K. Shung and G. A. Thieme, *Ultrasonic scattering in biological tissues*. CRC Press, 1993, p. 499, ISBN: 9780849365683. [Online]. Available: <https://www.routledge.com/Ultrasonic-Scattering-in-Biological-Tissues/Shung-Thieme/p/book/9780849365683>.
- [6] T. Hall, “An improved wall filter for flow imaging of low velocity flow”, vol. 3, IEEE, October 1994, 1701–1704 vol.3, ISBN: 0-7803-2012-3. DOI: 10.1109/ultsym.1994.401918. [Online]. Available: <http://ieeexplore.ieee.org/document/401918/>.
- [7] L. Bessi, F. Guidi, F. Gucci, C. Atzeni, and P. Tortoli, “Single-channel digital demodulation of ultrasound bandpass signals for flow-imaging applications”, in Springer, Boston, MA, 1995, pp. 331–335. DOI: 10.1007/978-1-4615-1943-0_34. [Online]. Available: http://link.springer.com/10.1007/978-1-4615-1943-0_34.
- [8] J. A. Jensen, “An Analysis of Pulsed Wave Ultrasound Systems for Blood Velocity Estimation”, *Acoustical Imaging*, L. T. Piero and Masotti, Eds., pp. 377–384, 1996. DOI: 10.1007/978-1-4419-8772-3\61. [Online]. Available: <http://link.springer.com/10.1007/978-1-4419-8772-3%5C%5F61>.
- [9] I. Güler and Y. Sava, “Design parameters of pulsed wave ultrasonic Doppler blood flowmeter”, *Journal of Medical Systems*, vol. 22, pp. 273–278, 4 1998, ISSN: 01485598. DOI: 10.1023/a:1022665802010.
- [10] P. Wells, “Doppler studies of the vascular system”, *European Journal of Ultrasound*, vol. 7, pp. 3–8, 1 February 1998, ISSN: 09298266. DOI: 10.1016/s0929-8266(98)00006-8. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0929826698000068>.
- [11] P. J. Fish, “Ultrasonic investigation of blood flow”, *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 213, pp. 169–180, 3 March 1999, Pmid: 10420772. DOI: 10.1243/0954411991534898. [Online]. Available: <https://doi.org/10.1243/0954411991534898>.

- [12] P. R. Hoskins, “A review of the measurement of blood velocity and related quantities using Doppler ultrasound”, *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 213, pp. 391–400, 5 1999, Pmid: 10581966. DOI: 10.1243/0954411991535004. [Online]. Available: <https://doi.org/10.1243/0954411991535004>.
- [13] T. Jansson, H. W. Peresson, and K. Lindström, “Estimation of blood perfusion using ultrasound”, *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 213, pp. 193–193, 3 March 1999, ISSN: 0954-4119. DOI: 10.1243/0954411991534915. [Online]. Available: <http://journals.sagepub.com/doi/10.1243/0954411991534915>.
- [14] J. A. Jensen, “Algorithms for estimating blood velocities using ultrasound”, *Ultrasonics*, vol. 38, pp. 358–362, 1-8 March 2000, ISSN: 0041624x. DOI: 10.1016/s0041-624x(99)00127-4. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0041624X99001274>.
- [15] P. Munk, “Estimation of blood velocity vectors using ultrasound”, Technical University of Denmark, 2000. [Online]. Available: <https://findit.dtu.dk/en/catalog/537f0d537401dbcc1200be04>.
- [16] E. Picano, “Sustainability of medical imaging”, *BMJ*, vol. 328, pp. 578–580, 7439 March 2004, ISSN: 0959-8138. DOI: 10.1136/bmj.328.7439.578. [Online]. Available: <https://www.bmjjournals.org/lookup/doi/10.1136/bmj.328.7439.578>.
- [17] A. Erguri, Y. Huang, X. Zhuang, O. Oralkan, G. Yarahoglu, and B. Khuri-Yakub, “Capacitive micromachined ultrasonic transducers: Fabrication technology”, *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 52, no. 12, pp. 2242–2258, 2005. DOI: 10.1109/TUFFC.2005.1563267.
- [18] *AD8021 Low Noise, High Speed Amplifier for 16-Bit Systems*, Analog Devices, Inc., May 2006. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD8021.pdf> (visited on April 10, 2023).
- [19] E. Chérin, R. Williams, A. Needles, G. Liu, C. White, A. S. Brown, Y.-Q. Zhou, and F. S. Foster, “Ultrahigh frame rate retrospective ultrasound microimaging and blood flow visualization in mice *in vivo*”, *Ultrasound in Medicine & Biology*, vol. 32, pp. 683–691, 5 May 2006, ISSN: 03015629. DOI: 10.1016/j.ultrasmedbio.2005.12.015. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0301562906000172>.
- [20] *EL7104 High Speed, Single Channel, Power MOSFET Driver*, Renesas Electronics, Jun. 2006. [Online]. Available: <https://www.renesas.com/us/en/document/dst/el7104-datasheet>.
- [21] I. K. H. Tsang, B. Y. S. Yiu, D. K. H. Cheung, H. C. T. Chiu, C. C. P. Cheung, and A. C. H. Yu, “Design of a multi-channel pre-beamform data acquisition system for an ultrasound research scanner”, IEEE, Sep. 2009, pp. 1840–1843, ISBN: 978-1-4244-4389-5. DOI: 10.1109/ultsym.2009.5441869. [Online]. Available: <http://ieeexplore.ieee.org/document/5441869/>.
- [22] E. A. Aydin and . Güler, “Design Of Pic-controlled Pulsed Ultrasonic Transmitter For Measuring Gingiva Thickness”, *Instrumentation Science & Technology*, vol. 38, pp. 411–420, 6 October 2010, ISSN: 1073-9149. DOI: 10.1080/10739149.2010.509149. [Online]. Available: <http://www.tandfonline.com/doi/abs/10.1080/10739149.2010.509149>.
- [23] P. R. Hoskins, “Haemodynamics and blood flow measured using ultrasound imaging”, *Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine*, vol. 224, pp. 255–271, 2 February 2010, Pmid: 20349818, ISSN: 0954-4119.

- DOI: 10.1243/09544119jeim572. [Online]. Available: <http://journals.sagepub.com/doi/10.1243/09544119JEIM572>.
- [24] N. Matsuoka, D.-G. Paeng, R. Chen, H. Ameri, W. Abdallah, Q. Zhou, A. Fawzi, K. K. Shung, and M. Humayun, “Ultrasonic Doppler measurements of blood flow velocity of rabbit retinal vessels using a 45-MHz needle transducer”, *Graefe's Archive for Clinical and Experimental Ophthalmology*, vol. 248, pp. 675–680, 5 2010, ISSN: 1435-702x. DOI: 10.1007/s00417-009-1298-9. [Online]. Available: <https://doi.org/10.1007/s00417-009-1298-9>.
- [25] *TX810 8-Channel, Programmable T/R Switch for Ultrasound*, Texas Instruments, April 2010. [Online]. Available: <https://www.ti.com/lit/gpn/tx810>.
- [26] K. L. Hansen, M. B. Nielsen, and J. A. Jensen, “In-vivo studies of new vector velocity and adaptive spectral estimators in medical ultrasound”, 2011. [Online]. Available: <https://orbit.dtu.dk/en/publications/in-vivo-studies-of-new-vector-velocity-and-adaptive-spectral-estimator>.
- [27] J. A. Jensen, S. I. Nikolov, J. Udesen, et al., “Recent advances in blood flow vector velocity imaging”, IEEE, October 2011, pp. 262–271, ISBN: 978-1-4577-1252-4. DOI: 10.1109/ultsym.2011.0064. [Online]. Available: <http://ieeexplore.ieee.org/document/6293382/>.
- [28] A. Sarvazyan, T. J. Hall, M. W. Urban, M. Fatemi, S. R. Aglyamov, and B. S. Garra, “An Overview of Elastography-An Emerging Branch of Medical Imaging”, *Current Medical Imaging Reviews*, vol. 7, pp. 255–282, 4 November 2011, ISSN: 15734056. DOI: 10.2174/157340511798038684. [Online]. Available: <http://www.eurekaselect.com/openurl/content.php?genre=article%5C&issn=1573-4056%5C&volume=7%5C&issue=4%5C&spage=255>.
- [29] T. A. Bigelow, A. Kellar, A. Tapia, F. Ferrer, J. Batcheler, J. Driggs, and R. Page, “Design Document for a Pulse Echo Ultrasound System”, Iowa State University, December 2012.
- [30] C.-C. Huang, P.-Y. Lee, P.-Y. Chen, and T.-Y. Liu, “Design and implementation of a smartphone-based portable ultrasound pulsed-wave doppler device for blood flow measurement”, *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 59, pp. 182–188, 1 January 2012, ISSN: 0885-3010. DOI: 10.1109/tuffc.2012.2171. [Online]. Available: <http://ieeexplore.ieee.org/document/6138742/>.
- [31] C. A. Winckler, P. R. Smith, D. M. J. Cowell, O. Olagunju, and S. Freear, “The design of a high speed receiver system for an ultrasound array research platform”, Ieee, October 2012, pp. 1481–1484, ISBN: 978-1-4673-4562-0. DOI: 10.1109/ultsym.2012.0370. [Online]. Available: <http://ieeexplore.ieee.org/document/6562380/>.
- [32] J. A. Jensen, *Estimation of Blood Velocities Using Ultrasound: A Signal Processing Approach*, Third Edition. Department of Electrical Engineering, Technical University of Denmark, August 2013, ISBN: 9780521464840.
- [33] E. Pirnia, “Blood velocities estimation using ultrasound”, Lund University, Sep. 2013.
- [34] *AD783 Complete Very High Speed Sample-and-Hold Amplifier*, Analog Devices, October 2014. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD783.pdf>.
- [35] J. E. Browne, “A review of doppler ultrasound quality assurance protocols and test devices”, *Physica Medica*, vol. 30, no. 7, pp. 742–751, 2014, ISSN: 1120-1797. DOI: <https://doi.org/10.1016/j.ejmp.2014.08.003>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1120179714005171>.

- [36] *MD1213DB1 High Speed ±100V 2A Pulser*, Supertex, Inc., March 2014. [Online]. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/md1213db1.pdf>.
- [37] R. K. Sagdiev, E. S. Denisov, Y. K. Evdokimov, M. G. Fazlyyyakhmatov, and N. F. Kashapov, “Phased array based ultrasound scanning system development”, *IOP Conference Series: Materials Science and Engineering*, vol. 69, p. 012012, 1 December 2014, ISSN: 1757-899x. DOI: 10.1088/1757-899x/69/1/012012. [Online]. Available: <https://iopscience.iop.org/article/10.1088/1757-899x/69/1/012012>.
- [38] T. L. Szabo, *Diagnostic Ultrasound Imaging: Inside Out*, Second Edition. Elsevier, 2014, ISBN: 9780123964878. DOI: 10.1016/c2011-0-07261-7. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/C20110072617>.
- [39] K. K. Shung, *Diagnostic Ultrasound: Imaging and Blood Flow Measurements*, Second Edition. CRC Press, December 2015, ISBN: 978-1-4665-8264-4.
- [40] L. Svilainis, A. Chazachmetovas, and V. Dumbrava, “Half bridge topology 500 V pulser for ultrasonic transducer excitation”, *Ultrasonics*, vol. 59, pp. 79–85, May 2015, ISSN: 0041624x. DOI: 10.1016/j.ultras.2015.01.014. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0041624X15000165>.
- [41] *AD8332 Ultralow Noise VGAs with Preamplifier and Programmable R_{IN}*, Analog Devices, May 2016. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/AD8331_8332_8334.pdf.
- [42] *AD8333 DC to 50 MHz, Dual I/Q Demodulator and Phase Shifter*, Analog Devices, May 2016. [Online]. Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD8333.pdf>.
- [43] P. Govindan, B. Wang, P. Ravi, and J. Saniie, “Hardware and software architectures for computationally efficient three-dimensional ultrasonic data compression”, *IET Circuits, Devices & Systems*, vol. 10, pp. 54–61, 1 January 2016, ISSN: 1751-858x. DOI: 10.1049/iet-cds.2015.0083. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1049/iet-cds.2015.0083>.
- [44] B. Wang and J. Saniie, “Ultrasonic Signal Acquisition and Processing platform based on Zynq SoC”, vol. 2016-August, IEEE, May 2016, pp. 0448–0451, ISBN: 978-1-4673-9985-2. DOI: 10.1109/eit.2016.7535282. [Online]. Available: <http://ieeexplore.ieee.org/document/7535282/>.
- [45] *MD1213 High-Speed Dual-MOSFET Driver*, Microchip Technology, Inc., Jun. 2017. [Online]. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/20005713B.pdf>.
- [46] *PYNQ-Z1 Board Reference Manual*, Digilent, Inc., April 2017. [Online]. Available: https://digilent.com/reference/_media/reference/programmable-logic/pynq-z1/pynq-rm.pdf (visited on April 10, 2023).
- [47] S. Ricci and V. Meacci, “Data-adaptive coherent demodulator for high dynamics pulse-wave ultrasound applications”, *Electronics 2018, Vol. 7, Page 434*, vol. 7, p. 434, 12 December 2018, ISSN: 2079-9292. DOI: 10.3390/ELECTRONICS7120434. [Online]. Available: <https://www.mdpi.com/2079-9292/7/12/434/htm%20https://www.mdpi.com/2079-9292/7/12/434>.
- [48] *BPF-C4R5+ Bandpass Filter 50Ω 2 to 7 MHz*, Mini-Circuits, August 2019. [Online]. Available: <https://www.minicircuits.com/pdfs/BPF-C4R5+.pdf> (visited on April 10, 2023).
- [49] B. Wang and J. Saniie, “A High Performance Ultrasonic System for Flaw Detection”, IEEE, October 2019, pp. 840–843, ISBN: 978-1-7281-4596-9. DOI: 10.1109/ultsym.2019.8926280. [Online]. Available: <https://ieeexplore.ieee.org/document/8926280/>.

- [50] H. Ding, D. Yang, J. Xu, X. Chen, X. Le, Y. Wang, L. Lin, and J. Xie, “Pulsed Wave Doppler Ultrasound Using 3.7 MHz Pmuts Toward Wearable Blood Flow Measurements”, IEEE, January 2020, pp. 400–403, ISBN: 978-1-7281-3581-6. DOI: 10.1109/mems46641.2020.9056430. [Online]. Available: <https://ieeexplore.ieee.org/document/9056430/>.
- [51] B. Jana, R. Biswas, P. K. Nath, G. Saha, and S. Banerjee, “Smartphone-Based Point-of-Care System Using Continuous-Wave Portable Doppler”, *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 8352–8361, 10 October 2020, ISSN: 0018-9456. DOI: 10.1109/tim.2020.2987654. [Online]. Available: <https://ieeexplore.ieee.org/document/9064842/>.
- [52] *MD0101 High Voltage Protection T/R Switch with Clamp Diodes*, Microchip Technology, Inc., May 2020. [Online]. Available: <https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MD0101-High-Voltage-Protection-TR-Switch-with-Clamp-Diodes-Data-Sheet-20005916A.pdf>.
- [53] J. Cumps. (Jun. 13, 2021), [Online]. Available: <https://community.element14.com/technologies/fpga-group/b/blog/posts/vhdl-pwm-generator-with-dead-time-the-design> (visited on May 9, 2023).
- [54] H. Ding, D. Yang, M. Qu, et al., “A Pulsed Wave Doppler Ultrasound Blood Flowmeter by PMUTs”, *Journal of Microelectromechanical Systems*, vol. 30, pp. 680–682, 5 October 2021, ISSN: 1941-0158. DOI: 10.1109/jmems.2021.3103756. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9515180>.
- [55] *ISL55111 Dual, High Speed MOSFET Driver*, Renesas Electronics, April 2021. [Online]. Available: <https://www.renesas.com/us/en/document/dst/isl55110-isl55111-datasheet>.
- [56] M. Winder, A. J. Owczarek, J. Chudek, J. Pilch-Kowalczyk, and J. Baron, “Are We Overdoing It? Changes in Diagnostic Imaging Workload during the Years 2010–2020 including the Impact of the SARS-CoV-2 Pandemic”, *Healthcare*, vol. 9, p. 1557, 11 November 2021, ISSN: 2227-9032. DOI: 10.3390/healthcare9111557. [Online]. Available: <https://www.mdpi.com/2227-9032/9/11/1557>.
- [57] “Heart disease facts”. (October 2022), [Online]. Available: <https://www.cdc.gov/heartdisease/facts.htm> (visited on February 23, 2023).
- [58] M. Omura, R. Nagaoka, K. Yagi, K. Yoshida, T. Yamaguchi, and H. Hasegawa, “Characterization of blood mimicking fluid with ultrafast ultrasonic and optical image velocimeters”, *Japanese Journal of Applied Physics*, vol. 61, Sg1067, Sg Jul. 2022, Publisher Copyright: © 2022 The Japan Society of Applied Physics., ISSN: 0021-4922. DOI: 10.35848/1347-4065/ac4ea9. [Online]. Available: <https://iopscience.iop.org/article/10.35848/1347-4065/ac4ea9>.
- [59] “Altium Designer - PCB design software”. (February 2023), [Online]. Available: <https://www.altium.com/altium-designer>.
- [60] J. Cumps, Private communication, Talked about implementation of PWM generators on PYNQ hardware platforms, 2023.
- [61] L. Li, L. Zhao, R. Hassan, and H. Ren, “Review on Wearable System for Positioning Ultrasound Scanner”, *Machines*, vol. 11, p. 325, 3 February 2023, ISSN: 2075-1702. DOI: 10.3390/machines11030325. [Online]. Available: <https://www.mdpi.com/2075-1702/11/3/325>.
- [62] *Sample and hold Sampling Zero-order hold First-order hold Digital-to-analog converter*. [Online]. Available: <https://www.pngwing.com/en/free-png-zueso> (visited on April 10, 2023).

A Source Code

A.1 Microcontroller Code

Code A.1: main.c

```
1  /*
2   * Copyright (c) 2023 JH
3   *
4   * SPDX-License-Identifier: Apache-2.0
5   */
6
7 /**
8  * @file Doppler PWM app
9 */
10
11 #include <zephyr/kernel.h>
12 #include <zephyr/sys/printk.h>
13 #include <zephyr/device.h>
14 #include <zephyr/drivers/pwm.h>
15
16 #include <spinner/drivers/svpwm.h>
17
18 static const struct pwm_dt_spec pwm_t1 = PWM_DT_SPEC_GET(DT_ALIAS(pwmch1));
19 //static const struct pwm_dt_spec pwm_ch1n = PWM_DT_SPEC_GET(DT_ALIAS(pwmch1n));
20 static const struct pwm_dt_spec pwm_t2 = PWM_DT_SPEC_GET(DT_ALIAS(pwmch2));
21 static const struct pwm_dt_spec pwm_t4 = PWM_DT_SPEC_GET(DT_ALIAS(pwmch4));
22 #define SLEEP_MSEC      100U
23
24 struct pwmcontrol {
25     const struct device *svpwm;
26 }
27
28 void main(void)
29 {
30     printk("Test af main - %s\r\n", CONFIG_BOARD);
31     uint32_t pulse_width_t1 = 152U;
32     uint32_t pulse_width_t2 = 1515U;
33     uint32_t pulse_width_t4 = 50000U;
34     int error = 0;
35
36     printk("PWM-firmware API\n");
37     if (!device_is_ready(pwm_t1.dev)) {
38         printk("Error: PWM device %s is not ready\n",
39               pwm_t1.dev->name);
40         return;
41     }
42
43     if (!device_is_ready(pwm_t2.dev)) {
44         printk("Error: PWM device %s is not ready\n",
45               pwm_t2.dev->name);
46         return;
47     }
48     if (!device_is_ready(pwm_t4.dev)) {
49         printk("Error: PWM device %s is not ready\n",
50               pwm_t4.dev->name);
51         return;
```

```

52     }*/
53     printk("Entering while loop\r\n");
54     while (1) {
55
56         error = pwm_set_pulse_dt(&pwm_t1, pulse_width_t1);
57         if (error) {
58             printk("Error ch1 %d: failed to set pulse width\n", error);
59             return;
60         }
61         error = pwm_set_pulse_dt(&pwm_t2, pulse_width_t2);
62         if (error) {
63             printk("Error ch1n %d: failed to set pulse width\n", error);
64             return;
65         }
66         error = pwm_set_pulse_dt(&pwm_t4, pulse_width_t4);
67         if (error) {
68             printk("Error ch1n %d: failed to set pulse width\n", error);
69             return;
70         }
71         for(int i=0;i<1000;i++) {
72             k_sleep(K_MSEC(SLEEP_MSEC));
73         }
74         printk("Loop iteration\r\n");
75     }
76 }
```

A.2 Field Programmable Gate Array Code

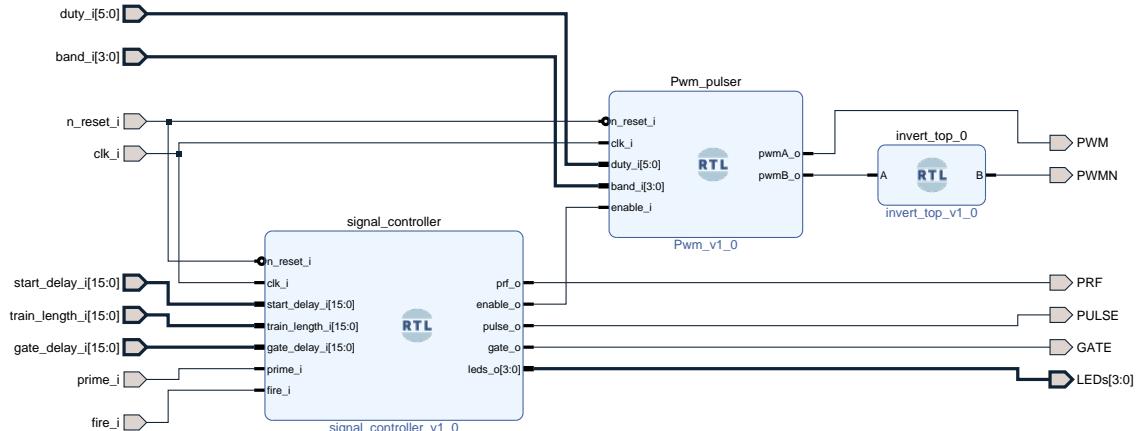


Figure A.1: Jupyter Notebook running on PYNQ Z1

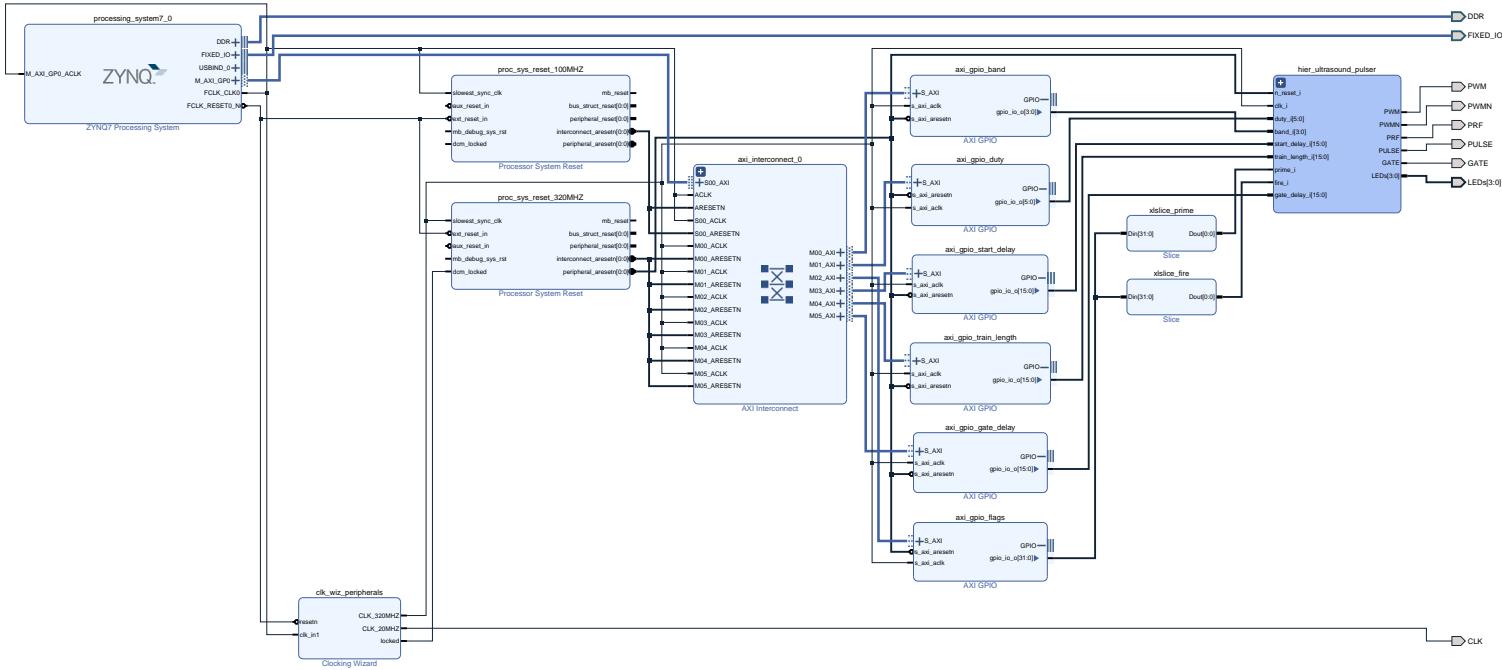


Figure A.2: ZYNQ Top Level block diagram

Code A.2: Pwm.vhdl

```

1  -----
2  -- Copyright (c) 2011-2014 by XESS Corp <http://www.xess.com>.
3  -- All rights reserved.
4  --
5  -- This library is free software; you can redistribute it and/or
6  -- modify it under the terms of the GNU Lesser General Public
7  -- License as published by the Free Software Foundation; either
8  -- version 3.0 of the License, or (at your option) any later version.
9  --
10 -- This library is distributed in the hope that it will be useful,
11 -- but WITHOUT ANY WARRANTY; without even the implied warranty of
12 -- MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
13 -- Lesser General Public License for more details.
14 --
15 -- You should have received a copy of the GNU Lesser General Public
16 -- License along with this library. If not, see
17 -- <http://www.gnu.org/licenses/>.
18 -----
19 --
20 -----
21 -- Module for generating repetitive pulses.
22 -----
23 
24 library IEEE;
25 use IEEE.std_logic_1164.all;
26 
27 package PulsePckg is
28 
29     constant HI    : std_logic := '1';
30     constant LO    : std_logic := '0';
31     constant ONE   : std_logic := '1';
32 
33     component Pwm is
34         generic (
35             resolution : integer := 6; -- bit width of the counter used for duty cycle
36             band_bits : integer := 4   -- bit width of the variable that holds the
37             -- desired deadband
38         );
39         port (
40             n_reset_i : in std_logic;           -- async reset
41             enable_i : in std_logic;          -- enable the outputs
42             clk_i : in std_logic;            -- Input clock.
43             duty_i : in std_logic_vector (resolution - 1 downto 0);      -- Duty-cycle
44             -- input.
45             band_i : in std_logic_vector (band_bits - 1 downto 0);        -- number of
46             -- clock-ticks to keep both signals low before rising edge
47             pwmA_o : out std_logic;          -- PWM output.
48             pwmB_o : out std_logic          -- PWM output inverse.
49         );
50     end component;
51 
52 end package;
53 
54 -----
55 -- PWM module.
56 -----
57 
58 library IEEE;

```

```

56  use IEEE.MATH_REAL.all;
57  use IEEE.std_logic_1164.all;
58  use IEEE.numeric_std.all;
59  use WORK.PulsePckg.all;
60
61  entity Pwm is
62      generic (
63          resolution : integer := 6;
64          band_bits : integer := 4
65      );
66      port (
67          n_reset_i : in std_logic;           -- async reset
68          enable_i : in std_logic;           -- enable the outputs
69          clk_i : in std_logic;              -- Input clock.
70          duty_i : in std_logic_vector (resolution - 1 downto 0);    -- Duty-cycle
71          -- input.
72          band_i : in std_logic_vector (band_bits - 1 downto 0);      -- number of
73          -- clock-ticks to keep both signals low before rising edge
74          pwmA_o : out std_logic;           -- PWM output.
75          pwmB_o : out std_logic           -- PWM output inverse.
76      );
77  end entity;
78
79  architecture arch of Pwm is
80      signal timer_r      : natural range 0 to 2**duty_i'length-1;
81  begin
82
83      clocked: process(clk_i)
84      begin
85
86          if rising_edge(clk_i) then
87              -- sync reset
88              if n_reset_i = '0' then
89                  pwmA_o    <= LO;
90                  pwmB_o    <= LO;
91                  timer_r  <= 0;
92              else
93                  -- timer
94                  timer_r  <= timer_r + 1;
95                  pwmA_o    <= LO;
96                  pwmB_o    <= LO;
97                  if enable_i = '0' then
98                      pwmB_o    <= LO;
99                      timer_r  <= 0;
100                 else
101                     -- output a
102                     if timer_r <= unsigned(duty_i) and timer_r >= unsigned(band_i)  then
103                         pwmA_o  <= HI;
104                     end if;
105                     -- output b
106                     if timer_r > to_integer(unsigned(band_i)) +
107                         to_integer(unsigned(duty_i)) then
108                         pwmB_o  <= HI;
109                     end if;
110                     end if; -- enable
111
112                     end if; -- sync reset
113                 end if; -- rising_edge
114             end process clocked;

```

```

114
115 end architecture;
```

Code A.3: signalcontroller.vhdl

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 package PulsePckg is
5
6     component signal_controller is
7         generic (
8             start_delay_resolution : integer := 6; -- bit width of the counter used
9             -- delaying pulse train start
10            train_length : integer := 7; -- bit width of the PWM burst train
11            -- counter
12            gate_delay_resolution : integer := 6; -- bit width of the counter used
13            -- delaying the gate signal start
14            counter_resolution : integer := 7 -- big enough to hold the largest of
15            -- above
16        );
17        port (
18             n_reset_i : in std_logic;
19             -- async reset
20             clk_i : in std_logic;
21             -- Input clock.
22             start_delay_i : in std_logic_vector (start_delay_resolution - 1 downto 0);
23             -- delay before the pulse train.
24             train_length_i: in std_logic_vector (train_length - 1 downto 0);
25             -- how many PWMs in a burst train.
26             gate_delay_i : in std_logic_vector (gate_delay_resolution - 1 downto 0);
27             -- delay before gate signal.
28             prime_i : in std_logic;
29             -- allow cycle start
30             fire_i : in std_logic;
31             -- start a cycle
32             prf_o : out std_logic;
33             -- PRF
34             enable_o : out std_logic;
35             -- enable pwm
36             pulse_o : out std_logic;
37             -- PULSE
38             gate_o : out std_logic;
39             -- GATE
40             leds_o : out std_logic_vector (3 downto 0)
41         );
42     end component;
43
44 end package;
45 library IEEE;
46 use IEEE.STD_LOGIC_1164.ALL;
47 use ieee.numeric_std.all;
48
49
50 entity signal_controller is
51     generic (
52         train_length : integer := 7; -- bit width of the PWM burst train
53         -- counter
54         start_delay_resolution : integer := 6; -- bit width of the counter used delaying
55         -- pulse train start
```

```

39      gate_delay_resolution : integer := 6; -- bit width of the counter used delaying
40      -- the gate signal start
41      counter_resolution    : integer := 7 -- big enough to hold the largest of
42      -- above
43  );
44  port (
45      n_reset_i      : in  std_logic;           --
46      -- async reset
47      clk_i          : in  std_logic;           --
48      -- Input clock.
49      start_delay_i : in  std_logic_vector (start_delay_resolution - 1 downto 0); --
50      -- delay before the pulse train.
51      train_length_i: in  std_logic_vector (train_length - 1 downto 0);           --
52      -- how many PWMs in a burst train.
53      gate_delay_i  : in  std_logic_vector (gate_delay_resolution - 1 downto 0); --
54      -- delay before gate signal.
55      prime_i        : in  std_logic;           --
56      -- allow cycle start
57      fire_i         : in  std_logic;           --
58      -- start a cycle
59      prf_o          : out std_logic;           --
60      -- PRF
61      enable_o       : out std_logic;           --
62      -- enable pwm
63      pulse_o        : out std_logic;           --
64      -- PULSE
65      gate_o         : out std_logic;           --
66      -- GATE
67      leds_o         : out std_logic_vector (3 downto 0)
68  );
69 end signal_controller;
70
71 architecture arch of signal_controller is
72     type states is (ready, delay, pulse, gate_delay, gate, done );
73     signal counter : integer range 0 to 2**counter_resolution-1;
74     signal state   : states;
75
76 begin
77
78     clocked: process(clk_i)
79
80     begin
81
82         if rising_edge(clk_i) then
83             -- sync reset
84             if n_reset_i = '0' then
85                 counter <= 0;
86                 state <= ready;
87                 -- reset output states
88                 prf_o <= '1';
89                 enable_o <= '0';
90                 pulse_o <= '0';
91                 gate_o <= '0';
92                 leds_o <= (others => '0');
93             else
94                 counter <= counter + 1;
95                 -- default output states
96                 prf_o <= '1';
97                 enable_o <= '0';
98                 pulse_o <= '0';
99             end if;
100            report "clocked process completed" severity note;
101        end if;
102    end process;
103
104    state: process(state)
105
106        begin
107            case state is
108                when ready =>
109                    if prime_i = '1' then
110                        state <= delay;
111                    else
112                        state <= done;
113                    end if;
114                when delay =>
115                    if counter = gate_delay_i then
116                        state <= pulse;
117                    else
118                        state <= delay;
119                    end if;
120                when pulse =>
121                    if counter = train_length_i then
122                        state <= gate;
123                    else
124                        state <= pulse;
125                    end if;
126                when gate =>
127                    if counter = gate_delay_i then
128                        state <= done;
129                    else
130                        state <= gate;
131                    end if;
132                when others =>
133                    state <= ready;
134            end case;
135        end process;
136
137    end;

```

```

87         gate_o <= '0';
88         leds_o <= (others => '0');
89
90     case State is
91
92         when ready => -- start a pulse group as soon as the fire bit is set
93             leds_o <= (0 => '1', others => '0');
94             -- check if the caller has reset the prime bit
95             if prime_i = '0' then
96                 -- if yes, start if the fire bit is set.
97                 if fire_i = '1' then
98                     counter <= 0;
99                     state <= delay;
100                end if;
101            end if;
102
103         when delay =>
104             leds_o <= (0 => '0', 1 => '1', others => '0');
105             prf_o <= '0';
106             if counter >= to_integer(unsigned(start_delay_i)) - 1 then
107                 counter <= 0;
108                 enable_o <= '1';
109                 state <= pulse;
110             end if;
111
112         when pulse =>
113             leds_o <= (0 => '1', 1 => '1', others => '0');
114             enable_o <= '1';
115             prf_o <= '0';
116             if counter < to_integer(unsigned(train_length_i)) - 1 then -- 
117                 -- let the pulse change sync with last edge of PWM
118                 pulse_o <= '1';
119             elsif counter = to_integer(unsigned(train_length_i)) - 1 then
120                 pulse_o <= '1';
121                 counter <= 0;
122                 state <= gate_delay;
123             end if;
124
125         when gate_delay =>
126             leds_o <= (0 => '0', 1 => '0', 2 => '1', others => '0');
127             if counter >= to_integer(unsigned(gate_delay_i)) - 1 then
128                 counter <= 0;
129                 state <= gate;
130             end if;
131
132         when gate =>
133             leds_o <= (0 => '1', 1 => '0', 2 => '1', others => '0');
134             gate_o <= '1';
135             if counter = to_integer(unsigned(train_length_i)) - 1 then
136                 counter <= 0;
137                 state <= done; -- todo implement
138             end if;
139
140         when done =>
141             leds_o <= (0 => '1', 1 => '1', 2 => '1', others => '0');
142             -- check if the caller has reset the fire bit
143             if fire_i = '0' then
144                 -- if yes, prime if the prime_bit is set.
145                 if prime_i = '1' then
146                     counter <= 0;
                     state <= ready;

```

```

147         end if;
148     end if;
149
150     end case;
151
152     end if; -- sync reset
153 end if; -- rising_edge
154
155 end process clocked;
156
157 end architecture;

```

Code A.4: inv_top.vhdl

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity invert_top is
5     Port ( A : in STD_LOGIC;
6             B : out STD_LOGIC);
7 end invert_top;
8
9 architecture Behavioral of invert_top is
10 begin
11     -- invert the signal from the push button switch and route it to the LED
12     B <= not A;
13 end Behavioral;

```

A.3 MATLAB Scripts

Code A.5: US Simulation Master Script.m

```

1 %% CIRCUIT: PA
2 clear; close all; clc;
3 path_LTSpice = '"C:/Program Files/LTC/LTspiceXVII/XVIIx64.exe"';
4 circuit_path = 'C:/Users/Jeppe/OneDrive - Danmarks Tekniske Universitet/10.
→ semester/MSc_Thesis_Teams/Simulation/';
5 circuit_name = 'Tx_PA';
6 command = ' -Run -ascii -b ';
7 %% Simulate and import data
8 % Run LTSpice simulation
9 command = [path_LTSpice command "" circuit_path circuit_name '.asc'];
10 dos(command);
11
12 % Import data
13 raw_data = LTspice2Matlab([circuit_path circuit_name '.raw']);
14
15 % Display variables
16 fprintf('var \t name\n')
17 for i=1:raw_data.num_variables
18     fprintf('%d \t\t %s \n', raw_data.selected_vars(i), raw_data.variable_name_list{i})
19 end
20 %% Plot data
21 % Trace numbers
22 var_va = 6;

```

```

23 var_vb = 1;
24 var_vo = 9;
25 var_irout = 26;
26
27 Open_TimeVect = raw_data.time_vect;
28 Open_va = raw_data.variable_mat(var_va,:);
29 Open_vb = raw_data.variable_mat(var_vb,:);
30 Open_var_vo = raw_data.variable_mat(var_vo,:);
31 Open_var_irout = raw_data.variable_mat(var_irout,:);

32
33 figure(1)
34 subplot(3,1,1)
35 hold on
36 plot(raw_data.time_vect*1e9, raw_data.variable_mat(var_va,:), 'k');
37 plot(raw_data.time_vect*1e9, raw_data.variable_mat(var_vb,:), '--k');
38 title(sprintf('Waveform %s and %s',
39         ↳ raw_data.variable_name_list{var_va},raw_data.variable_name_list{var_vb}));
39 grid minor
40 ylabel(raw_data.variable_type_list{var_va});
41 xlabel('Time [ns]');
42 xlim([min(raw_data.time_vect*1e9) max(raw_data.time_vect*1e9)])
43 legend(raw_data.variable_name_list{var_va},
44         ↳ raw_data.variable_name_list{var_vb}, 'location', 'best');
44 hold off
45 subplot(3,1,2)
46 hold on
47 plot(raw_data.time_vect*1e9, raw_data.variable_mat(var_vo,:), 'k');
48 title(sprintf('Waveform %s', raw_data.variable_name_list{var_vo}));
49 grid minor
50 ylabel(sprintf('%s [V]', raw_data.variable_type_list{var_vo}));
51 xlabel('Time [ns]');
52 xlim([min(raw_data.time_vect*1e9) max(raw_data.time_vect)*1e9])
53 %xlim([SimDelay,SimStop]);
54 %ylim([min(raw_data.variable_mat(var_opa_out,:)-raw_data.variable_mat(var_spk_m,:)),max
55 %       ↳ (raw_data.variable_mat(var_opa_out,:)-raw_data.variable_mat(var_spk_m,:))])
56 %legend(raw_data.variable_name_list{var_vo}, 'location', 'best');
56 hold off
57 %sgt = sgtitle('Open loop', 'Color', 'black');
58 %sgt.FontSize = 20;
59 subplot(3,1,3)
60 hold on
61 plot(raw_data.time_vect*1e9,
62         ↳ -raw_data.variable_mat(var_vo,:).*raw_data.variable_mat(var_irout,:), 'k');
63 %title(sprintf('Waveform -%s \%|\cdot% s', raw_data.variable_name_list
64 %         ↳ {var_vo},raw_data.variable_name_list{var_irout}), 'Interpreter', 'latex');
64 title(sprintf('Waveform -%s*%s',
65         ↳ raw_data.variable_name_list{var_vo},raw_data.variable_name_list{var_irout}));
66 grid minor
67 ylabel('Power [W]');
68 xlabel('Time [ns]');
69 xlim([min(raw_data.time_vect*1e9) max(raw_data.time_vect*1e9)])
70 %legend(raw_data.variable_name_list{var_vo}, 'location', 'best');
70 hold off
71 %% CIRCUIT: Preamplifier
72 clear; close all; clc;
73 path_LTSpice = '"C:/Program Files/LTC/LTspiceXVII/XVIIx64.exe"';
74 circuit_path = 'C:/Users/Jeppe/OneDrive - Danmarks Tekniske Universitet/10.
75         ↳ Semester/MSc_Thesis_Teams/Simulation/';
75 circuit_name = 'Rx_preamp';
76 command = ' -Run -ascii -b ';
76 %% Simulate and import data

```

```

77 % Run LTSpice simulation
78 command = [path_LTSpice command """ circuit_path circuit_name '.asc"'];
79 dos(command);
80
81 % Import data
82 raw_data = LTspice2Matlab([circuit_path circuit_name '.raw']);
83
84 % Display variables
85 fprintf('var \t name\n')
86 for i=1:raw_data.num_variables
87     fprintf('%d \t %s \n', raw_data.selected_vars(i), raw_data.variable_name_list{i})
88 end
89 %% Plot data
90 % Trace numbers
91 var_input = 18;
92 var_opa_out = 14;
93 var_vga_vip = 7;
94 var_vga_vin = 6;
95 var_vga_oh = 15;
96 var_vga.ol = 16;
97
98 Open_TimeVect = raw_data.time_vect;
99 Open_opa_out = raw_data.variable_mat(var_opa_out,:);
100 Open_input = raw_data.variable_mat(var_input,:);
101 Open_vga_vip = raw_data.variable_mat(var_vga_vip,:);
102 Open_vga_vin = raw_data.variable_mat(var_vga_vin,:);
103 Open_vga_oh = raw_data.variable_mat(var_vga_oh,:);
104 Open_vga.ol = raw_data.variable_mat(var_vga.ol,:);
105
106 figure(1)
107 subplot(2,1,1)
108 hold on
109 plot(raw_data.time_vect, raw_data.variable_mat(var_input,:), 'k');
110 plot(raw_data.time_vect, raw_data.variable_mat(var_opa_out,:), '--k');
111 title(sprintf('Waveform %s and %s',
112     raw_data.variable_name_list{var_input}, raw_data.variable_name_list{var_opa_out}));
113 grid minor
114 ylabel(raw_data.variable_type_list{var_input});
115 xlabel('Time [sec]');
116 xlim([0 3e-7])
117 legend(raw_data.variable_name_list{var_input},
118     raw_data.variable_name_list{var_opa_out}, 'location', 'best');
119 hold off
120 %subplot(3,1,2)
121 %hold on
122 %plot(raw_data.time_vect, raw_data.variable_mat(var_vga_vip,:), 'k');
123 %plot(raw_data.time_vect, raw_data.variable_mat(var_vga_vin,:), '--k');
124 %title(sprintf('Waveform %s and %s',
125     raw_data.variable_name_list{var_vga_vip}, raw_data.variable_name_list{var_vga_vin}));
126 %grid on
127 %ylabel(raw_data.variable_type_list{var_opa_out});
128 %xlabel('Time [sec]');
129 %xlim([SimDelay,SimStop]);
130 %ylim([min(raw_data.variable_mat(var_opa_out,:)-raw_data.variable_mat(var_spk_m,:)), max
131     (raw_data.variable_mat(var_opa_out,:)-raw_data.variable_mat(var_spk_m,:))])
132 %legend(raw_data.variable_name_list{var_vga_vip},
133     raw_data.variable_name_list{var_vga_vin}, 'location', 'best');
134 %hold off
135 %sgt = sgtitle('Open loop', 'Color', 'black');
136 %sgt.FontSize = 20;
137 subplot(2,1,2)

```

```

133 hold on
134 plot(raw_data.time_vect, raw_data.variable_mat(var_vga_oh,:), 'k');
135 plot(raw_data.time_vect, raw_data.variable_mat(var_vga_ol,:), '--k');
136 title(sprintf('Waveform %s and %s',
137    ↪ raw_data.variable_name_list{var_vga_oh},raw_data.variable_name_list{var_vga_ol}));
138 grid minor
139 ylabel(raw_data.variable_type_list{var_opa_out});
140 xlabel('Time [sec]');
141 xlim([0 3e-7])
142 %xlim([SimDelay,SimStop]);
143 ylim([min(raw_data.variable_mat(var_opa_out,:)-raw_data.variable_mat(var_spk_m,:)),max
144    ↪ (raw_data.variable_mat(var_opa_out,:)-raw_data.variable_mat(var_spk_m,:))])
145 legend(raw_data.variable_name_list{var_vga_oh},
146    ↪ raw_data.variable_name_list{var_vga_ol}, 'location', 'best');
147 hold off
148 %% CIRCUIT: DEMODULATOR
149 clear; close all; clc;
150 path_LTSpice = 'C:/Program Files/LTC/LTspiceXVII/XVIIx64.exe';
151 circuit_path = 'C:/Users/Jeppe/OneDrive - Danmarks Tekniske Universitet/10.
152    ↪ semester/MSc_Thesis_Teams/Simulation/';
153 circuit_name = 'Rx_demod';
154 command = ' -Run -ascii -b ';
155 %% Simulate and import data
156 % Run LTSpice simulation
157 command = [path_LTSpice command " " circuit_path circuit_name '.asc'];
158 dos(command);
159
160 % Import data
161 raw_data = LTspice2Matlab([circuit_path circuit_name '.raw']);
162
163 % Display variables
164 fprintf('var \t name\n')
165 for i=1:raw_data.num_variables
166     fprintf('%d \t %s \n', raw_data.selected_vars(i), raw_data.variable_name_list{i})
167 end
168 %% Plot data
169 %% Trace numbers
170 var_lop = 9;
171 var_lon = 10;
172 var_vga_ol = 7;
173 var_vga_oh = 6;
174 var_out_i = 17;
175 var_out_q = 19;
176
177 figure(1)
178 %subplot(2,1,1)
179 hold on
180 plot(raw_data.time_vect, raw_data.variable_mat(var_vga_ol,:), 'k');
181 plot(raw_data.time_vect, raw_data.variable_mat(var_vga_oh,:), '--k');
182 title(sprintf('Waveform %s and %s',
183    ↪ raw_data.variable_name_list{var_vga_ol},raw_data.variable_name_list{var_vga_oh}));
184 grid minor
185 ylabel(raw_data.variable_type_list{var_vga_ol});
186 xlabel('Time [sec]');
187 xlim([1.5e-5 1.53e-5])
188 legend(raw_data.variable_name_list{var_vga_ol},
189    ↪ raw_data.variable_name_list{var_vga_oh}, 'location', 'best');
190 hold off
191 figure(2)
192 %subplot(2,1,2)
193 hold on

```

```

188 plot(raw_data.time_vect, raw_data.variable_mat(var_lop,:), 'k');
189 plot(raw_data.time_vect, raw_data.variable_mat(var_lon,:), ':k');
190 grid minor
191 legend(raw_data.variable_name_list{var_lop},
192     ↳ raw_data.variable_name_list{var_lon}, 'location', 'best');
193 ylabel(raw_data.variable_type_list{var_lop});
194 xlabel('Time [sec]');
195 xlim([1.5e-5 1.53e-5])
196 ylim([1.05 1.35])
197 hold off
198
199 figure(3)
200 hold on
201 %plot(raw_data.time_vect, movmean(raw_data.variable_mat(var_out_i,:),1000), 'k');
202 plot(raw_data.time_vect, raw_data.variable_mat(var_out_i,:), 'k');
203 %plot(raw_data.time_vect, movmean(raw_data.variable_mat(var_out_q,:),1000), '--k');
204 plot(raw_data.time_vect, raw_data.variable_mat(var_out_q,:), '--k');
205 title(sprintf('Waveform %s and %s',
206     ↳ raw_data.variable_name_list{var_out_i},raw_data.variable_name_list{var_out_q}));
207 grid minor
208 ylabel(raw_data.variable_type_list{var_out_i});
209 xlabel('Time [sec]');
210 %xlim([1.5e-5 1.5e-4])
211 legend(raw_data.variable_name_list{var_out_i},
212     ↳ raw_data.variable_name_list{var_out_q}, 'location', 'best');
213 hold off
214 %% CIRCUIT: dc-coupler TRANSIENT
215 clear; close all; clc;
216 path_LTSpice = '"C:/Program Files/LTC/LTspiceXVII/XVIIx64.exe"';
217 circuit_path = 'C:/Users/Jeppe\OneDrive - Danmarks Tekniske Universitet/10.
218     ↳ semester/MSc_Thesis_Teams/Simulation/';
219 circuit_name = 'dc-coupler';
220 command = ' -Run -ascii -b ';
221 % Simulate and import data
222 % Run LTSpice simulation
223 command = [path_LTSpice command '"' circuit_path circuit_name '.asc'"'];
224 dos(command);
225
226 % Import data
227 raw_data = LTspice2Matlab([circuit_path circuit_name '.raw']);
228
229 % Display variables
230 fprintf('var \t name\n')
231 for i=1:raw_data.num_variables
232     fprintf('%d \t\t %s \n', raw_data.selected_vars(i), raw_data.variable_name_list{i})
233 end
234 % Plot data
235 % Trace numbers
236 var_in = 4;
237 var_out = 1;
238
239 figure(1)
240 hold on
241 plot(raw_data.time_vect, raw_data.variable_mat(var_in,:), '--k');
242 plot(raw_data.time_vect, raw_data.variable_mat(var_out,:), 'k');
243 grid minor
244 legend(raw_data.variable_name_list{var_in},
245     ↳ raw_data.variable_name_list{var_out}, 'location', 'best');
246 ylabel(raw_data.variable_type_list{var_in});
247 xlabel('Time [sec]');
248 hold off

```

```

244 %% FREQ analysis
245 clear; close all; clc;
246 fidi = fopen('dc-coupler.txt');
247 Dc = textscan(fidi, '%f(%dB,%f)', 'CollectOutput',1);
248 D = cell2mat(Dc);
249 figure
250 sgttitle('\textbf{Bode Plot}', 'interpreter', 'latex');
251 subplot(2,1,1)
252 semilogx(D(:,1), D(:,2), 'k')
253 ylabel('Amplitude [dB]', 'Interpreter', 'latex')
254 grid
255 subplot(2,1,2)
256 semilogx(D(:,1), D(:,3), 'k')
257 ylabel('Phase [deg]', 'Interpreter', 'latex')
258 grid
259 xlabel('Frequency [Hz]', 'Interpreter', 'latex')

```

Code A.6: Analysis.m

```

1 close all; clc; clear
2 index=6;
3 %% Setting
4 param_title= {'fontsize',18, 'fontname','Arial'};
5 param_label={ 'fontsize',15,'fontname','Arial'};

6
7 save_fig=0;
8 F_prf=1/99.99e-6;
9 M=5;
10 f0=5e6;
11 fres=5;
12 dB_scale=0;
13 padding = 1;
14 sampled_interpolation = 0;
15 interpolation_factor=100;

16
17 %% Get data
18 file_name1=['scope_' num2str(index) '_2.csv'];
19 file_name2=['scope_' num2str(index) '_4.csv'];
20 data1 = xlsread(file_name1);
21 data2 = xlsread(file_name2);
22 time=data1(3:length(data1)-3,1);
23 v1=data1(3:length(data1)-3,2);
24 v2=data2(3:length(data2)-3,2);

25
26
27 %% chose signal
28 signal = v1;

29
30
31 %% Bandpass filter before sampling
32 dt=mean(diff(time));
33 fs = 1/dt;
34 fn=fs/2;
35 [c, d]=butter(2, [3e6 7e6]/fn, 'bandpass');
36 signal=filter(c,d,signal);

37
38 %% Power
39 power_signal = (rms(signal))^2;

```

```

41
42 %% t_start
43 idx_start=min(find(v2>max(v2)*0.7));
44 t_start = time(idx_start);
45
46 % idx_start = max(find(time<-411e-6));
47 % t_start = time(idx_start);
48 %% N count
49 count = t_start;
50 N=0;                                     %number of elements in sampled array
51 while count<time(end)
52     count = count + 1/F_prf;
53     N=N+1;
54 end
55
56 %% Cutting
57 idx_nxt=min(find(time>(t_start+ 1/F_prf)));
58 t_nxt = time(idx_nxt);
59 Is_prf=1/(t_nxt-t_start);
60 idx_length= idx_nxt - idx_start;
61 signal_array = zeros(N, idx_length);
62 idx_count = idx_start;
63 height_offset =max(signal)*2;
64 cut_time = (1:idx_length)*dt;
65 figure()
66 hold on
67 for i=1:N
68     if (idx_count+idx_length-1 < length(signal))
69         signal_array(i,:)= signal(idx_count:idx_count+idx_length-1);
70         signal_array(i,:) = signal_array(i,:) + height_offset*(i-1);
71         plot(cut_time*1e6, signal_array(i,:))
72         xlabel('Time(\mu s)',param_label{:})
73 % %         title('Cutted data',param_title{:})
74         xlim([0 cut_time(end)*1e6])
75         idx_count = idx_count+idx_length;
76     end
77 end
78 hold off
79
80 figure()
81 plot(cut_time*1e6, sum(signal_array,1)/N);
82 xlabel('Time(\mu s)',param_label{:})
83 title('Summed Data',param_title{:})
84
85
86 figure()
87 plot(cut_time*1e6, signal_array(1,:))
88 xlabel('Time(\mu s)',param_label{:})
89 % title('Cutted data 1',param_title{:})
90
91 figure()
92 plot(time*1e6, signal)
93 xlabel('Time(\mu s)',param_label{:})
94 ylabel('Voltage (V)',param_label{:})
95
96 a=[6 7]*1e-6;
97 dt = mean(diff(time));
98 t_offset = a(1):10*dt:a(2);
99 vel_avg=zeros(1,length(t_offset));
100 vel_max=zeros(1,length(t_offset));
101

```

```

102    for j=1:length(t_offset)
103        %% Sampling
104        time_sam = zeros(1,N);
105        signal_sam= zeros(1,N);
106        count= t_start+t_offset(j);
107        for i= 1:N
108            idx= max(find(time<count));
109            time_sam(i) = time(idx);
110            signal_sam(i) = signal(idx);
111            count = count + 1/F_prf;
112        end
113
114        %% Bandpass filter after sampling
115        dt_sam=mean(diff(time_sam));
116        fs_sam = 1/dt_sam;
117        fn_sam=fs_sam/2;
118        [c, d]=butter(3, [100 F_prf/2*0.8]/fn_sam, 'bandpass');
119        signal_sam=filter(c,d,signal_sam);
120
121
122        %% plot in time domain
123        % figure()
124        % hold on
125        % plot(time*1e6, signal)
126        % for i=1:length(time_sam)
127        %     scatter(time_sam(i)*1e6, signal_sam(i), 'r')
128        % end
129        % hold off
130        % xlabel('time(\mu s)',param_label{:})
131        % % title(['time offset: ' num2str(t_offset(j)*1e6) '\mu s'],param_title{:})
132        % if save_fig
133        %     saveas(gcf,['time offset= ' num2str(t_offset(j)*1e6) 'us.jpg'])
134        % end
135        %
136        % scatter(time_sam*1e6, signal_sam, 'r')
137        % xlabel('Time(\mu s)',param_label{:})
138        % ylabel('Voltage(V)',param_label{:})
139
140        %% Fourier transform
141        L = length(time_sam);
142        fs=1/mean(diff(time_sam));
143        if padding
144            Lnew = round(fs/fres);
145            P = Lnew-L;
146            if P<0
147                error('P should be bigger than 0. You should increase the ''fres'' ')
148            end
149            signal_sam = [signal_sam zeros(1,P)];
150        else
151            Lnew=L;
152        end
153
154
155        Y= fft(signal_sam);
156        f = fs/Lnew*(0:(Lnew/2));
157        P2 = abs(Y/Lnew);
158        P1 = P2(1:Lnew/2+1);
159        P1(2:end-1) = 2*P1(2:end-1);
160        if dB_scale
161            P1 = mag2db(P1); %dB scale conversion
162        end

```

```

163
164 avg_f = sum(f.*P1)/sum(P1);
165 idx1 = find(P1==max(P1));
166 avg_f = sum(f.*P1)/sum(P1);
167
168
169 % Plot Fourier Transform
170 figure()
171 hold on
172 plot(f,P1)
173 plot(f(1),P1(1))
174 plot(f(2),P1(2))
175 plot(f(3),P1(3))
176 hold off
177 title(['FFT of time offset: ' num2str(t_offset(j)*1e6) '\mu s'],param_title{::})
178 xlabel('frequency',param_label{::})
179 if dB_scale
180     ylabel('Magnitude (dB)')
181 end
182 legend(['Max frequency: ' num2str(f(idx1)) 'Hz'],...
183 [' Velocity: ' num2str(1540*f(idx1)/f0/2*sqrt(2)) 'm/s'],...
184 [' Velocity: ' num2str(1540*avg_f/f0/2*sqrt(2)) 'm/s'])
185
186 if save_fig
187     saveas(gcf,['FFT of time offset= ' num2str(t_offset(j)*1e6) 'us.jpg'])
188 end
189 %% Sweep graph
190 vel_avg(j)=1540*avg_f/f0/2 *sqrt(2);
191 vel_max(j)= 1540*f(idx1)/f0/2* sqrt(2);
192 end
193
194 figure();
195 hold on
196 scatter(t_offset*1e6,vel_avg*100)
197 scatter(t_offset*1e6,vel_max*100)
198 legend('Avg', 'Max')
199 % title(['Swept data index: ' num2str(index)],param_title{::})
200 ylabel('velocity (cm/s)',param_label{::})
201 xlabel('time offset(\mu s)',param_label{::})
202
203 %% Data
204 figure();
205 hold on
206 % scatter(t_offset*1540/sqrt(2)/2*1e3,vel_avg*100)
207 scatter(t_offset*1540/sqrt(2)/2*1e3,vel_max*100)
208 % legend('Avg', 'Max')
209 % title(['Swept data index: ' num2str(index)],param_title{::})
210 ylabel('Velocity (cm/s)',param_label{::})
211 xlabel('Depth (cm)',param_label{::})

```

Code A.7: demodulator.m

```

1 clc;clear;close all;
2
3 %out_t2 = readtable('demodulator/scope_204_2.csv', 'ReadVariableNames', false,
4 %                         'HeaderLines', 1);
4 %out_t4 = readtable('demodulator/scope_204_4.csv', 'ReadVariableNames', false,
5 %                         'HeaderLines', 1);
5 %in_t2 = readtable('demodulator/scope_206_2.csv', 'ReadVariableNames', false,
6 %                         'HeaderLines', 1);

```

```

6 %in_t4 = readtable('demodulator/scope_206_4.csv', 'ReadVariableNames', false,
7 %    ↳ 'HeaderLines', 1);
8 t2 = readtable('demodulator/tek0010ALL.csv', 'ReadVariableNames', false, 'HeaderLines',
9 %    ↳ 20);
10 t1 = readtable('demodulator/tek0011ALL.csv', 'ReadVariableNames', false, 'HeaderLines',
11 %    ↳ 20);
12 t1.Properties.VariableNames = ["time", "ch1", "ch2", "ch3", "ch4"];
13 t2.Properties.VariableNames = ["time", "ch1", "ch2", "ch3", "ch4"];
14 %%%
15 figure(1)
16 hold on
17 %title('Demodulated signal $f_d=1~\mathrm{kHz}$', 'interpreter', 'latex');
18 plot(t1.time, movmean(t1.ch1, 200), 'k')
19 plot(t1.time, movmean(t1.ch4, 200), 'k-.', 'Color', [0.5 0.5 0.5])
20 plot(t1.time, t1.ch2, '--k')
21 plot(t1.time, t1.ch3, ':k')
22 grid minor
23 set(gca, 'TickLabelInterpreter', 'latex')
24 ylabel('Voltage [V]', 'interpreter', 'latex');
25 xlabel('Time [sec]', 'interpreter', 'latex');
26 %xlim([-2e-4 2e-4])
27 legend('Demodulated signal $I$ 1~kHz', 'Demodulated signal $Q$ 1~kHz', 'Differential signal
28 %    ↳ A 5.001~MHz', 'Differential signal B 5.001~MHz
29 %    ↳ 180\angle', 'Location', 'best', 'interpreter', 'latex')
30 hold off
31 % create a new pair of axes inside current figure
32 axes('position', [.65 .5125 .25 .25])
33 hold on
34 box on % put box around new pair of axes
35 indexOfInterest = (t1.time < 1.5e-7) & (t1.time > -1.5e-7); % range of t near
36 % perturbation
37 %indexOfInterest = (t < 11*pi/8) & (t > 9*pi/8)
38 %indexOfInterestp = indexOfInterest.';
39 plot(t1.time(indexOfInterest), t1.ch2(indexOfInterest), '--k') % plot on new axes
40 plot(t1.time(indexOfInterest), t1.ch3(indexOfInterest), ':k') % plot on new axes
41 set(gca, 'TickLabelInterpreter', 'latex')
42 axis tight
43 grid on
44 ylim([min(t1.ch2) max(t1.ch2)])
45 hold off
46 %%%
47 figure(2)
48 %subplot(2, 1, 1)
49 hold on
50 %title('Received signal $f_d=5.3~\mathrm{MHz}$', 'interpreter', 'latex');
51 plot(t2.time, movmean(t2.ch2, 10), ':k')
52 plot(t2.time, movmean(t2.ch3, 10), '--k')
53 plot(t2.time, t2.ch4, 'k')
54 grid minor
55 ylabel('Voltage [V]', 'interpreter', 'latex');
56 xlabel('Time [sec]', 'interpreter', 'latex');
57 xlim([-0.15e-6 0.15e-6])
58 set(gca, 'TickLabelInterpreter', 'latex')
59 legend('Differential signal A 5.001~MHz', 'Differential signal B 5.001~MHz
60 %    ↳ 180\angle', 'Local Oscillator $L_{\mathrm{OSC4}}=$
61 %    ↳ 20~MHz', 'Location', 'best', 'interpreter', 'latex')
62 hold off
63 %subplot(2, 1, 2)
64 %hold on
65 %title('Local Oscillator signal
66 %    ↳ $L_{\mathrm{osc4}}=20~\mathrm{MHz}$', 'interpreter', 'latex');

```

```

58 %plot(t2.time,t2.ch3,'k')
59 %plot(t2.time,t2.ch4,'k')
60 %grid minor
61 %ylabel('Voltage [V]');
62 %xlabel('Time [sec]');
63 %xlim([3e-6 3.2e-6])
64 %hold off

```

Code A.8: preamplifier.m

```

1 clc;clear;close all;
2
3 %out_t2 = readtable('scope_208_2.csv', 'ReadVariableNames', false, 'HeaderLines', 1);
4 %out_t4 = readtable('scope_208_4.csv', 'ReadVariableNames', false, 'HeaderLines', 1);
5 %in_t2 = readtable('scope_208_1.csv', 'ReadVariableNames', false, 'HeaderLines', 1);
6 t1 = readtable('tek0012ALL.csv', 'ReadVariableNames', false, 'HeaderLines', 20);
7 t1.Properties.VariableNames = ["time","input","outa","outb"];
8
9
10 figure(1)
11 subplot(2,1,1)
12 title('AC coupled input signal','interpreter','latex');
13 hold on
14 plot(t1.time,t1.input,'k','DisplayName','Input')
15 %plot(out_t4.Var1,out_t4.Var2,'k','DisplayName','LOP')
16 grid minor
17 ylabel('Voltage [V]', 'interpreter','latex')
18 set(gca,'TickLabelInterpreter','latex')
19 %xlabel('Time [sec]');
20 xlim([-0.2e-6 0.2e-6])
21 %l=legend('show');
22 hold off
23 subplot(2,1,2)
24 %figure(2)
25 hold on
26 title('Amplified differential signal with DC coupling','interpreter','latex');
27 plot(t1.time,movmean(t1.outa,10), 'k')
28 plot(t1.time,movmean(t1.outb,10), '--k')
29 grid minor
30 ylabel('Voltage [V]', 'interpreter','latex');
31 xlabel('Time [sec]', 'interpreter','latex');
32 set(gca,'TickLabelInterpreter','latex')
33 xlim([-0.2e-6 0.2e-6])
34 legend('Differential signal A 5.001-MHz','Differential signal B 5.001-MHz
35 %set(l,'Interpreter','Latex')
36 %set(l,'Interpreter','Latex');

```

Code A.9: sampler.m

```

1 clc;clear;close all;
2
3 t1 = readtable('tek0001ALL.csv', 'ReadVariableNames', false, 'HeaderLines', 20);
4 t1.Properties.VariableNames = ["time","output","pulse","input"];
5
6 figure(1)
7 subplot(3,1,1)

```

```

8 hold on
9 title('Demodulated input waveform','interpreter','latex');
10 plot(t1.time,t1.input,'k')
11 set(gca,'TickLabelInterpreter','latex')
12 grid minor
13 ylabel('Voltage [V]','interpreter','latex');
14 %xlabel('Time [sec]');
15 hold off
16 subplot(3,1,2)
17 hold on
18 title('Sample pulse input','interpreter','latex');
19 plot(t1.time,t1.pulse,'k')
20 set(gca,'TickLabelInterpreter','latex')
21 grid minor
22 ylabel('Voltage [V]','interpreter','latex');
23 %xlabel('Time [sec]');
24 hold off
25 subplot(3,1,3)
26 hold on
27 title('Sampled demodulated output','interpreter','latex');
28 plot(t1.time,t1.output,'k')
29 set(gca,'TickLabelInterpreter','latex')
30 grid minor
31 ylabel('Voltage [V]','interpreter','latex');
32 xlabel('Time [sec]','interpreter','latex');
33 hold off

```

Code A.10: transmitter.m

```

1 clc;clear;close all;
2
3 t1 = readtable('scope_202_1.csv', 'ReadVariableNames', false, 'HeaderLines', 1);
4 t2 = readtable('scope_202_2.csv','ReadVariableNames', false, 'HeaderLines', 1);
5 t4 = readtable('scope_202_4.csv','ReadVariableNames', false, 'HeaderLines', 1);
6
7 figure(1)
8 subplot(2,1,1)
9 hold on
10 title('PWM Input to gate driver','interpreter','latex');
11 plot(t1.Var1,movmean(t1.Var2,25),'k','DisplayName','INA')
12 plot(t2.Var1,movmean(t2.Var2,25),':k','DisplayName','INB')
13 set(gca,'TickLabelInterpreter','latex')
14 grid minor
15 ylabel('Voltage [V]','interpreter','latex');
16 xlabel('Time [sec]','interpreter','latex');
17 legend('show','interpreter','latex')
18 hold off
19 subplot(2,1,2)
20 hold on
21 title('Power stage output voltage','interpreter','latex');
22 plot(t4.Var1,t4.Var2,'k')
23 set(gca,'TickLabelInterpreter','latex')
24 grid minor
25 ylabel('Voltage [V]','interpreter','latex');
26 xlabel('Time [sec]','interpreter','latex');
27 hold off

```

Code A.11: vna.m

```

1 clc;clear;close all;
2 %
3 mag = readtable('TRACE01.CSV','ReadVariableNames', false, 'HeaderLines', 3);
4 phase = readtable('TRACE02.CSV','ReadVariableNames', false, 'HeaderLines', 3);
5 mag.Properties.VariableNames = ["x","y","zero"];
6 phase.Properties.VariableNames = ["x","y","zero"];
7
8 figure()
9 sgtitle('\textbf{Bode Plot}', 'interpreter', 'latex');
10 subplot(2,1,1)
11 hold on
12 %plot(mag.x,mag.y, 'k')
13 semilogx(mag.x,mag.y, 'k')
14 grid minor
15 set(gca,'TickLabelInterpreter','latex')
16 ylabel('Magnitude [dB]', 'interpreter', 'latex');
17 xlabel('Frequency [Hz]', 'interpreter', 'latex');
18 %xlim([900e3 11e6])
19 ylim([-15 1])
20 hold off
21 subplot(2,1,2)
22 hold on
23 %plot(phase.x,unwrap(phase.y), 'k')
24 semilogx(phase.x,unwrap(phase.y), 'k')
25 grid minor
26 set(gca,'TickLabelInterpreter','latex')
27 ylabel('Phase [deg]', 'interpreter', 'latex');
28 xlabel('Frequency [Hz]', 'interpreter', 'latex');
29 xlim([0 14e6])
30 hold off

```

Code A.12: bpf datasheet s21.m

```

1 clc;clear;close all;
2 %
3 import_table = readtable('BPF-C4R5+_Plus25DegC.txt','ReadVariableNames', false,
4                           'HeaderLines', 10);
4 import_table.Properties.VariableNames = ["freq", "mag_s11", "phase_s11", "mag_s21",
5                           "phase_s21", "mag_s12", "phase_s12", "mag_s22", "phase_s22"];
5
6 freq = import_table(:, 'freq');
7 mag = import_table(:, "mag_s21");
8 phase = import_table(:, "phase_s21");
9 freq = freq.*1e6;
10 phase = unwrap(phase);
11 %
12 figure()
13 sgtitle('\textbf{Bode Plot}', 'interpreter', 'latex');
14 subplot(2,1,1)
15 hold on
16 semilogx(freq,mag, 'k')
17 grid on
18 set(gca,'TickLabelInterpreter','latex')
19 ylabel('Magnitude [dB]', 'interpreter', 'latex');
20 xlabel('Frequency [Hz]', 'interpreter', 'latex');
21 xlim([0 14e6])
22 ylim([-15 1])

```

```

23 hold off
24 subplot(2,1,2)
25 hold on
26 semilogx(freq,phase,'k')
27 grid on
28 set(gca,'TickLabelInterpreter','latex')
29 ylabel('Phase [deg]', 'interpreter', 'latex');
30 xlabel('Frequency [Hz]', 'interpreter', 'latex');
31 xlim([0 14e6])
32 hold off

```

Code A.13: switch.m

```

1 clc;clear;close all;
2
3 t2 = readtable('230317	scope_114_2.csv', 'ReadVariableNames', false, 'HeaderLines', 1);
4 t4 = readtable('230317	scope_114_4.csv', 'ReadVariableNames', false, 'HeaderLines', 1);
5
6 figure(1)
7 subplot(2,1,1)
8 hold on
9 title('Transmitted signal', 'interpreter', 'latex');
10 plot(t4.Var1,t4.Var2, 'k')
11 grid minor
12 ylabel('Voltage [V]', 'interpreter', 'latex');
13 %xlabel('Time [sec]', 'interpreter', 'latex');
14 set(gca,'TickLabelInterpreter','latex')
15 xlim([9.8942e-5 1.21698e-4])
16 hold off
17 subplot(2,1,2)
18 hold on
19 title('Received signal', 'interpreter', 'latex');
20 plot(t2.Var1,t2.Var2, 'k')
21 grid minor
22 ylabel('Voltage [V]', 'interpreter', 'latex');
23 xlabel('Time [sec]', 'interpreter', 'latex');
24 set(gca,'TickLabelInterpreter','latex')
25 xlim([9.8942e-5 1.21698e-4])
26 hold off

```

B Simulation Models

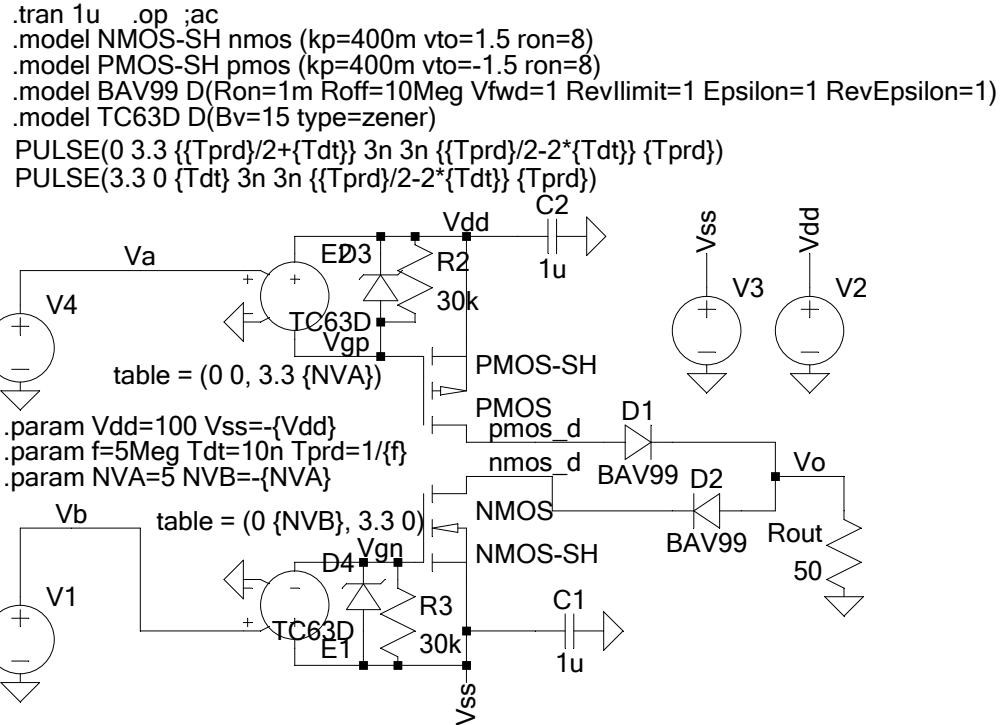


Figure B.1: LTspice model of transmitter

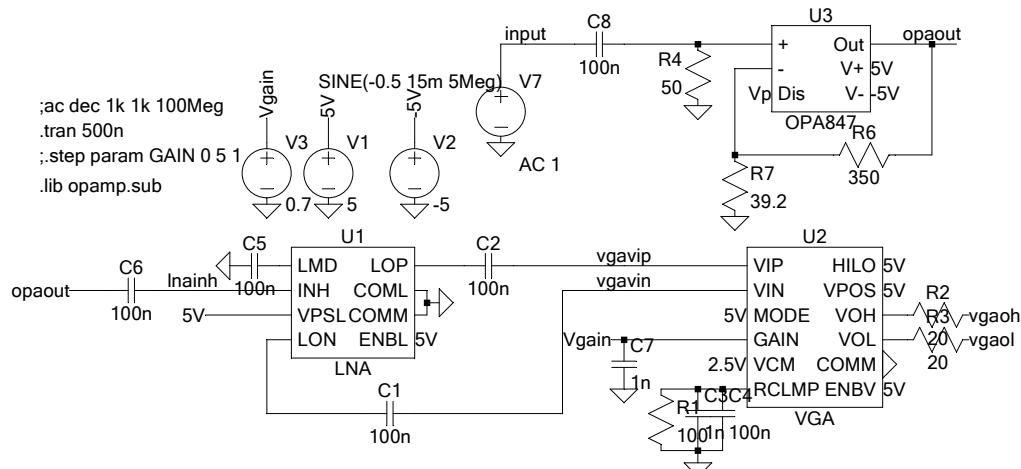


Figure B.2: LTspice model of preamplifier

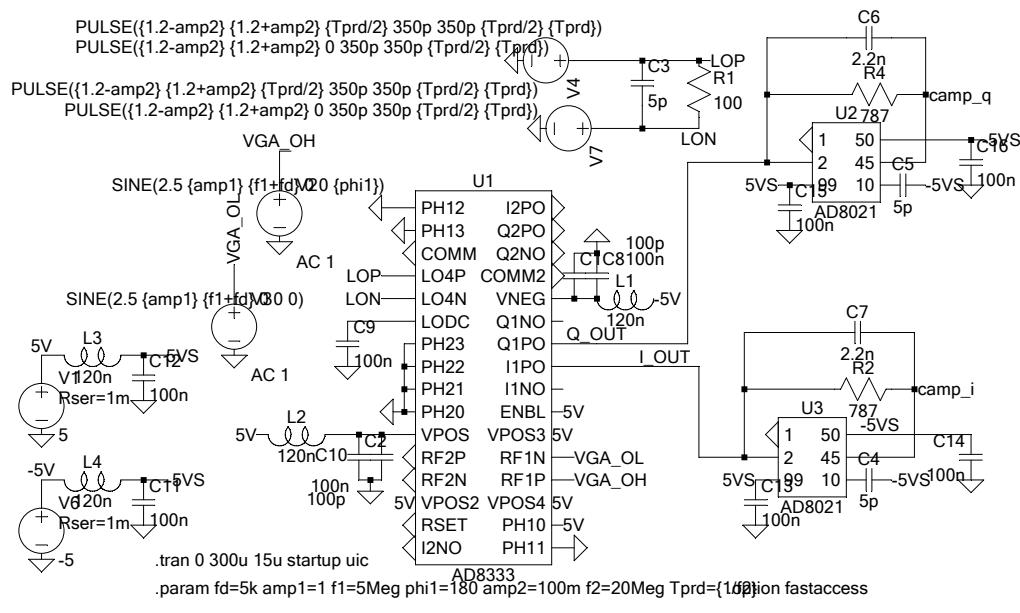


Figure B.3: LTspice model of demodulator

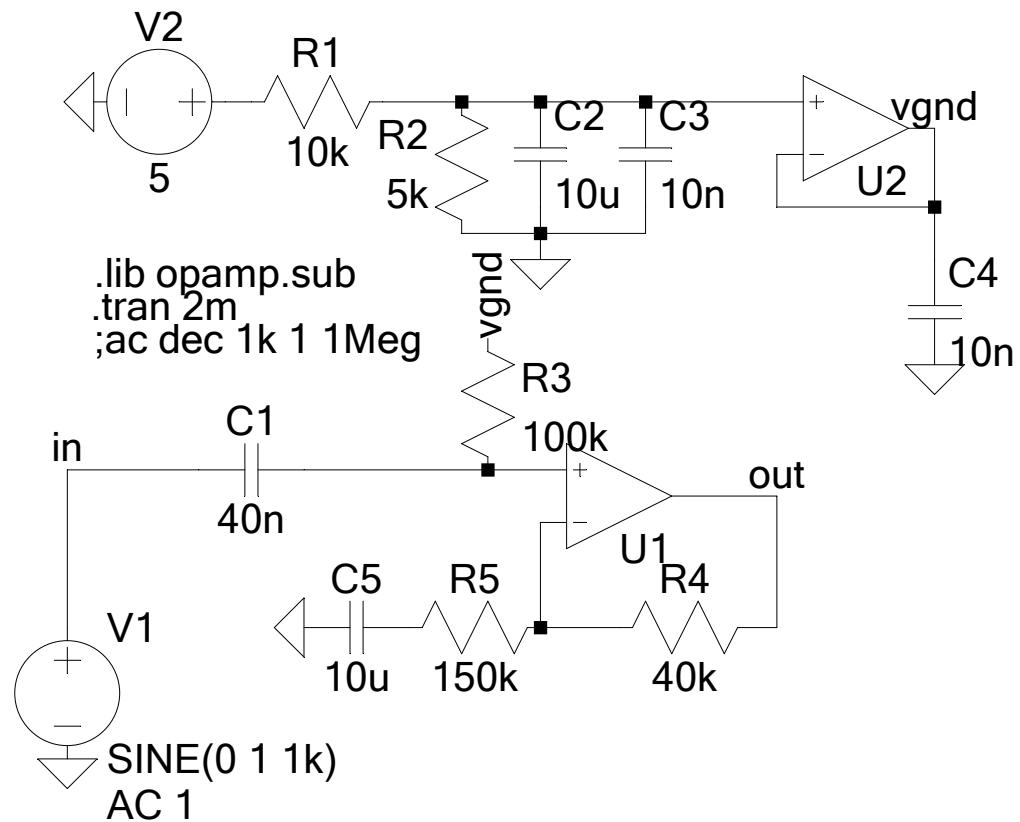


Figure B.4: LTspice model of PRF filter

C Circuit Schematics

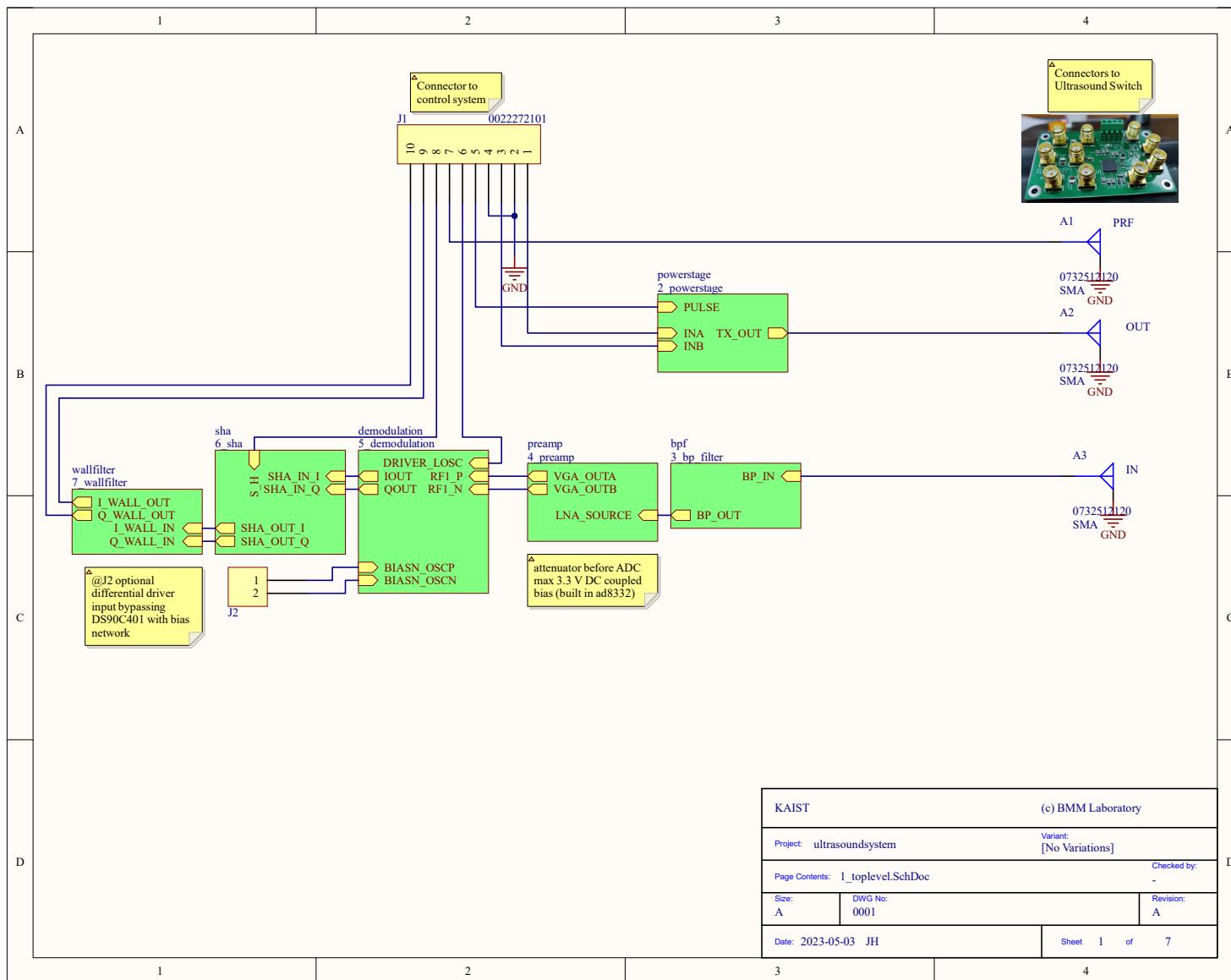


Figure C.1: AFE Top Level

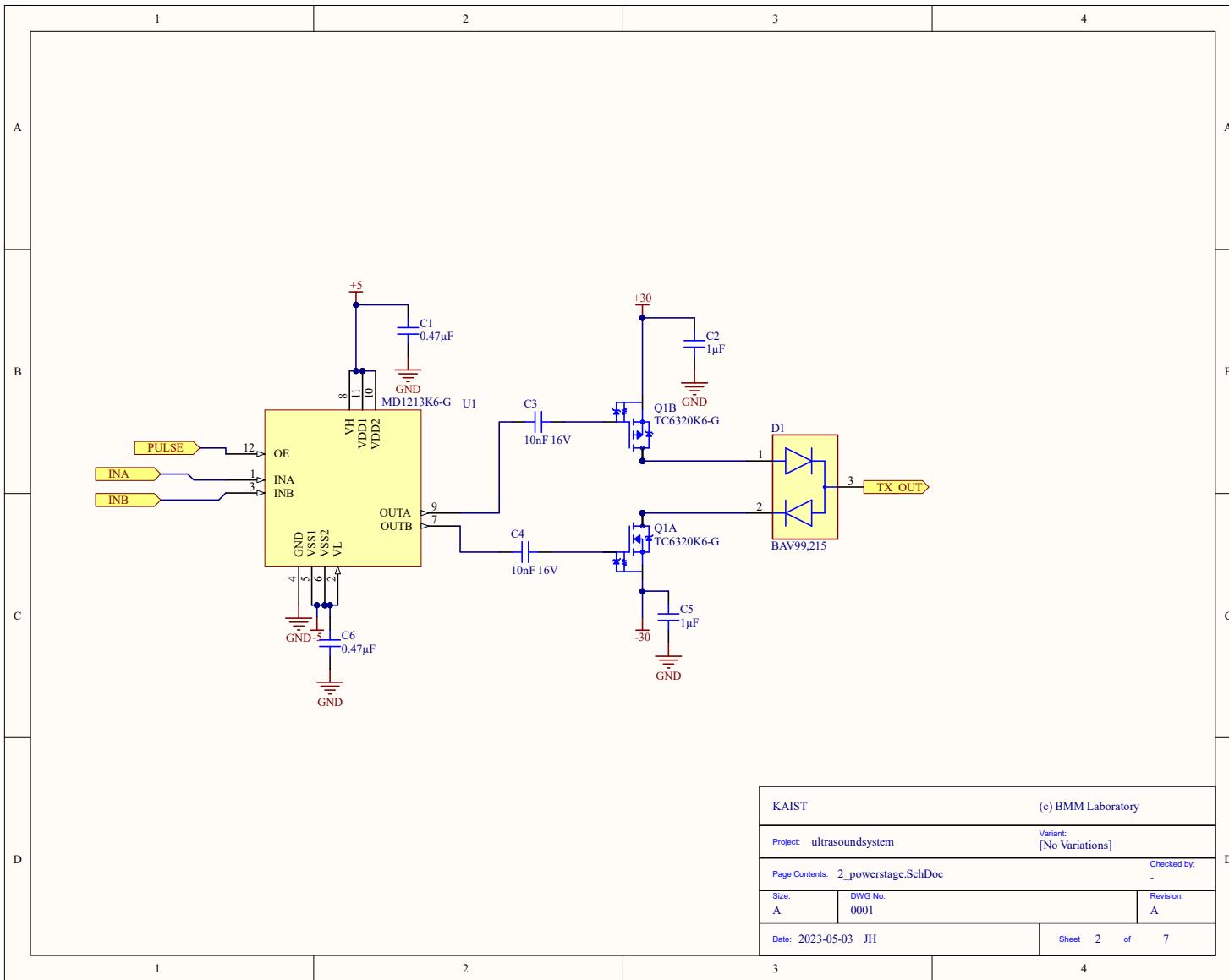


Figure C.2: AFE Power Stage

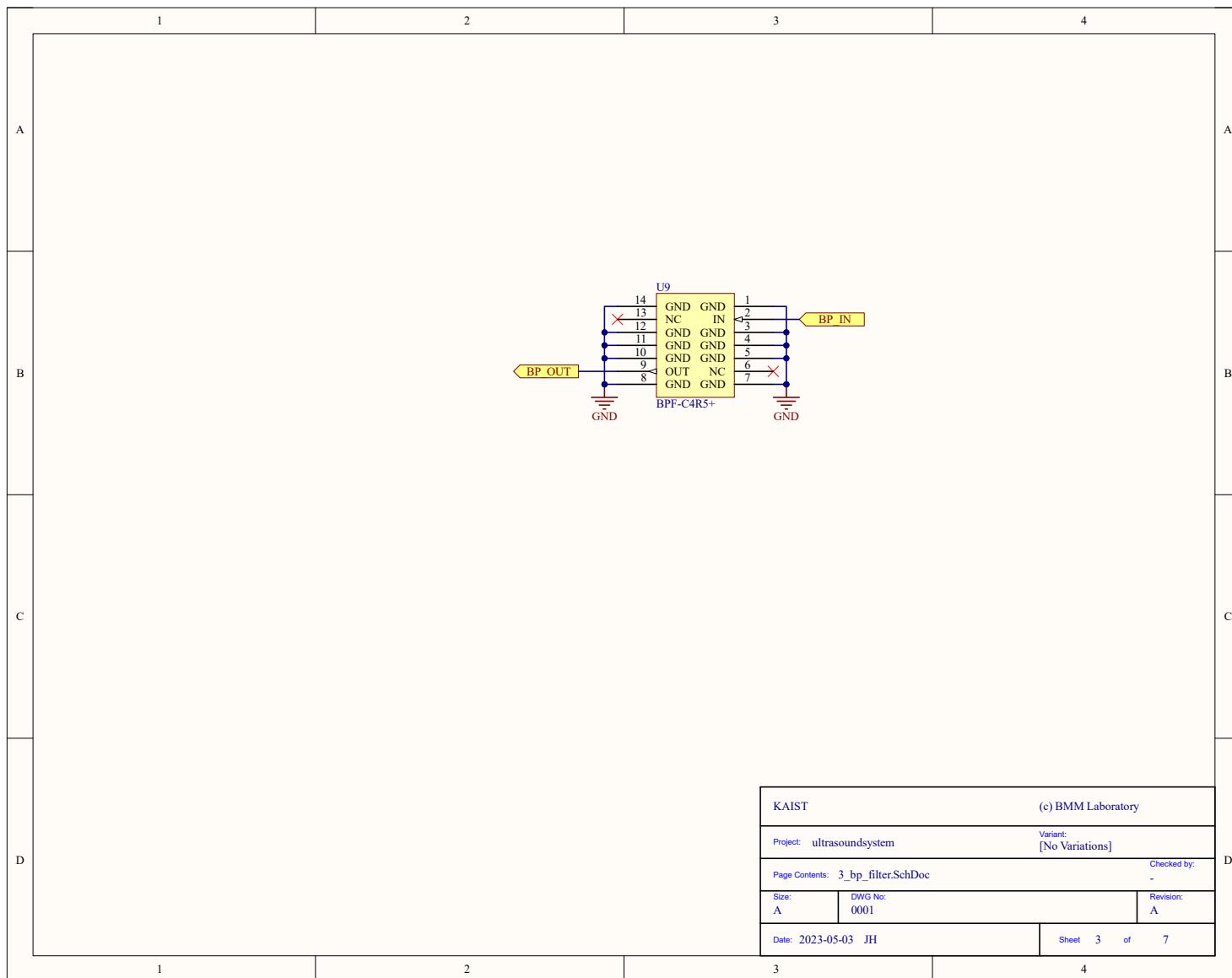


Figure C.3: AFE BPF

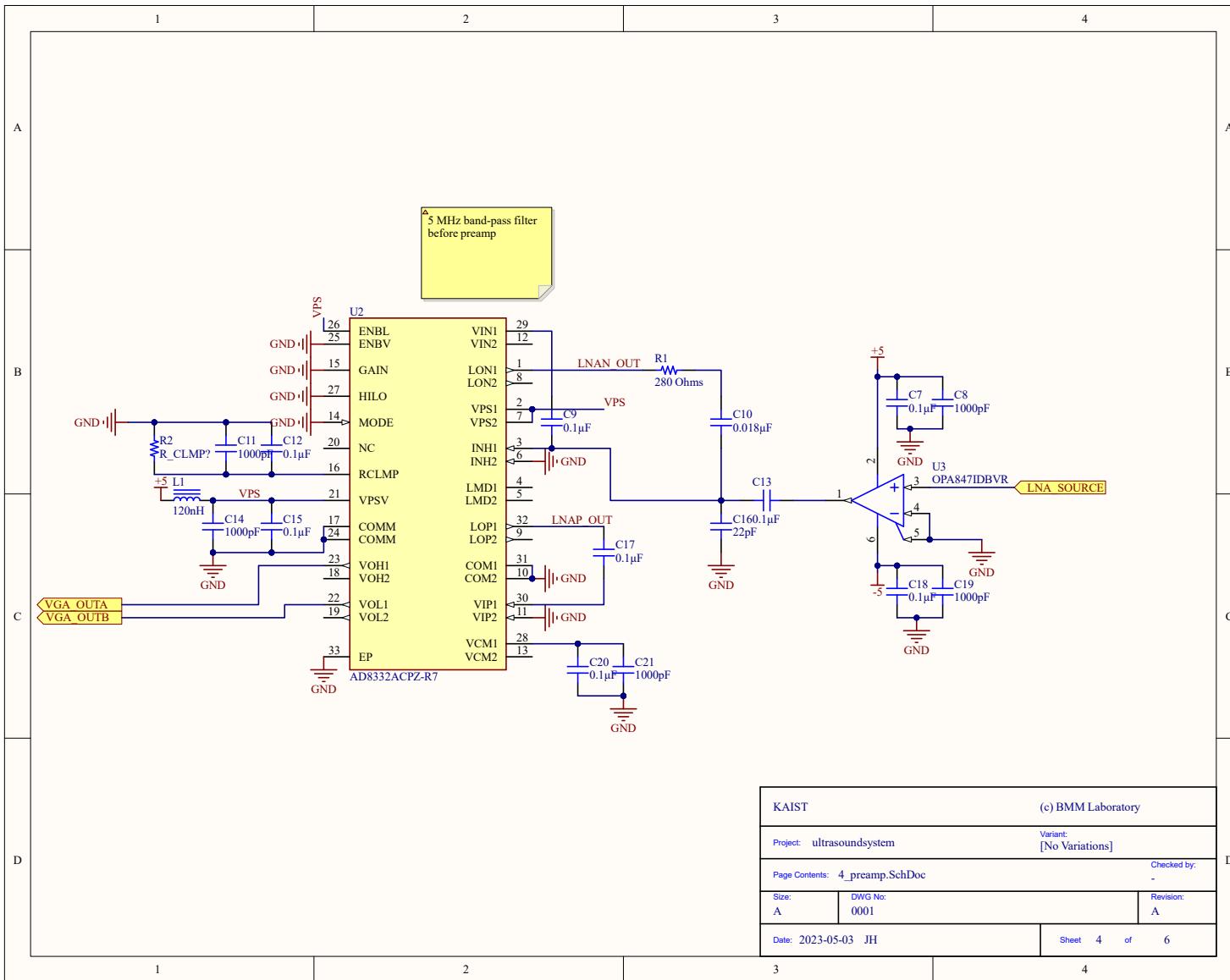


Figure C.4: AFE Preamplifier

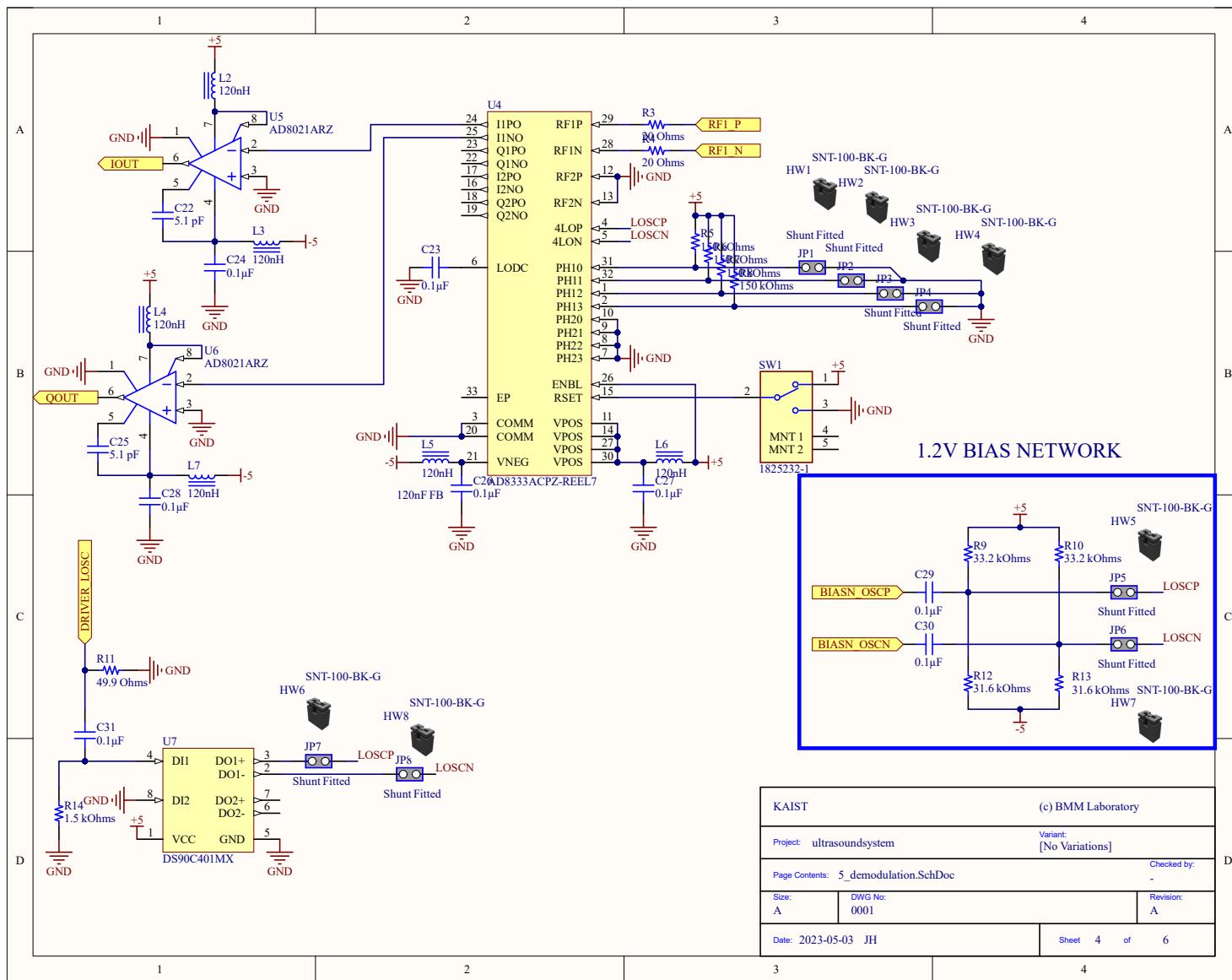


Figure C.5: AFE Demodulator

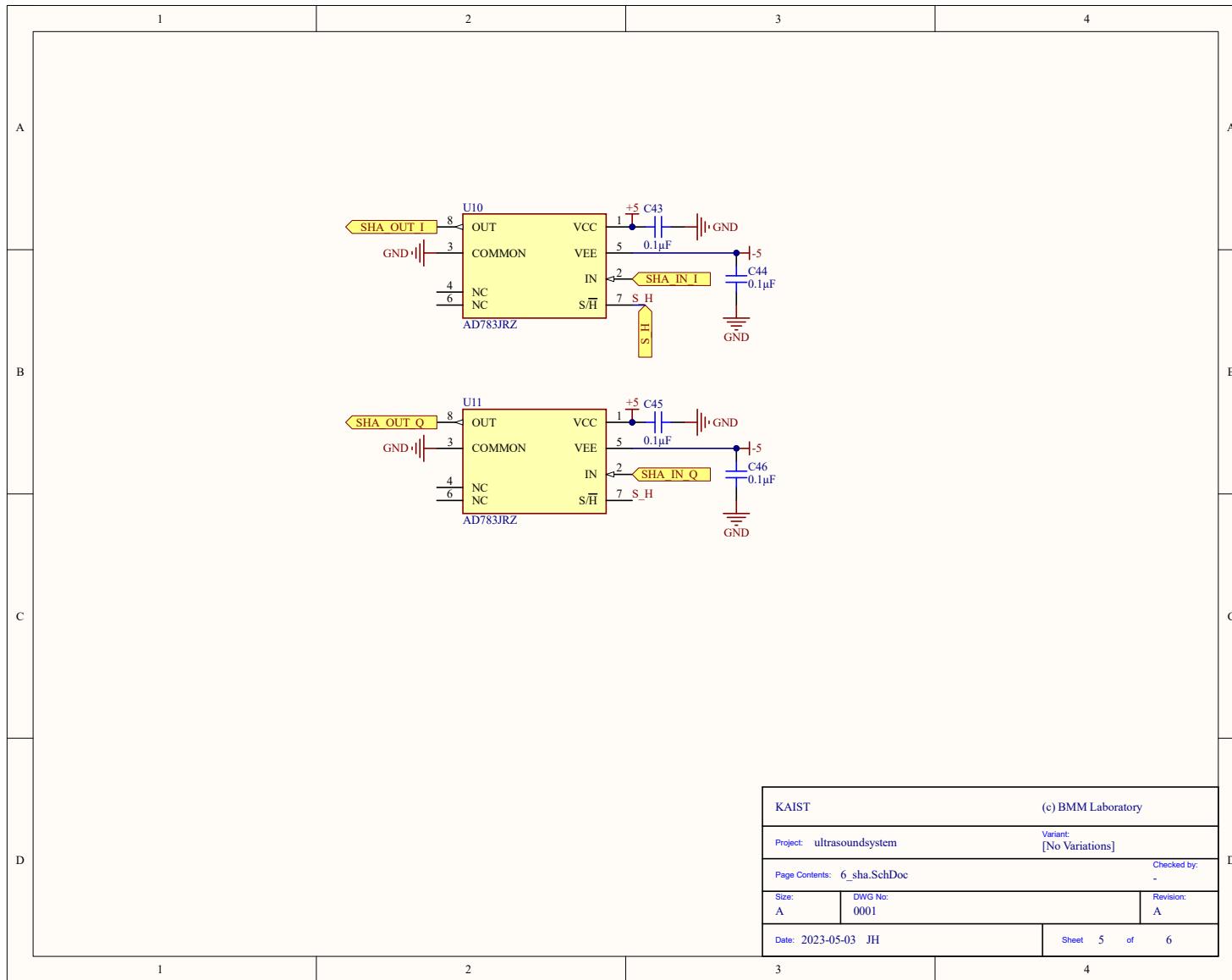


Figure C.6: AFE Sample and Hold Amplifier

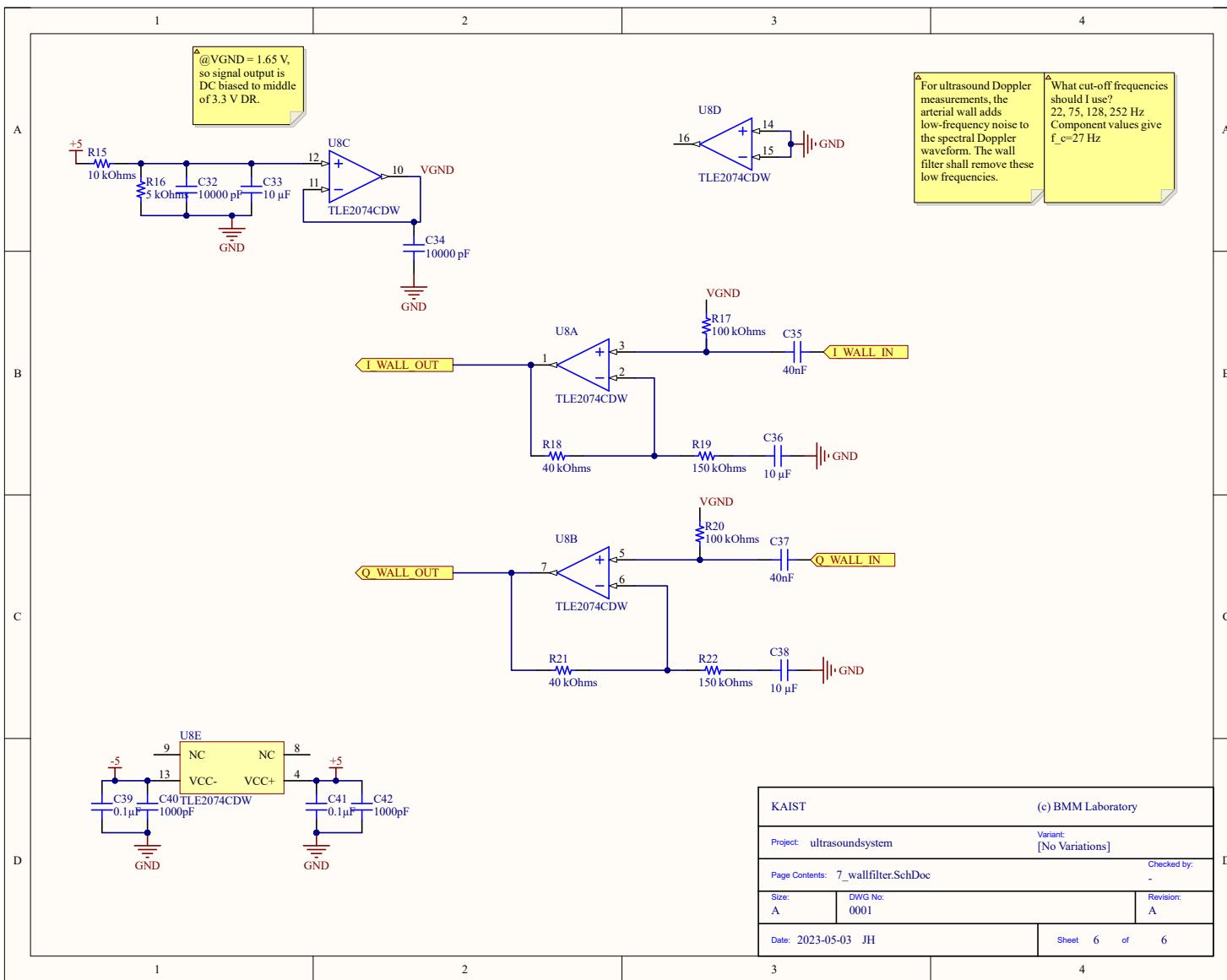
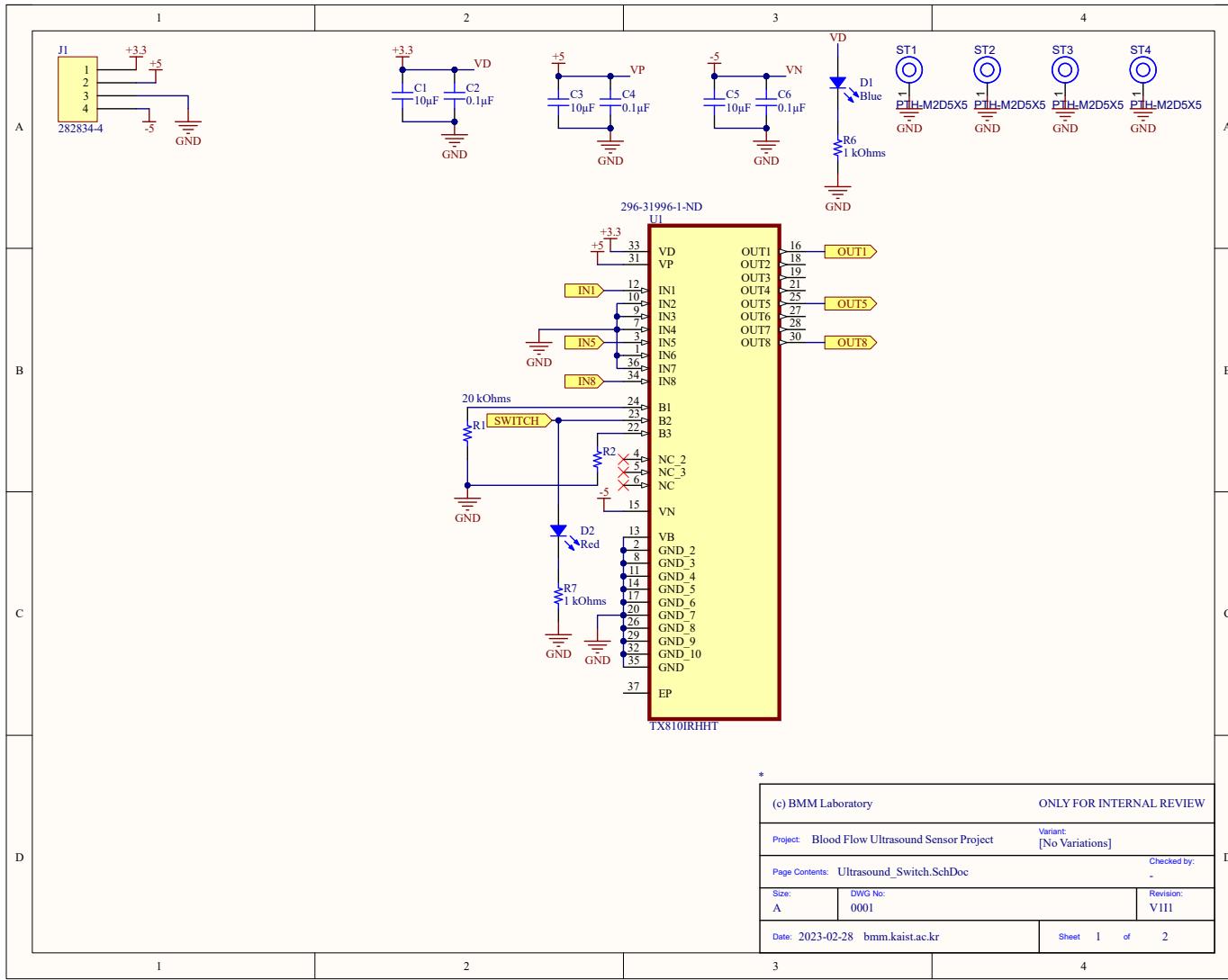


Figure C.7: AFE Wall Filter



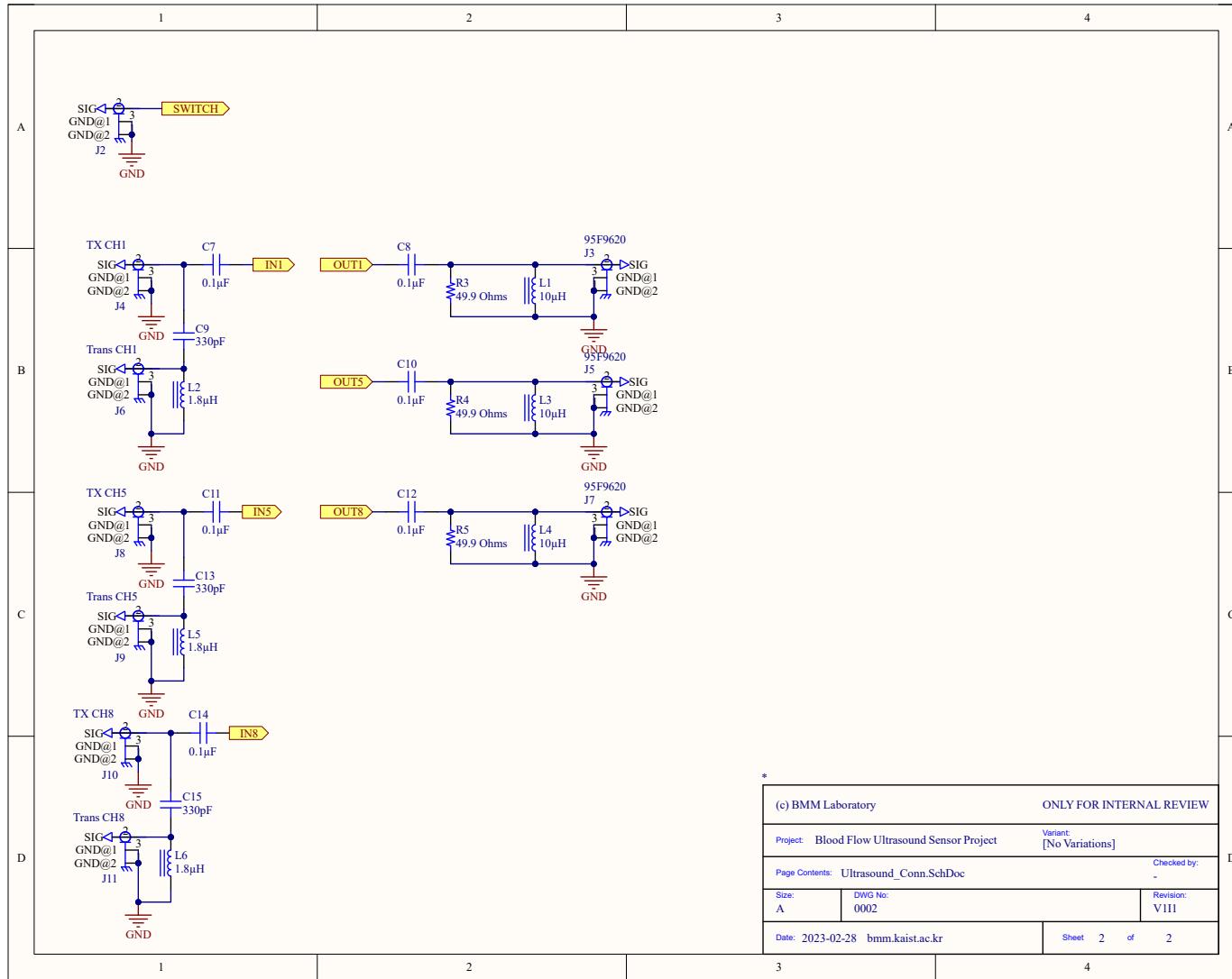


Figure C.9: UltrasoundSwitch Schematic B

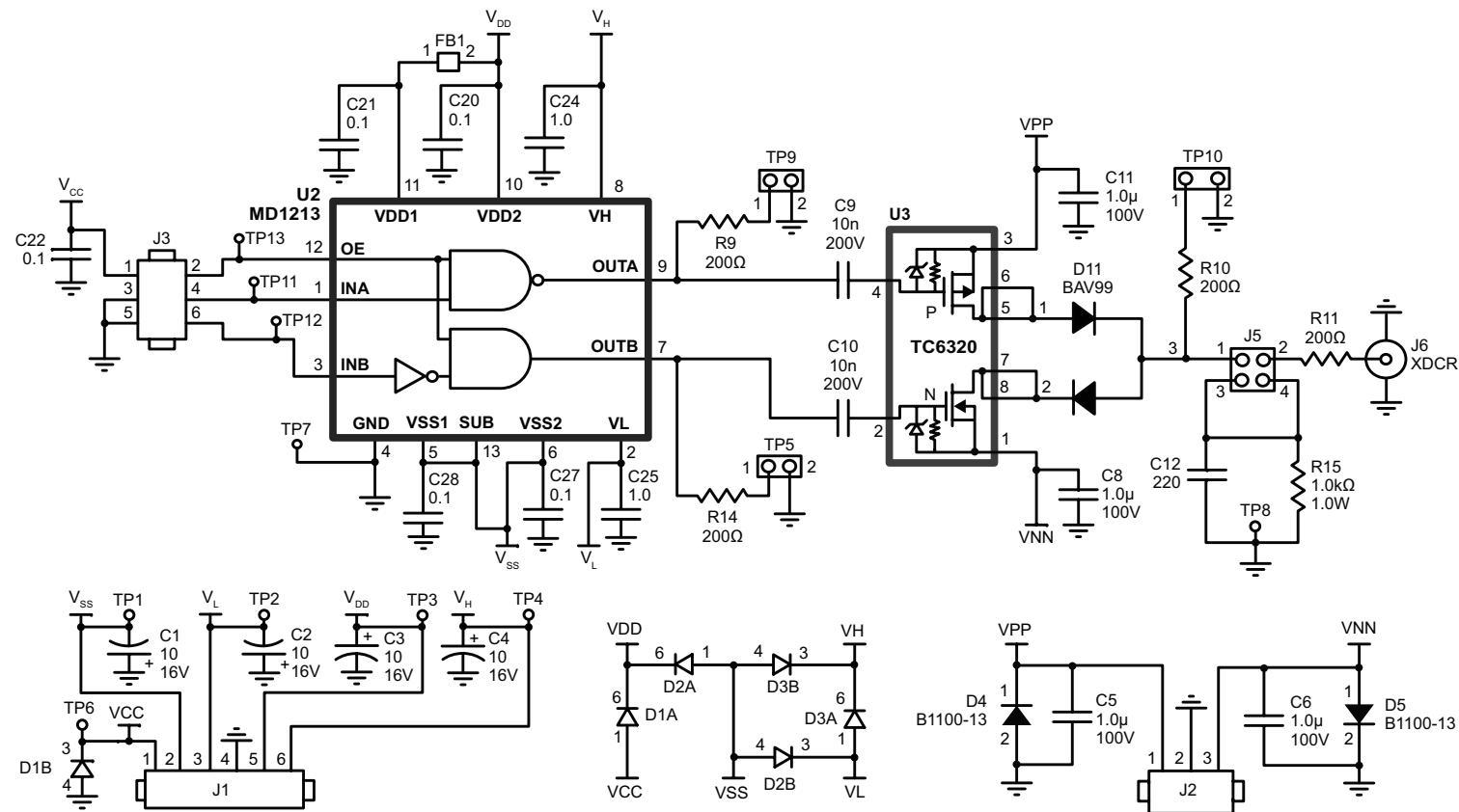


Figure C.10: MD1213DB1 Transmitter Schematic

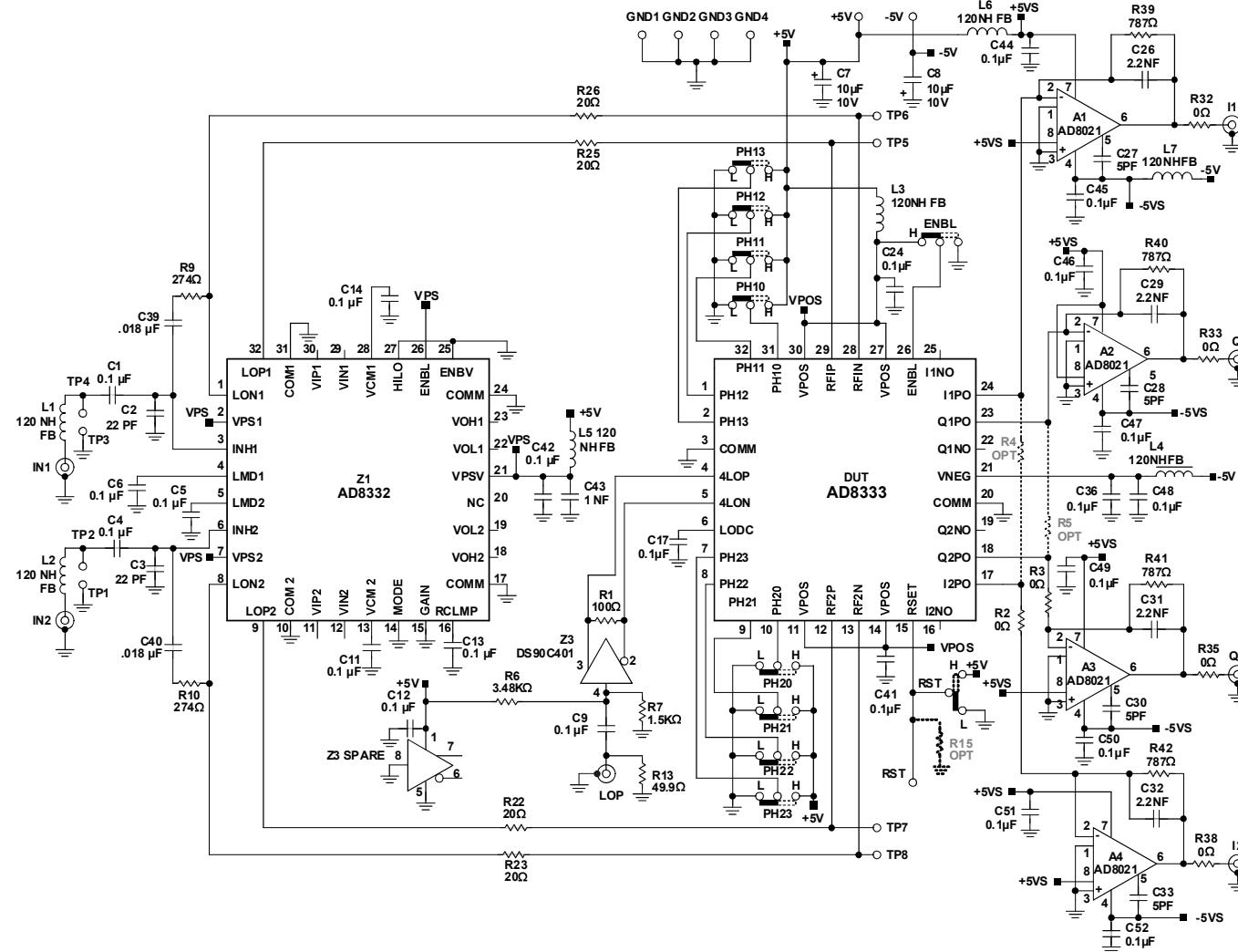


Figure C.11: AD8332 Preamplifier, AD8333 IQ Demodulator Schematic

D Circuit CAD Assembly Documentation

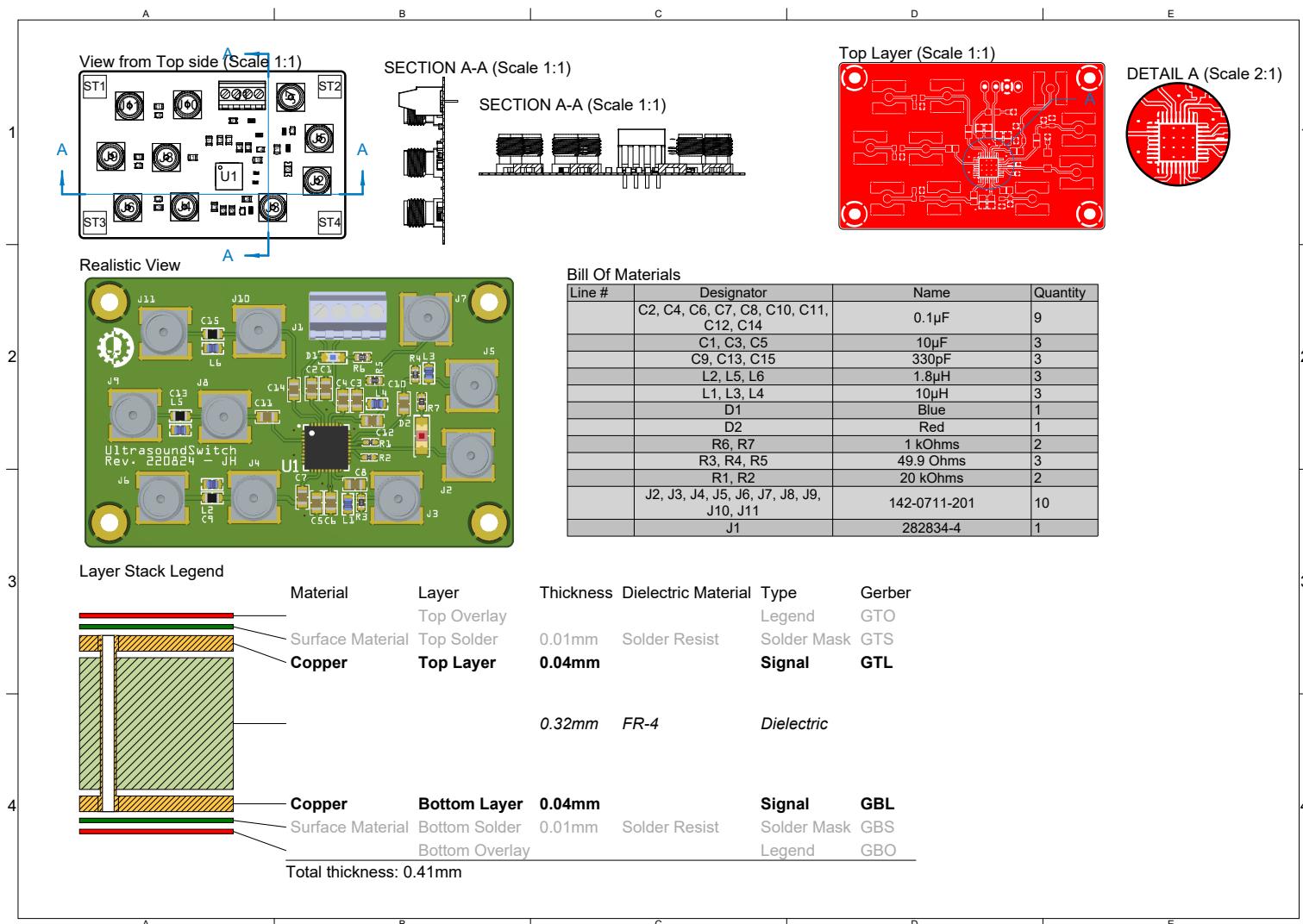


Figure D.1: UltrasoundSwitch Assembly Information

E Instruments

Table E.1: List of instruments used for solder work

Function	Manufacturer	Model
Visual inspection microscope	Leica	A60
Manual soldering	Weller	WX2
Heat gun	Thermaltronics	TMT-HA600-2
Solder paste	Chip Quik	SMD291AX250T3
Solder flux	Chip Quik	SMD291NL
Reflow oven	Puhui	T-962A
DMM	Fluke	175

Table E.2: List of instruments used in experiments

Function	Manufacturer	Model
DCPS 1	RIGOL	DP832A 200W
DCPS 2	Keysight	E3631A 80W
Function generator 1	Keysight	33500B
Function generator 2	Tektronix	AFG3102
DMM	Fluke	175
Transducer (PZT)	HAGISONIC	M715-SB-S 204 5 MHz
Transducer (CMUT)	BMM Creation	6ch 3.3 MHz C.F.
RF Amplifier	Tomco	BT00100-AlphaS-CW
Oscilloscope 1	Keysight	DSO-X 2024A
Oscilloscope 2	Tektronix	MSO4054
Physiological simulator	CIRS	Doppler String Phantom 043A
Vector Network Analyzer	Agilent	E5071B ENA Series Network Analyzer

