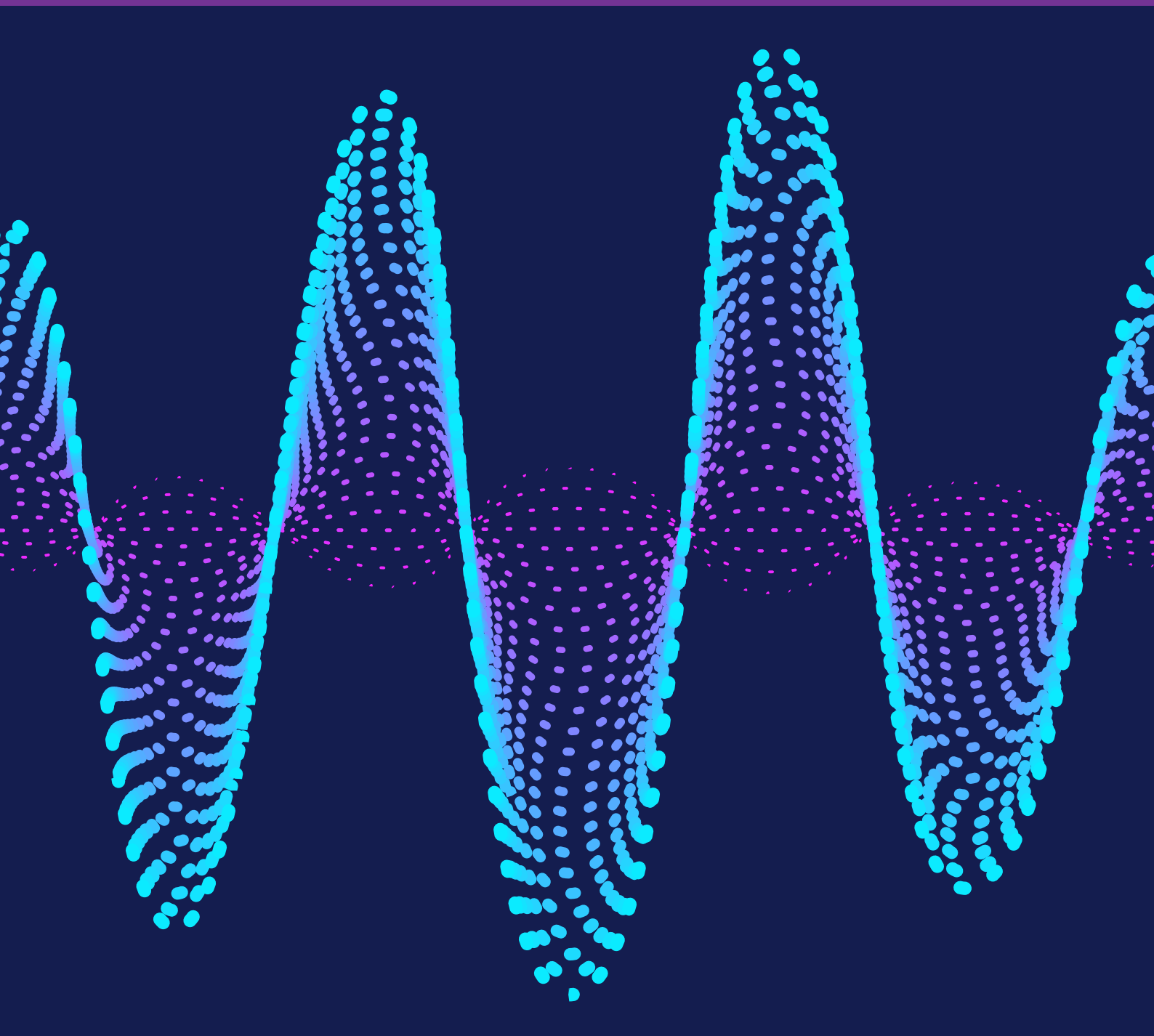
DTU Department of Electrical Engineering
DTU Elektro, Institut for Elektroteknologi
KAIST EE
한국과학기술원 전기 및 전자공학부

Portable Ultrasound System for Blood Velocity Estimation

Jeppe Hinrichs

Master Thesis



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

Contents

List of Figures

List of Tables

List of Source Codes

Acronyms

Notation	Description
ADC	analogue-to-digital converter
AFE	analogue front end
BLE	bluetooth low energy
BP	band-pass
CMUT	capacitive micromachined ultrasound transducer
CPU	central processing unit
DSP	digital signal processor
DTS	device tree source
DTSI	device tree source include
ELF	executable and linkable format
FET	field-effect transistor
HP	high-pass
I/O	input/output
IC	integrated circuit
IoT	Internet of Things
LNA	low noise amplifier
LSB	least significant bit
MCU	microcontroller unit
MOSFET	metal-oxide-semiconductor field-effect transistor
PRF	pulse repetition frequency
PZT	lead circonate titanate
RAM	random access memory
RTOS	real-time operating system
SoC	system-on-a-chip
VGA	variable gain amplifier
VNA	vector network analyzer

Glossary

Notation	Description
flashing	Refers to the process of programming the non-volatile memory of a microcontroller
HIGH	Logic high voltage level
Internet of Things	A term used to describe physical things equipped with sensors, computing power, software, and other technologies that communicate data with other systems and devices across communication networks such as the Internet
LOW	Logic low voltage level
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

1 Synthesis

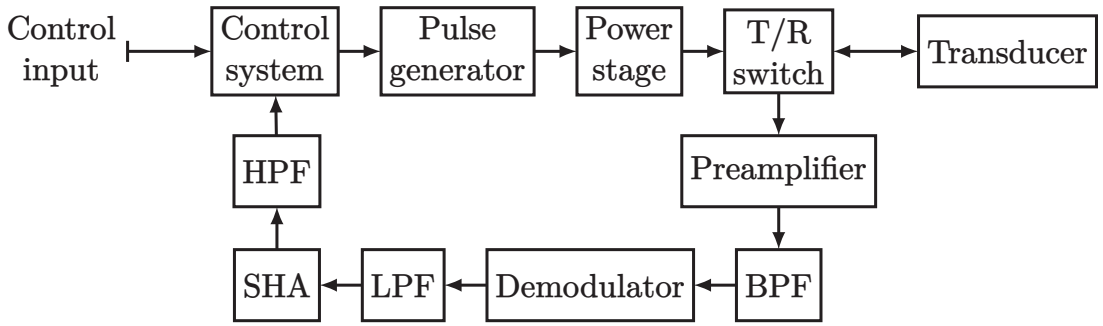


Figure 1.1: Simplified overview of the entire system

A simplified overview of the entire system can be seen in fig. 1.1. 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.

1.1 Control System

The choice of platform for the control system is a microcontroller. 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 input/output (*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 STM32F411RE, 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 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.

1.1.1 Development Environment

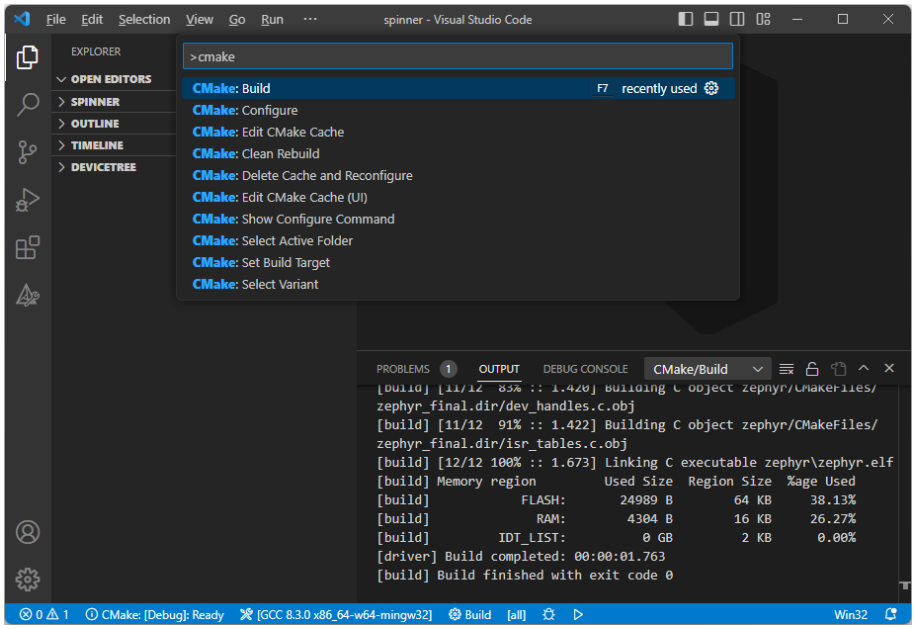


Figure 1.2: VSCode editor with CMake Tools active, displaying available tasks

Visual Studio Code (VSCode) is an open-source code editor developed by Microsoft. It is designed to be highly customizable and efficient, with a wide range of features and extensions that allow developers to customize their workflows and improve their productivity. VSCode works in synergy with CMake through an extension. CMake is a cross-platform build system that helps developers manage the build process for their projects. It is used to generate build files for different platforms and build systems, to build projects on a wide range of platforms. When using CMake with VSCode, the basic idea is to use the CMake extension for VSCode to generate the appropriate build files for the target platform, and then use the VSCode tasks and debugging capabilities to build and debug the project. After setup of VSCode, the CMake Tools [15] extension can be found in the VSCode Marketplace. The CMake Tools extension allows you to create, configure and build CMake projects from within VSCode. Once the extension is installed, you can create a new CMake project and configure it by specifying the path to the `CMakeLists.txt` file and other settings like the target platform and build configuration. After configuring the project, the VSCode tasks are provided to build and run the project. These tasks are defined in the `tasks.json` file, which can be customized to specify the build command and other options. Examples of tasks are build the project, cleaning the build directory, or running tests. The available

CMake tasks are shown in fig. 1.2. In addition to building and running the project, VSCode allows for debugging capabilities to debug code.

1.1.2 Zephyr

Zephyr is an open-source *RTOS* designed to be lightweight and run on a wide range of devices, from *MCU*s with as little as 20 kB of random access memory (*RAM*) to more powerful systems with multiple processors. Zephyr is designed to be modular and scalable, with a focus on security and low power consumption. It includes support for a wide range of hardware architectures, including ARM Cortex-M, x86, and RISC-V, and it can be used with a variety of development boards and microcontrollers. One of the key features of Zephyr is its ability to run on very small devices with limited resources. It includes support for various networking protocols, such as bluetooth low energy (*BLE*), IPv4, and IPv6, which makes it well-suited for use in Internet of Things (*IoT*) applications. Zephyr is developed as part of the Linux Foundation's Zephyr Project, and it is widely used in the development of IoT and embedded systems.

Build System

To build an application with the Zephyr kernel, you use CMake which has two phases - configuration and build. During configuration phase, CMake executes the `CMakeLists.txt` build scripts to generate an internal model of the Zephyr build. Starting with device tree source (*DTS*) and device tree source include (*DTSI*) and then using the device-tree nodes and Kconfig to configure the set of build files for ninja. Seen in fig. 1.3 is the

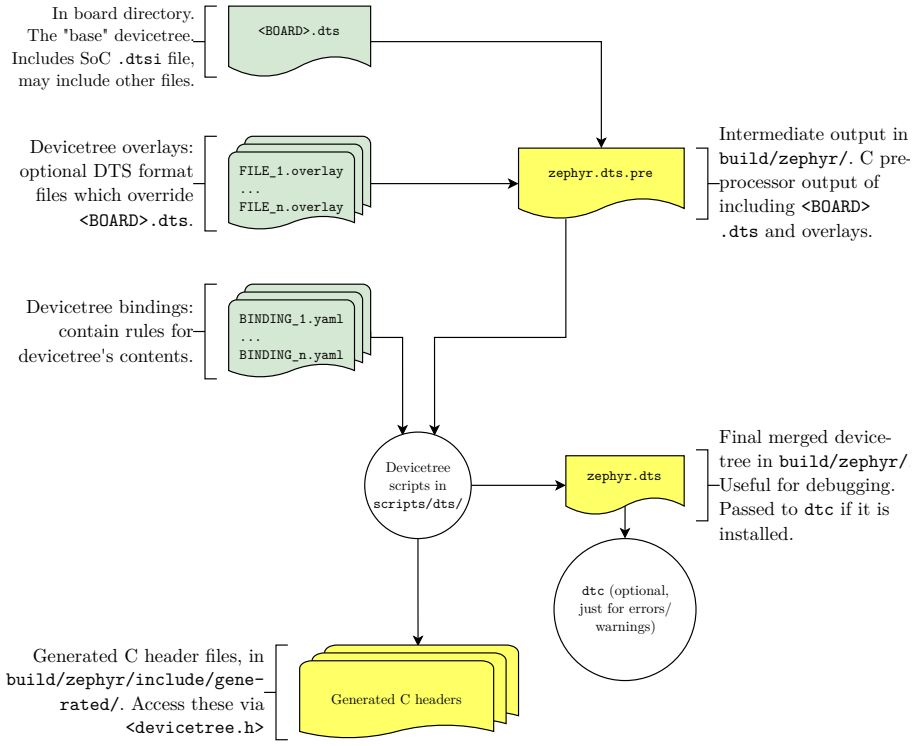


Figure 1.3: Devicetree input files (green) and output files (yellow) [13]

process in which the build system searches out device-trees in certain locations and merges them into the `zephyr.dts` that will be used for configuring and mapping every peripheral. Typically, each supported board has a file called `BOARD.dts` that defines the hardware of the board. The `BOARD.dts` file includes one or more `.dtsi` files that describe the CPU or system-on-chip that Zephyr runs on, and other common hardware features shared by multiple boards. These `.dtsi` files may also include other `.dtsi` files. Additionally, the

`BOARD.dts` file provides a description of the specific hardware of the board. After parsing the `BOARD.dts` file, the main point being the merge with the `.overlay` file specific to both the project and the board. This overlay enables the portability feature of Zephyr. A significant degree of the workload when implementing Zephyr projects are thus in writing hardware devicetrees and then writing as generic firmware implementations as possible. Next, the configuration phase can be seen in fig. 1.4. Once configuration phase is done, CMake generates build scripts that are native to the host platform and initiate the build sequence with the build system Ninja [12]. Afterwards, the generated build scripts are executed to begin the second phase, build. The build scripts can recompile the application without involving CMake after most code changes. Zephyr uses the "target" concept of CMake to organize the build, where a target can be an executable, a library, or a generated file. The final output of Ninja is a binary file ready for flashing using a microcontroller programmer.

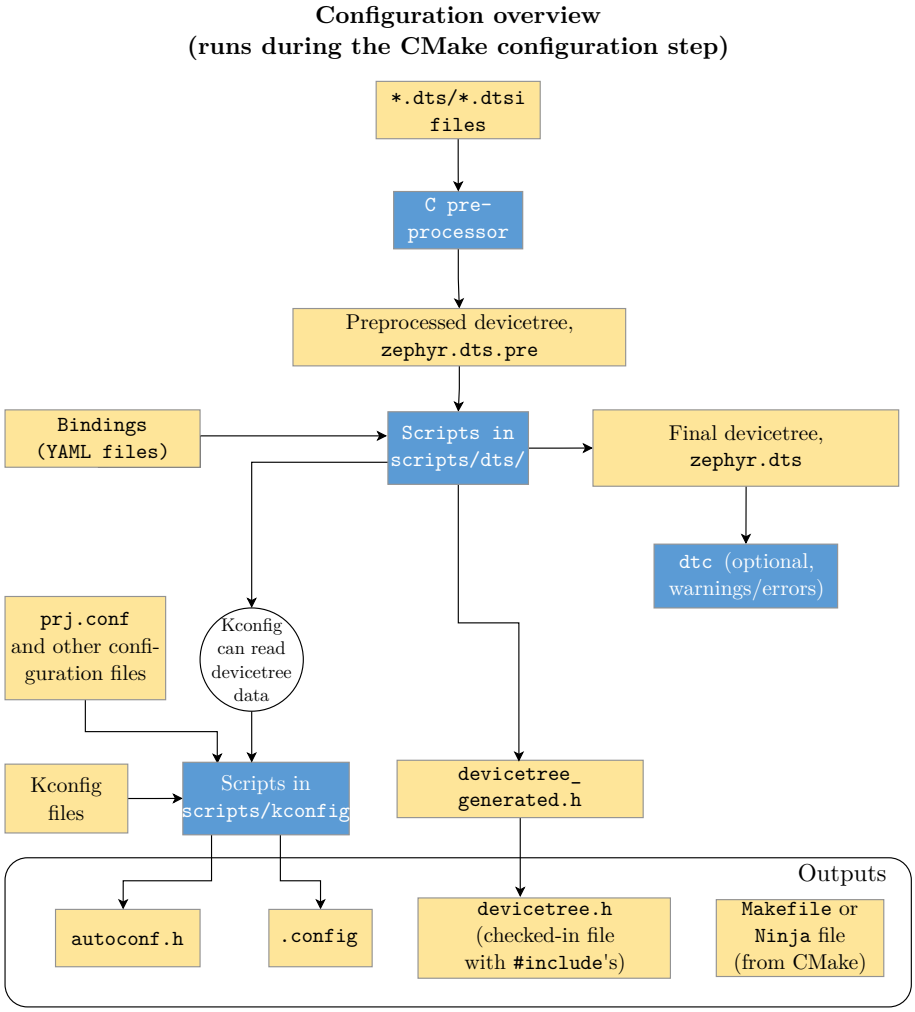


Figure 1.4: Configuration phase of a Zephyr application [13]

After cmake configuration phase has completed, the build phase begins. CMake invokes the build system, which (conceptually) has five (six, one is repeated) stages: (I) pre-build, (II) generation and compilation, (III) first-pass binary (and (IV) second-pass binary), (V) final binary and (VI) post-processing. In fig. 1.5 the build system binds system call functions to implementations. Next, in fig. 1.6 the build system collects source files for various subsystems (decided by the configuration phase) and compiles into archives. The `gen_app_-`

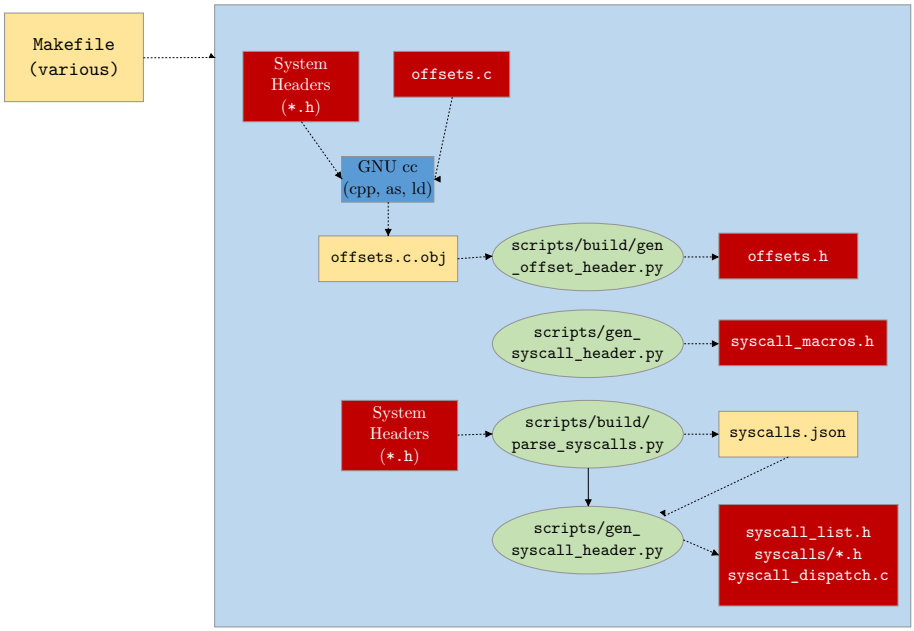


Figure 1.5: Flowchart of build stage I, pre-build

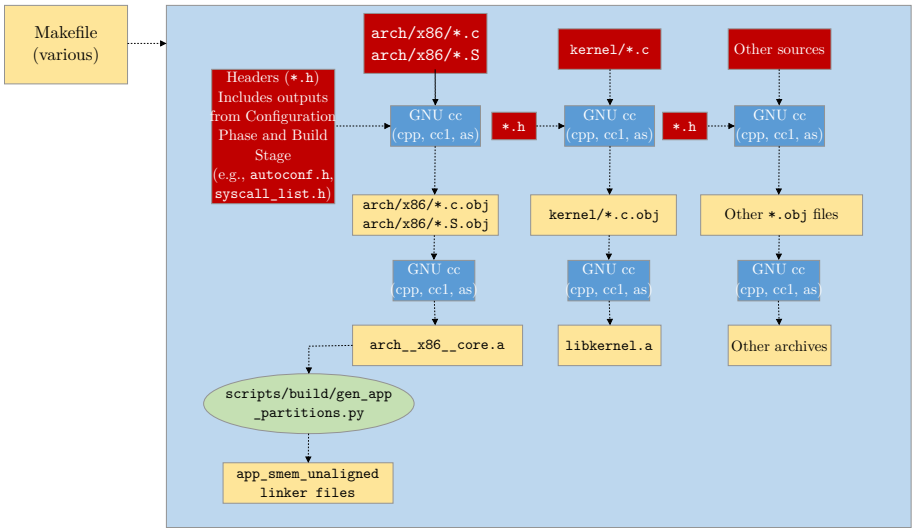


Figure 1.6: Flowchart of build stage II, generation and compilation

`partitions.py` script examines all the archives that are produced and creates linker scripts that organize and align application partitions correctly, based on the memory protection hardware of the target. Then `cpp` process involves merging linker script fragments from various sources, including the target's architecture/system-on-a-chip (SoC), the kernel tree, partition output (if memory protection is enabled), and any other selected fragments from the configuration process. These are combined into a `linker.cmd` file. The compiled archives are then linked with `ld`, following the specifications in the `linker.cmd` file. Shown

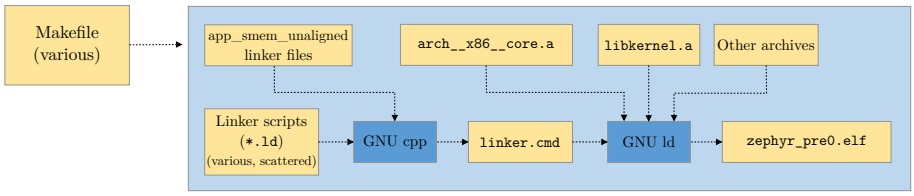


Figure 1.7: Flowchart of build stage III, intermediate binary

in fig. 1.7 is the process, in which an unfixed size intermediate binary is produced. If a devicetree is being used, an intermediate binary is generated that has a variable size. The binary is not fixed in size, which means that it can be modified by post-processing steps that affect the size of the final binary. In Figure 1.8 the fixed size intermediate binary is

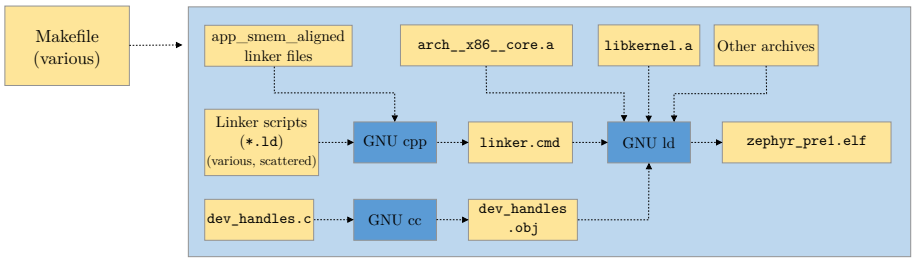


Figure 1.8: Flowchart of build stage IV, second intermediate binary

generated. The previous stage's binaries are not fully formed and contain sections that are empty or marked as placeholders. These sections must be filled in by a process similar to reflection. To finish building, certain scripts are run on the intermediate binaries. These scripts generate the necessary components that are missing from the final binary. Next, in

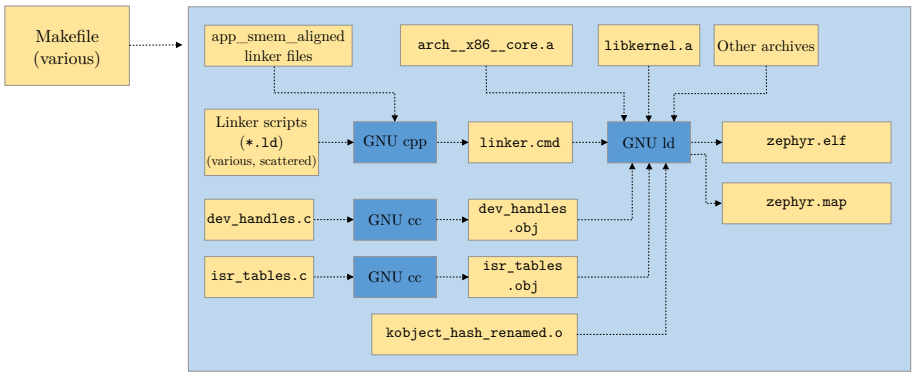


Figure 1.9: Flowchart of build stage V, final binary

fig. 1.9, the final binary is produced by repeating the link from the previous stage, but this time with all the missing pieces populated. Finally, in fig. 1.10, using `GNU objdump`, the

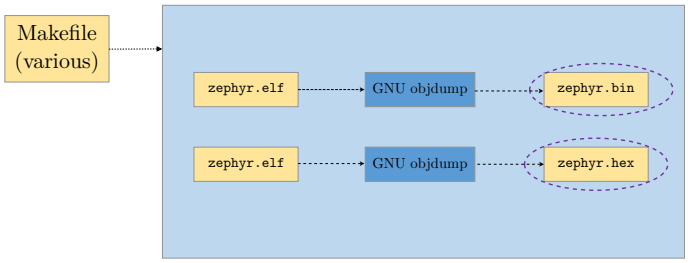


Figure 1.10: Flowchart of build stage VI, post-processing

completed kernel is converted from a executable and linkable format (*ELF*) file to the hex file that is expected by the flash tool compatible with the target device.

1.2 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 *MCU* 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 microcontroller with the functionality though with a trade-off in resolution. However, for this application there is no need to amplitude modulate the output signal and therefore no downside. In the context of a PW Doppler system, it is desired to generate 4 signals:

- 5 MHz complementary signal with dead-time for the pulsed burst during transmit mode.
- 10 kHz signal as f_{prf} for the timing control of the transmit/receive switch.
- 20 MHz clock signal for the demodulation circuit in the receiver.
- 10 kHz gate pulse for S/H control with pulse length $t_{\text{pulse}} = N_{\text{pulse}} \times T_{\text{prd}}$, where $T_{\text{prd}} = {}^{1}/_{f_{\text{pwm}}}$

1.3 Power Stage

Several MOSFET drivers were considered, e.g. ISL55111[11], EL7104[2], and MD1213[8]. 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 in medical ultrasound and other applications needing a high output current for a capacitive

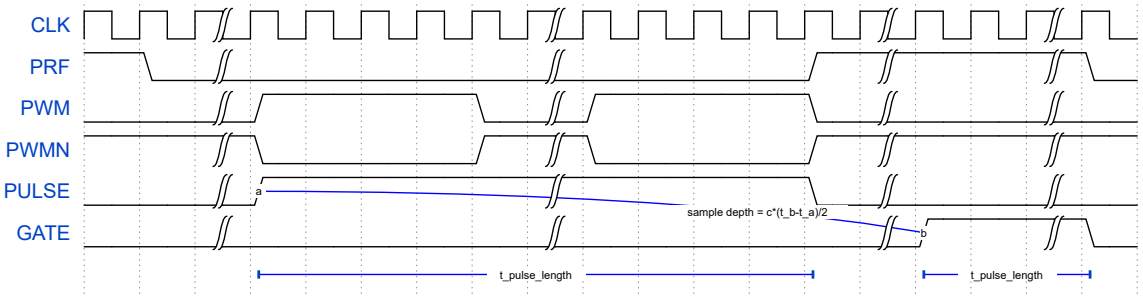


Figure 1.11: Timing diagram of various control signals for an arbitrary n length pulse chain expressed by the second diagram gap

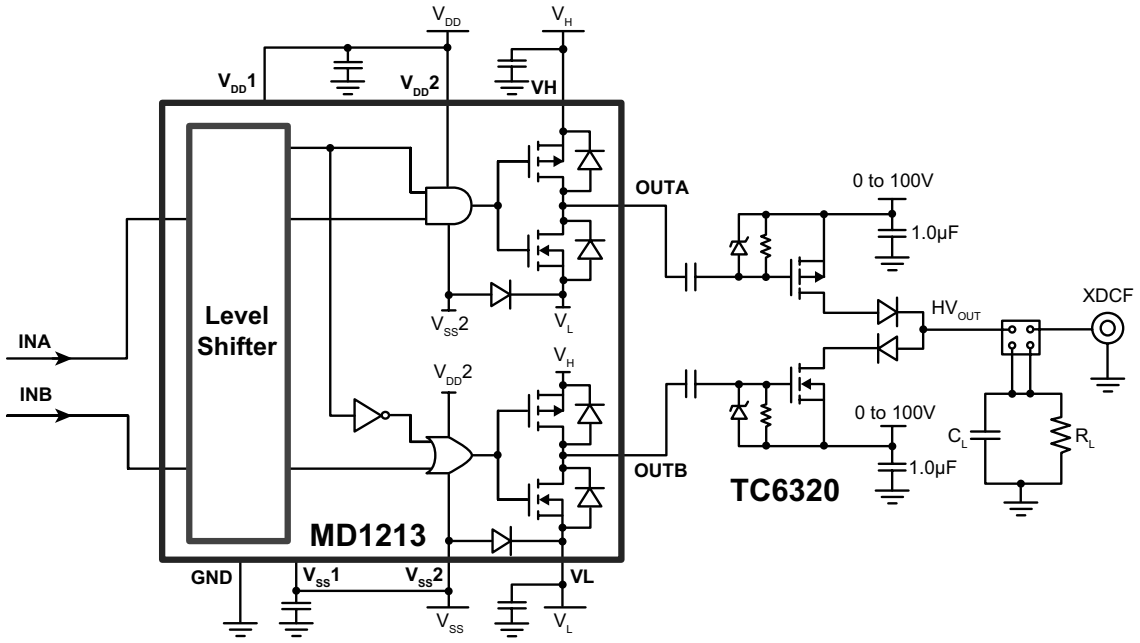


Figure 1.12: Block diagram of power stage [5]

load. It has a high-speed input stage with a logic interface that can function from $1.8\,\mathrm{V}$ to $5\,\mathrm{V}$ and an ideal operating input signal range of $1.8\,\mathrm{V}$ to $3.3\,\mathrm{V}$. 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 features 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 $0\,\mathrm{V}$ and $1.8\,\mathrm{V}$, whereas the control logic is powered by $+5\,\mathrm{V}$ to $-5\,\mathrm{V}$. The output LOW and HIGH values, on the other hand, may be changed between $-5\,\mathrm{V}$ to $+5\,\mathrm{V}$. 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 $2\,\mathrm{A}$, depending on the load capacitance and supply voltages employed. Seen in section 1.3 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

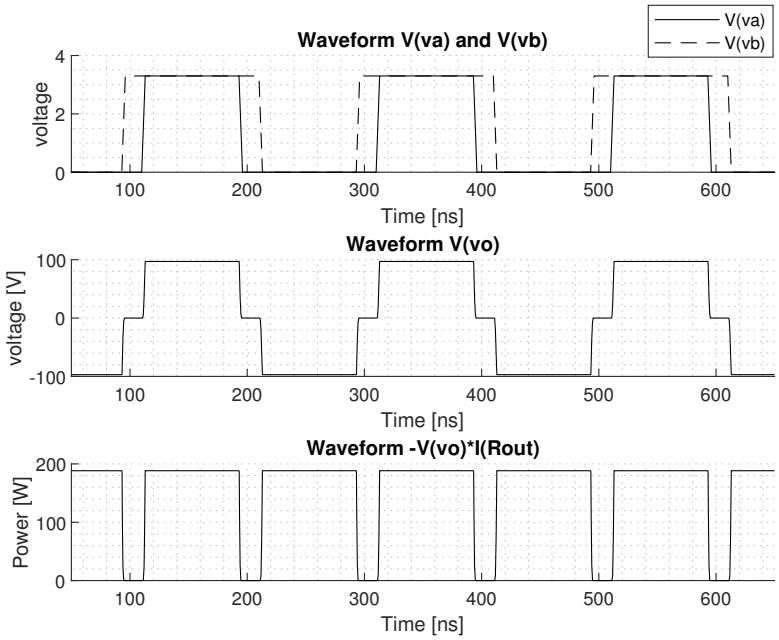


Figure 1.13: LTspice simulation output of transmitter with level shifter and half-bridge power stage from fig. B.1

simulation of the power stage was implemented where the full model can be seen in fig. B.1. The resulting waveforms are seen in fig. 1.13. In the top subplot, the input voltages `INA` and `INB` are seen with their dead-time visible on each overlapped on period. Since `INB` is driving an N-channel metal-oxide-semiconductor field-effect transistor (*MOSFET*), the driving pulse-train should be thought of having the opposite polarity. When looking at the middle subplot, it is noted that dead-time is visible as the time where 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 equipment available for testing, the pulse peak voltage will be less than $100\,\mathrm{V}$.

1.4 Transmit/Receive Switch

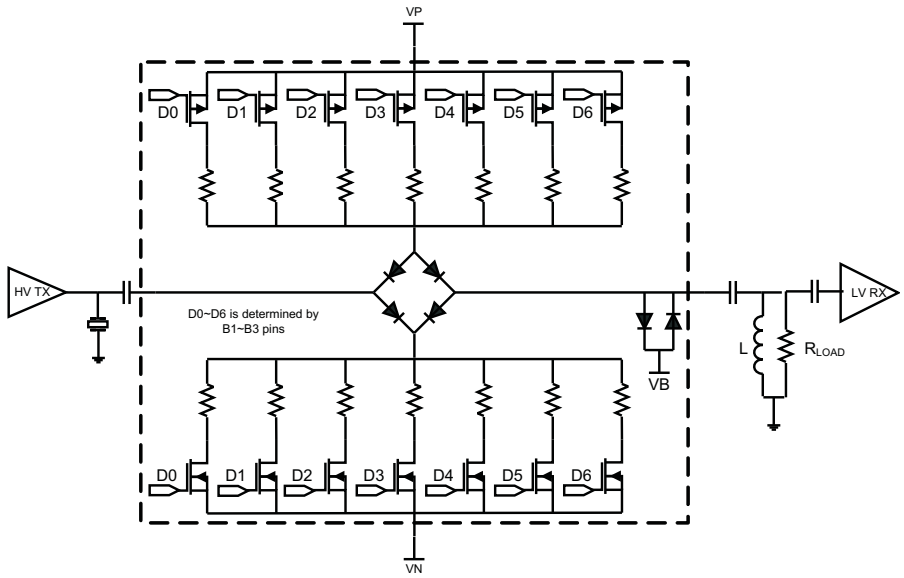


Figure 1.14: Block diagram of TX/RX switching circuit where the inputs D1 through D6 are binary decoded from inputs B_1, B_2, B_3 [3]

Among the design considerations for the transmit and receive switch were the TX810[3] and MD0101[10]. 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 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. 1.15 for a visualisation of the switching operation, where `INPUT` is the incoming Doppler waveform being picked up from the transducer, `B3/B2/B1` is the switching signal closing the switch and thereby going in receive mode, and `OUTPUT` is the received signal seen in the analogue front end (*AFE*). The TX810 can switch the transmit and receive paths

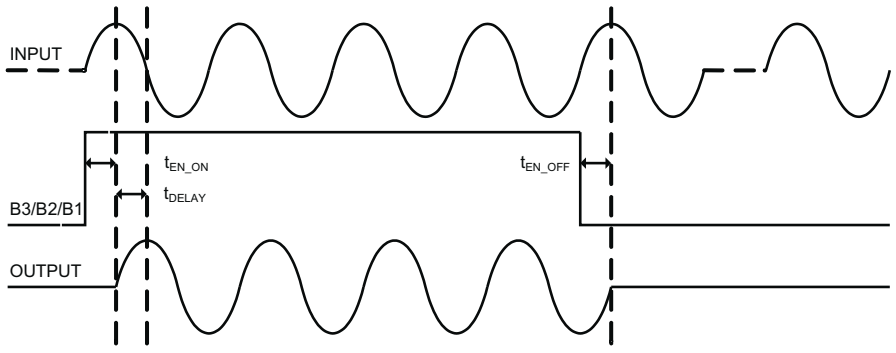


Figure 1.15: Timing diagram of switching interface where $t_{\mathrm{EN\_ON}} = 0.6\,\mu\mathrm{s}$, $t_{\mathrm{EN\_OFF}} = 2.4\,\mu\mathrm{s}$, and $t_{\mathrm{DELAY}} = 1.3\,\mathrm{ns}$ for the condition $B_1 = B_2 = B_3$ [3]

for up to 8 different transducers at the same time. The TX810 is programmed to switch

the transmit and receive paths at specific times, as determined by the user. For example, the user can program the TX810 to switch the transmit and receive paths of a particular transducer at a specific time during the ultrasound examination. The IC is typically used in conjunction with an ultrasound system and one or more transducers. Transducers are used to transmit and receive ultrasound waves, which are used to generate images of the body's internal structures. The TX810 is used to switch the transmit and receive paths for each transducer at the appropriate times, allowing the ultrasound system to capture images from multiple angles simultaneously. 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 $7\,\mathrm{mA}$ to $0\,\mathrm{mA}$ 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 $0\,\mathrm{mA}$ (high-impedance mode). The TX810 does not put significant load on high-voltage transmitters when operating in the high-impedance mode. A PCB design was implemented in Altium Designer [14] utilising three channels of the maximum eight available channels in the IC. Seen in fig. 1.16 is a 3D render of the designed PCB. The module is designed with three usable channels, either three separate transducers

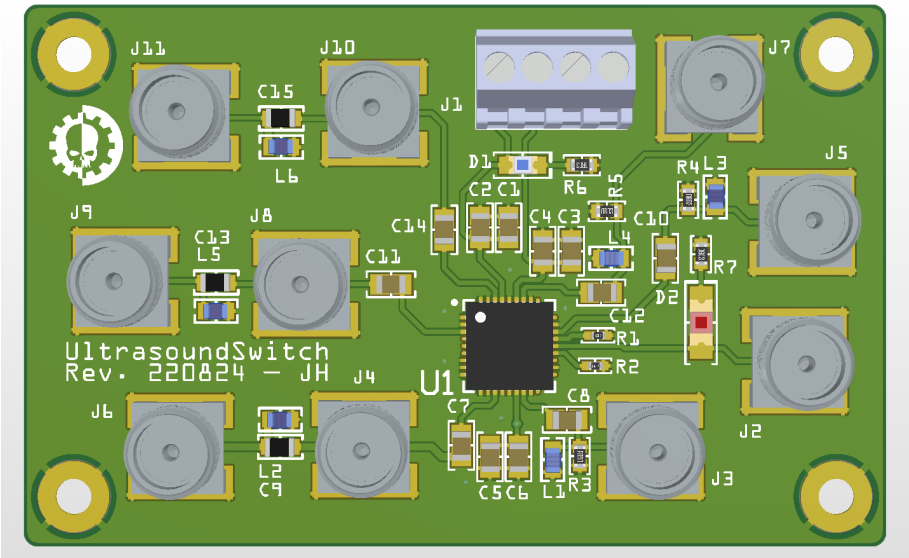
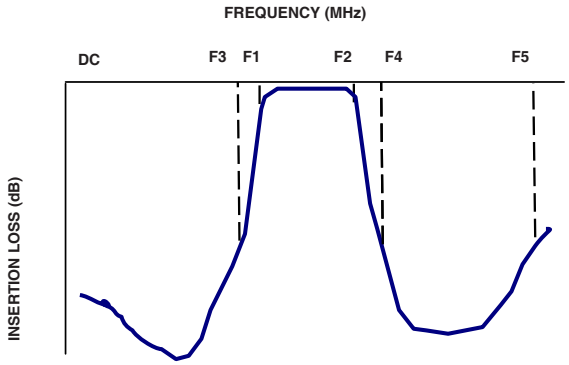


Figure 1.16: 3D Render of PCB in Altium Designer

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

1.5 Band-Pass Filter

After the signal is received, it is filtered with a band-pass (*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 [9] are seen in fig. 1.17. 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.



Electrical Specifications at 25°C

Parameter		F#	Frequency (MHz)	Min.	Typ.	Max.	Unit
Pass Band	Center Frequency	—	—	-	4.5	-	MHz
	Insertion Loss	F1-F2	2-7	-	0.5	1.5	dB
	VSWR	F1-F2	2-7	-	1.1	1.5	:1
Stop Band, Lower	Insertion Loss	DC-F3	DC-0.6	20	35.9	-	dB
	VSWR	DC-F3	DC-0.6	-	20	-	:1
Stop Band, Upper	Insertion Loss	F4-F5	17-2100	20	28.9	-	dB
	VSWR	F4-F5	17-2100	-	20	-	:1

Figure 1.17: 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 stop band attenuation of 20 dB

1.6 Preamplifier

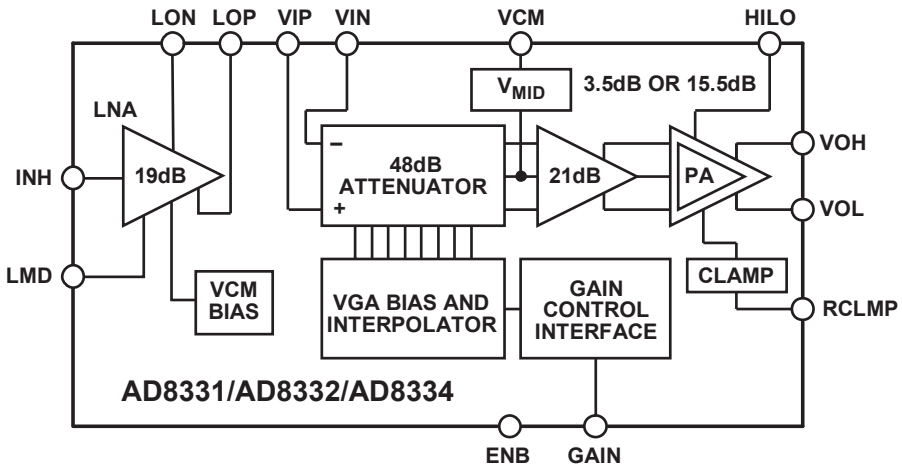


Figure 1.18: Block diagram of preamplifier AD8332 [6]

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 [6], which is a device that combines a dual-channel *LNA* and *VGA*, designed specifically for ultrasound systems. A diagram of its internal functional blocks can be seen in fig. 1.18. 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 postamp 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 postamp can be pin-selected to $3.5\,\mathrm{dB}$ or $15.5\,\mathrm{dB}$, depending on the converter requirements. The output can be limited to a user-defined clamping level to avoid input overload to a subsequent analogue-to-digital converter (ADC), with the clamping level adjusted using an external resistor. A SPICE macromodel is provided by the vendor and the preamplification is succesfully simulated using LTspice with the full LTspice model found in fig. B.2, and the probed inputs and outputs seen in fig. 1.19.

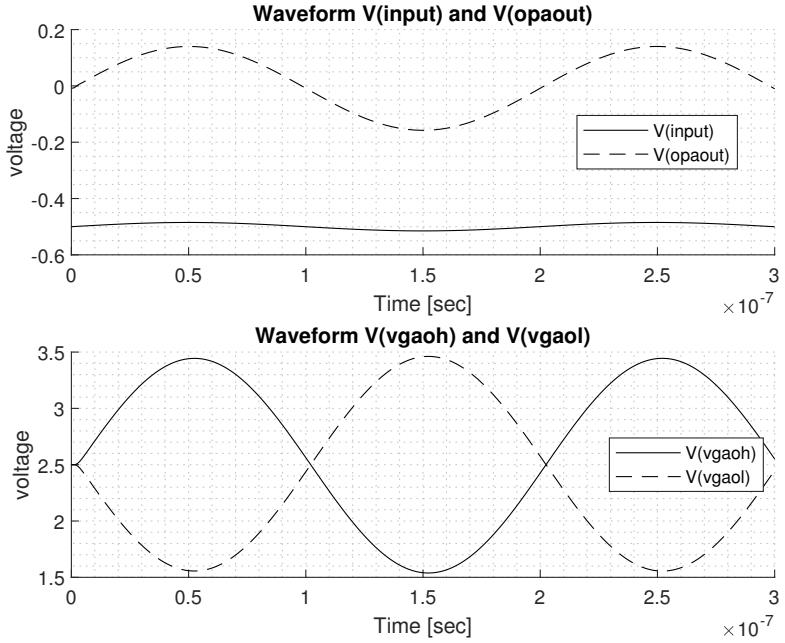


Figure 1.19: LTspice simulation output of preamplifier LNA and VGA from fig. B.2

1.7 Quadrature Demodulator

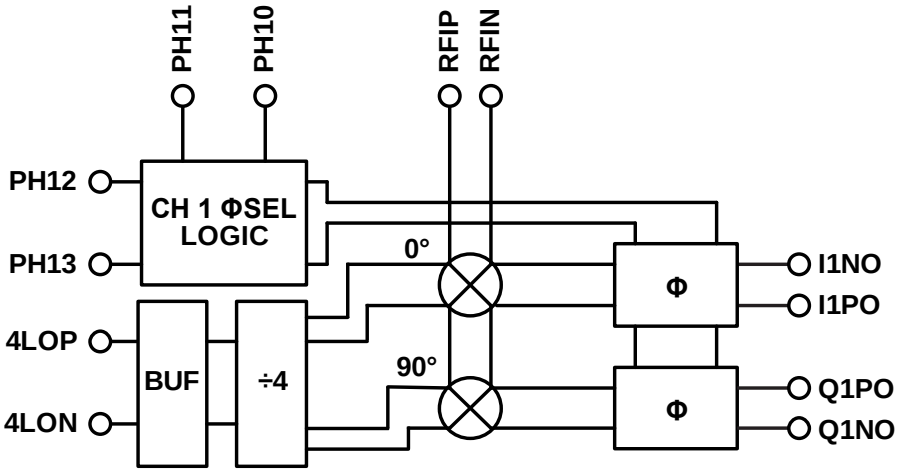


Figure 1.20: Block diagram of demodulator AD8333 [7]

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 [7] I/Q demodulator. A diagram of the internal functional blocks can be seen in fig. 1.20 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 [1], which functions as a transimpedance amplifier. A SPICE macromodel is provided by the vendor and the I/Q demodulation is succesfully simulated using LTspice using the LTspice model found in appendix fig. B.3, with the probed inputs and outputs seen in figs. 1.21 and 1.22.

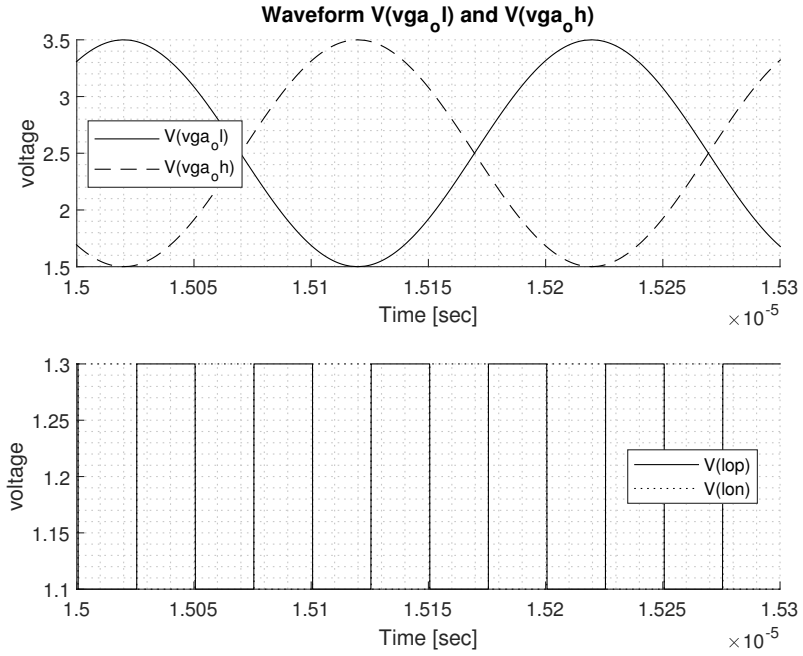


Figure 1.21: LTspice simulation demodulator input variables from fig. B.3

1.8 Sample and Hold

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 analog signal and retains it during certain operations, usually analog-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

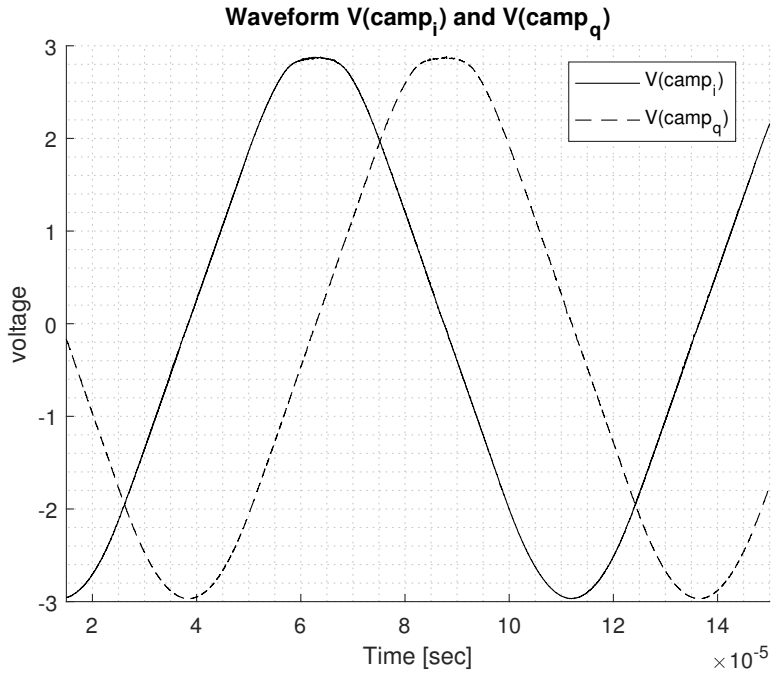


Figure 1.22: LTspice simulation demodulator output variables Q and I voltages from fig. B.3

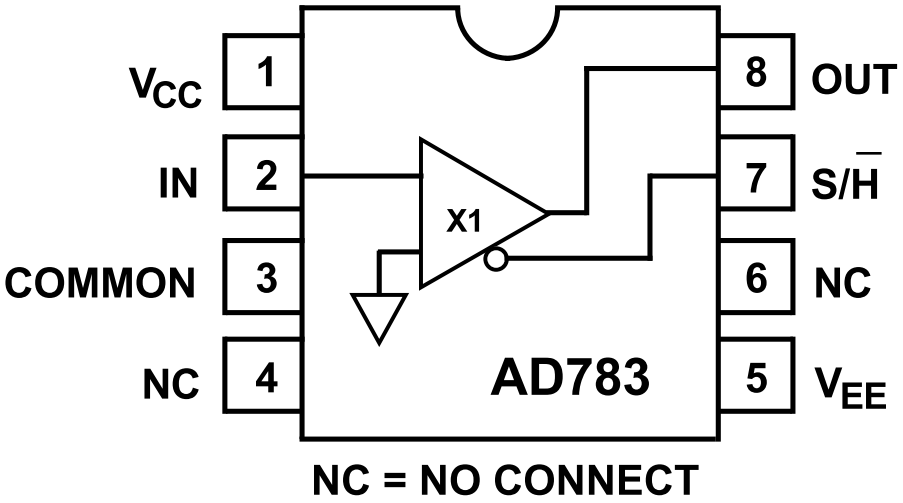


Figure 1.23: AD783 Sample and Hold Amplifier functional block diagram [4]

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. 1.24.

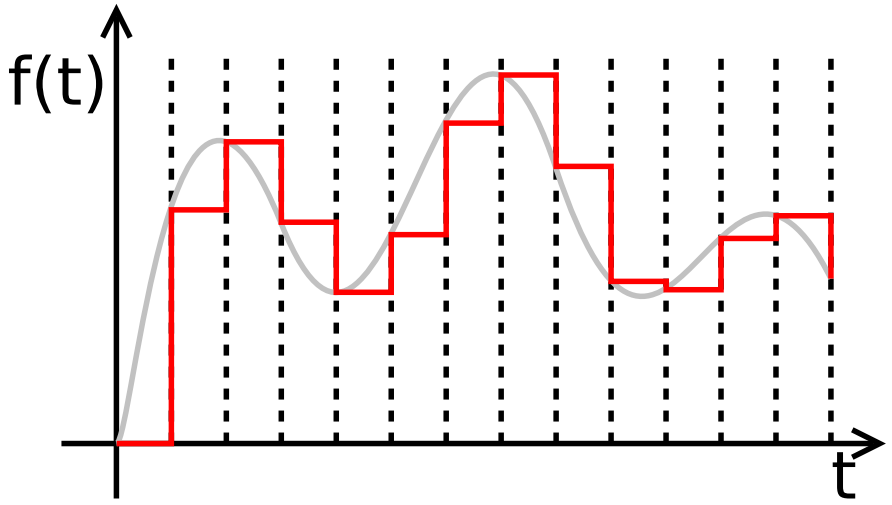


Figure 1.24: Sample and Hold function with input function $f(t)$ over time [16]

1.9 Pulse-Repetition, Wall Filter, DC-Coupling

Finally, the signal should be DC coupled for sampling. In the circuit diagram in fig. 1.25, a *HP* filter is used as a combined pulse repetition frequency (*PRF*) and Wall-filter in conjunction with a $2\,\mathrm{dB}$ gain amplification to maximize dynamic range of the *ADC* measurements from $0\,\mathrm{V}$ to $3.3\,\mathrm{V}$. In the diagram, $V_{DC} = 1.65\,\mathrm{V}$, which is half of the *ADC* dynamic range of $0\,\mathrm{V}$ to $3.3\,\mathrm{V}$, and v_{in} is the input signal to the filter from the output of the sample-and-hold amplifier. 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. 1.26 and the transient analysis can be seen in fig. 1.27.

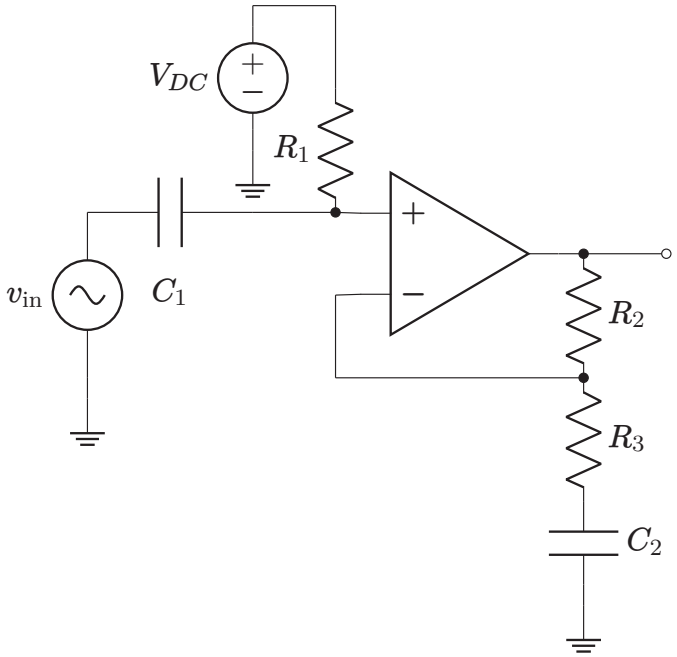


Figure 1.25: Circuit diagram of *HP* PRF and Wall filter

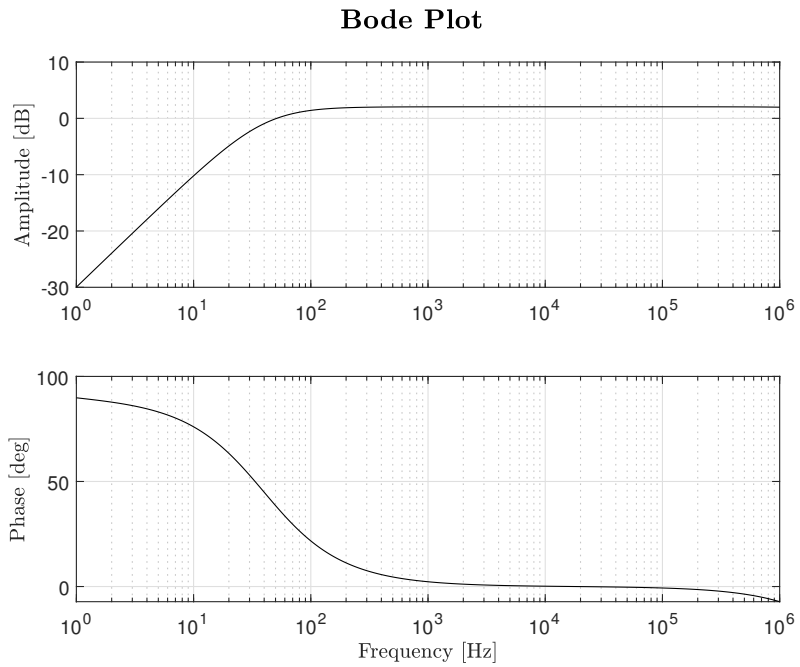


Figure 1.26: Small-signal analysis of DC-Coupling filter circuit

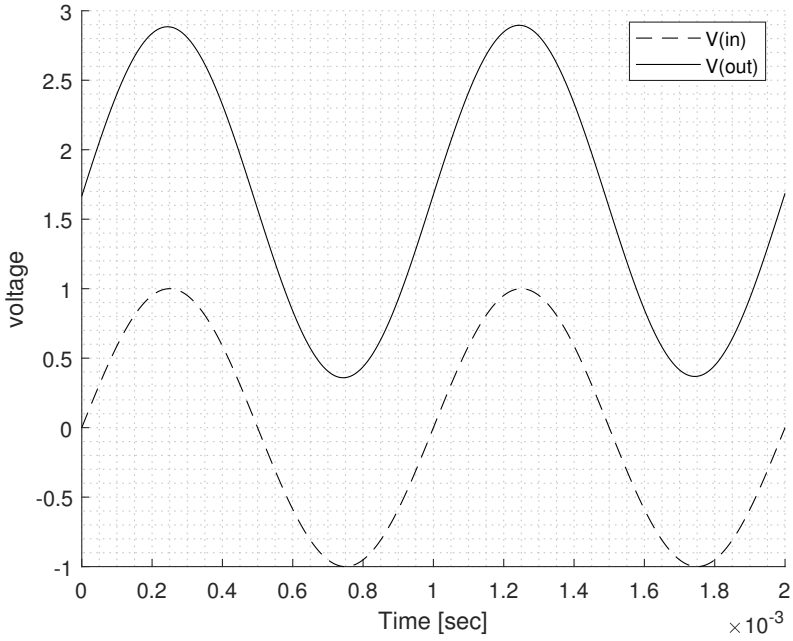


Figure 1.27: Transient analysis of DC-Coupling filter circuit

2 Production

In this chapter, the steps involved in turning a theoretical design into a tangible system will be outlined. Since the synthesis chapter dealt with a explanations 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 setups and reproducing the desired results using lab experiments. As such, each module will be tested independently to validate its function before a larger, more comprehensive experiment is conducted.

2.1 Control System

2.2 Power Stage

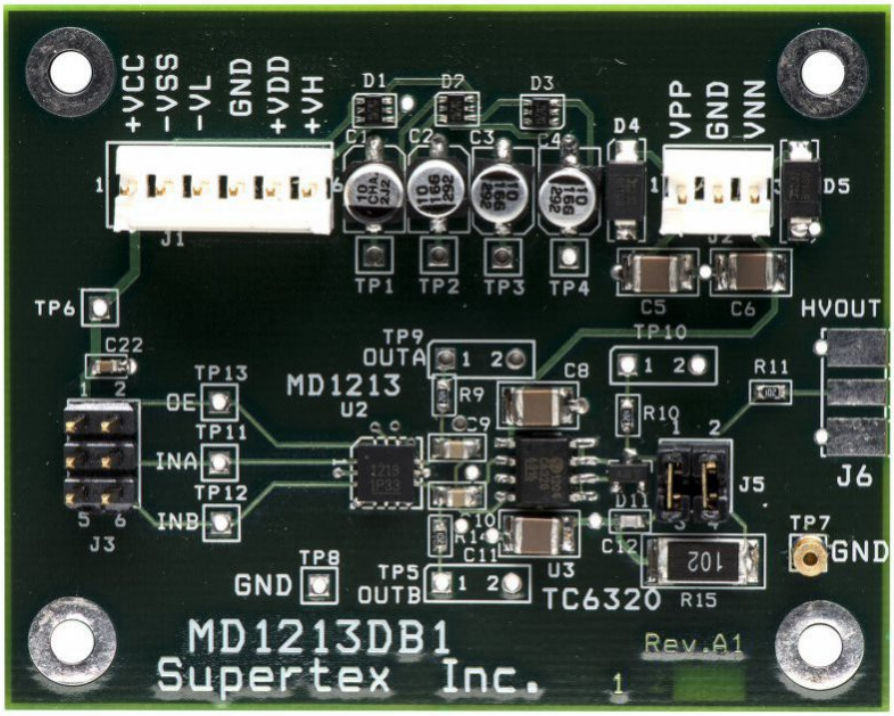


Figure 2.1: MD1213DB1 High Speed Pulser

A picture of the power stage PCB can be seen in fig. 2.1. 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 lead circonate titanate (PZT) transducer is attached with a splitter adapter to connect the other side to an oscilloscope for data acquisition. Seen in fig. 2.2 are actual measured inputs and outputs of the power stage. On the input, there are two complementary $5\,\mathrm{MHz}$ 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 **??**. Noticeable noise is observed in the input signal top and base, but is negligible for the successful operation. Possibly, the noise is due to a cable and adapter problem.

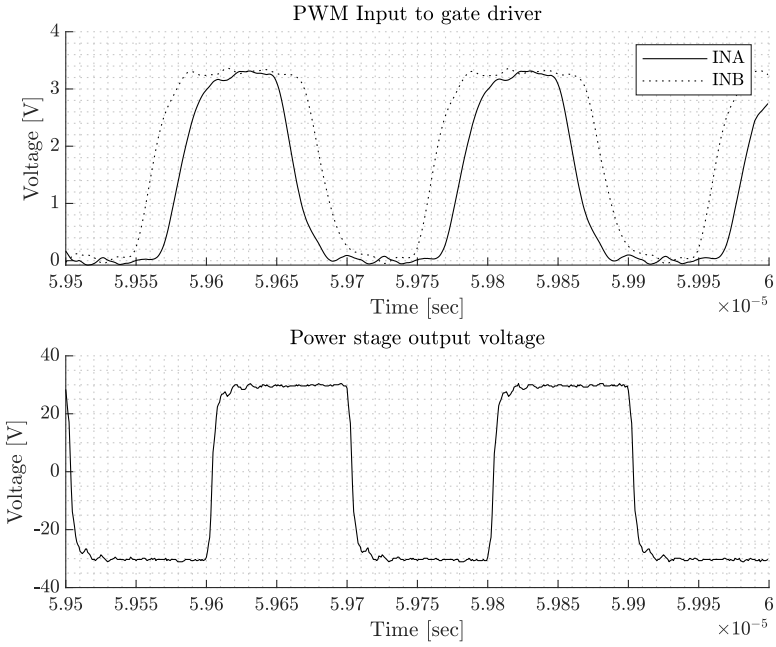


Figure 2.2: Measured input and output of power stage PCB. (Above) Input to gate driver with dead-time (Below) Output of MOSFET half-bridge and the voltage across the load

2.3 Transmit/Receive Switch

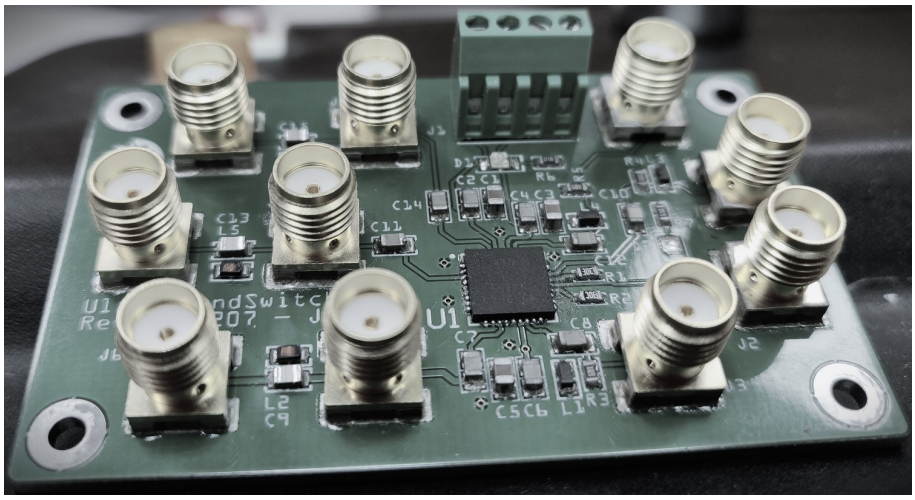


Figure 2.3: Transmit/Receive Switch after assembly

The entire schematic of the transmit/receive switch can be found in the appendix in **????**. 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 aligned for solder paste application. After 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 C.1. The finished assembly can be seen in fig. 2.3. 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 $f_0 = 5\,\text{MHz}$

Figure 2.4: TX/RX Switch reflection experiment with water tank

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. 2.4. 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. 2.5.

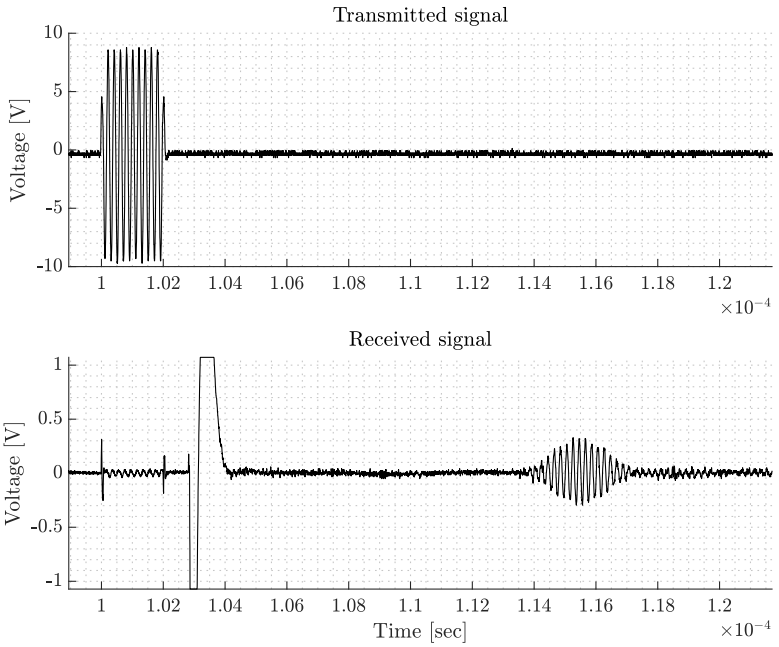


Figure 2.5: Measured transmit and receive on Transmit/Receive Switch PCB (Above) Measured transmit voltage (Below) Received reflected signal off water tank

2.4 Band-Pass Filter

As the band-pass filter was described in the previous chapter, it is desired to validate its frequency response to determine if it lives up to its function. To obtain the frequency response, a bode plot of the magnitude and phase is measured from $300\,\mathrm{kHz}$ until $20\,\mathrm{MHz}$ 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. 2.6 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 $-0.5\,\mathrm{dB}$ frequencies at $1.5\,\mathrm{MHz}$ and $7\,\mathrm{MHz}$. Though, the roll-off in the higher stop band appears somewhat lower than the lower stop-band. That would mean that it is plausible that more higher frequency noise components are retained than in the lower stop band. For the phase, it seems to have a significant phase delay, going from around $100°$ to $250°$ from the start to the end of the pass band.

2.5 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. Seen in fig. 2.7 are measurements of the preamplifier

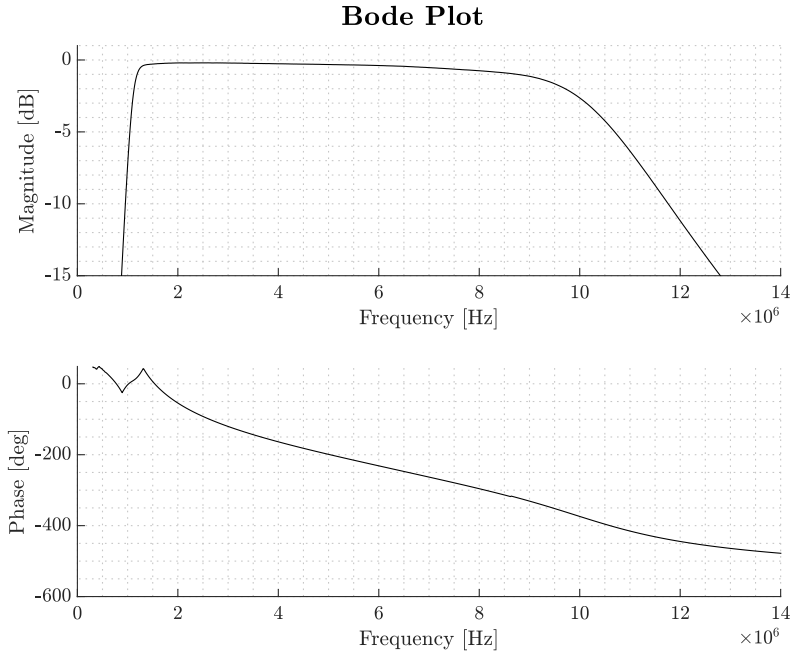


Figure 2.6: Band-pass Filter bode plot from $0.3\,\mathrm{MHz}$ to $14\,\mathrm{MHz}$ with (above) magnitude and (below) phase

circuits showing a $70\,\mathrm{mV}$ input signal and a $300\,\mathrm{mV}$ output signal with a $2.5\,\mathrm{V}$ 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 **??**.

2.6 Quadrature Demodulator

As described in the previous chapter, the demodulator use a I/Q quadrature demodulation scheme to take two differential RF signals and a quadruple frequency signal, in this case $5\,\mathrm{MHz}$ and $20\,\mathrm{MHz}$, respectively, and determines the frequency difference between the fundamental frequency and the Doppler frequency on the output. Seen in fig. 2.9 are the input signals, differential signals of $5.001\,\mathrm{MHz}$ and $20\,\mathrm{MHz}$ local oscillator signal. Seen in fig. 2.10 are 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. The entire schematic of the demodulator can be found in the appendix in **??**.

2.7 Sample and Hold

After each demodulated burst is sampled, between each sample line puse 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. Therefore, 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. 2.11 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.

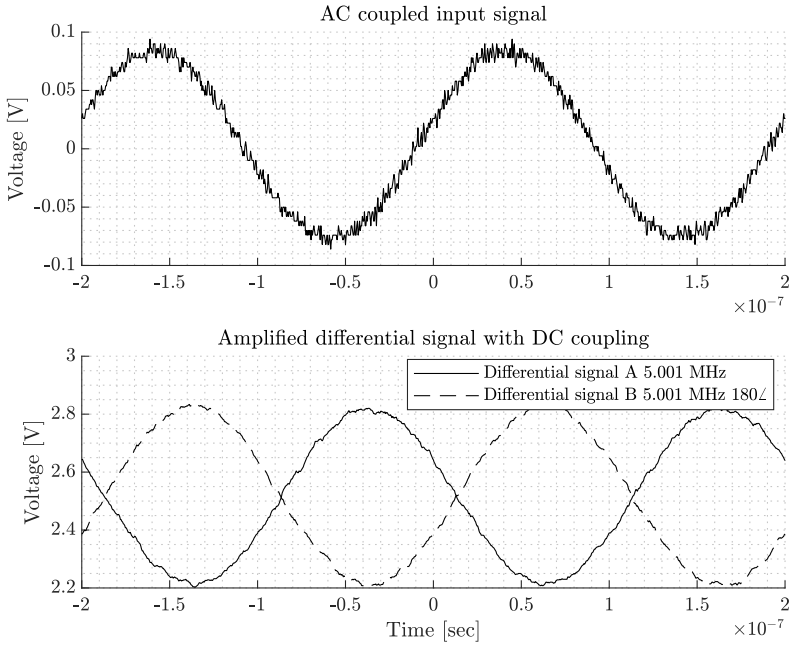


Figure 2.7: Measured input of preamplifier PCB, (Above) AC coupled input signal with amplitude 1 V (Below Measured output of preamplifier PCB, Differential signal with DC coupling and ×19 dB amplification)

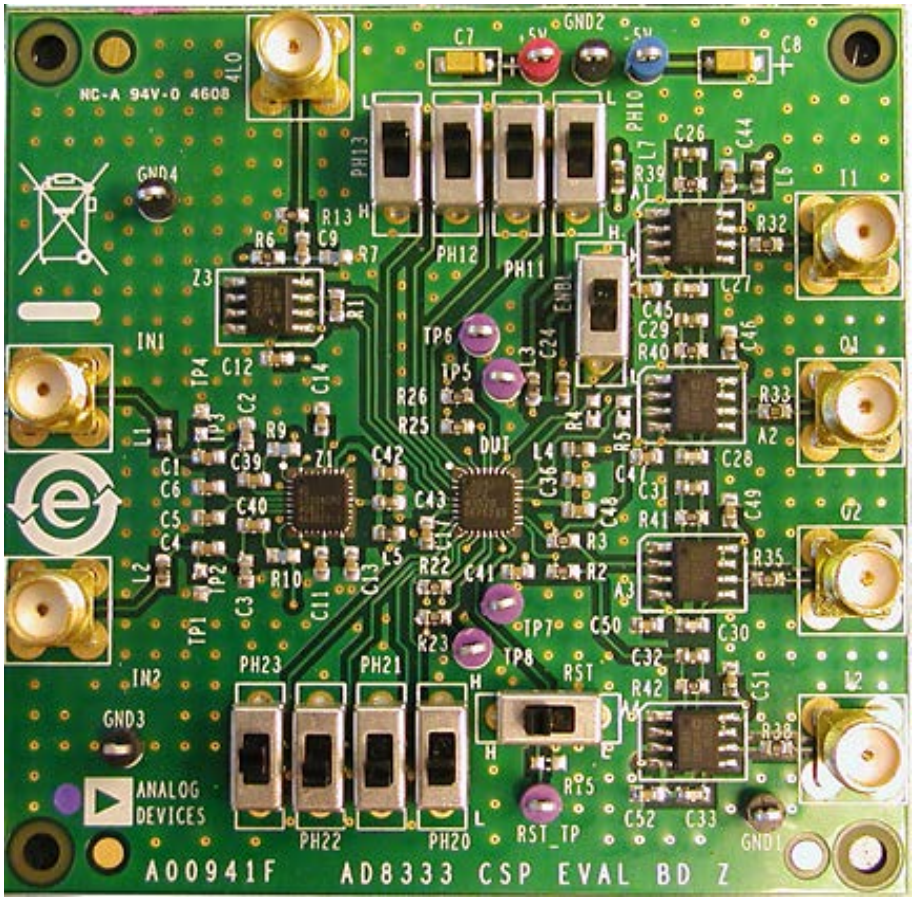


Figure 2.8: Demodulator PCB AD8333-EVALZ

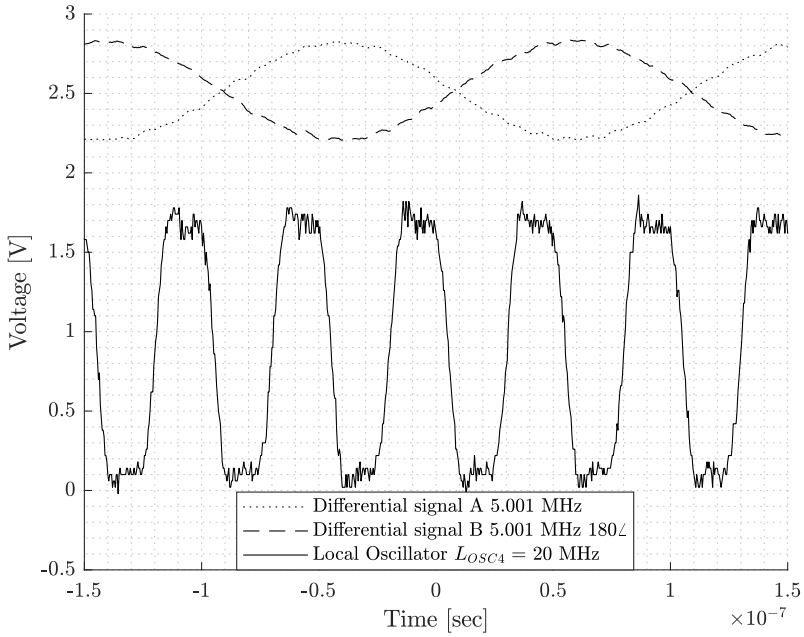


Figure 2.9: Measured input of demodulator PCB (Above) Input from received signal (Below) Input from local oscillator ($f_0 \cdot 4$)

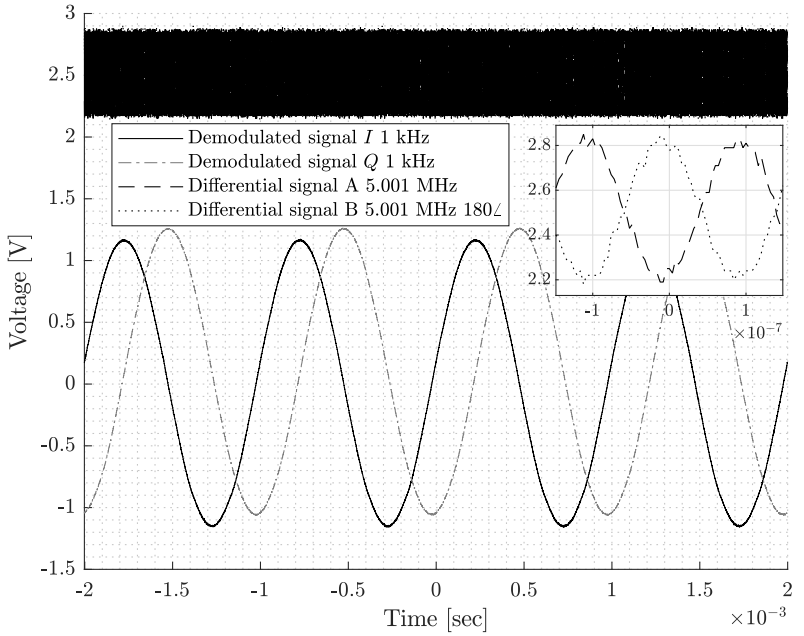


Figure 2.10: Measured output of demodulator PCB

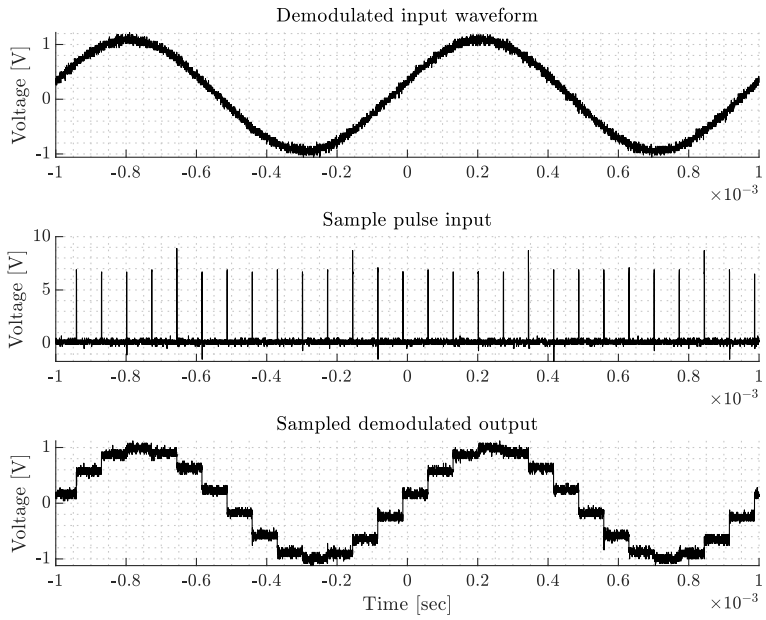


Figure 2.11: Measured input and output of Sample and Hold amplifier

2.8 Pulse-Repetition and Wall Filter

2.9 Digital Signal Processor

Bibliography

[1] *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.

[2] *EL7104 High Speed, Single Channel, Power MOSFET Driver*, Renesas Electronics, Jun. 2006. [Online]. Available: https://www.renesas.com/us/en/document/dst/el7104-datasheet.

[3] *TX810 8-Channel, Programmable T/R Switch for Ultrasound*, Texas Instruments, April 2010. [Online]. Available: https://www.ti.com/lit/gpn/tx810.

[4] *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.

[5] *MD1213DB1 High Speed ±100V 2A Pulser*, Supertex, Inc., March 2014. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/md1213db1.pdf.

[6] *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.

[7] *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.

[8] *MD1213 High-Speed Dual-MOSFET Driver*, Microchip Technology, Inc., Jun. 2017. [Online]. Available: https://ww1.microchip.com/downloads/en/DeviceDoc/20005713B.pdf.

[9] *BPF-C4R5+ Bandpass Filter 50Ω 2 to 7 MHz*, Mini-Circuits, August 2019. [Online]. Available: https://www.minicircuits.com/pdfs/BPF-C4R5+.pdf.

[10] *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.

[11] *ISL55111 Dual, High Speed MOSFET Driver*, Renesas Electronics, April 2021. [Online]. Available: https://www.renesas.com/us/en/document/dst/isl55110-isl55111-datasheet.

[12] "Ninja, a small build system with a focus on speed". (August 2022), [Online]. Available: https://ninja-build.org/ (visited on February 15, 2023).

[13] "Zephyr Project Documentation". (Sep. 2022), [Online]. Available: https://docs.zephyrproject.org/latest/ (visited on February 15, 2023).

[14] "Altium Designer - PCB design software". (February 2023), [Online]. Available: https://www.altium.com/altium-designer.

[15] "CMake Tools - Visual Studio Marketplace". (February 2023), [Online]. Available: https://marketplace.visualstudio.com/items?itemName=ms-vscode.cmake-tools (visited on February 15, 2023).

[16] *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

B Simulation Models

```
.tran 1u    .op  ;ac
.model NMOS-SH nmos (kp=400m vto=1.5 ron=8)
.model PMOS-SH pmos (kp=400m vto=-1.5 ron=8)
.model BAV99 D(Ron=1m Roff=10Meg Vfwd=1 RevIlimit=1 Epsilon=1 RevEpsilon=1)
.model TC63D D(Bv=15 type=zener)
PULSE(0 3.3 {{Tprd}/2+{Tdt}} 3n 3n {{Tprd}/2-2*{Tdt}} {Tprd})
PULSE(3.3 0 {Tdt} 3n 3n {{Tprd}/2-2*{Tdt}} {Tprd})
```

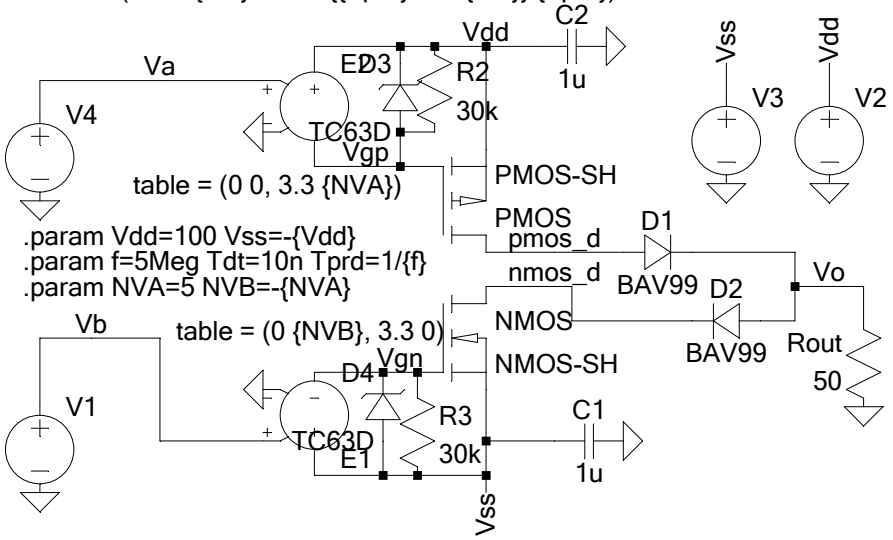


Figure B.1: LTspice model of transmitter

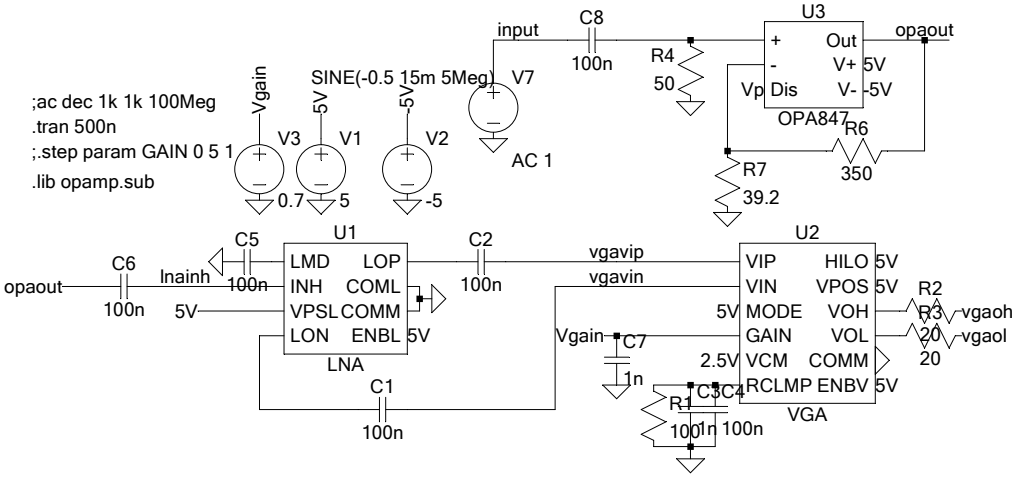


Figure B.2: LTspice model of preamplifier

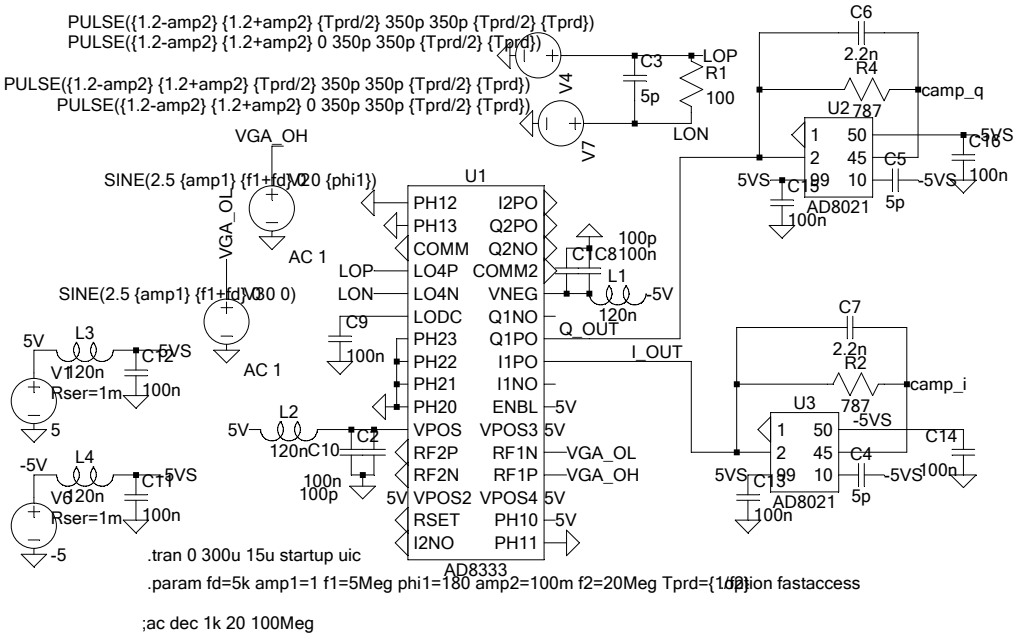


Figure B.3: LTspice model of demodulator

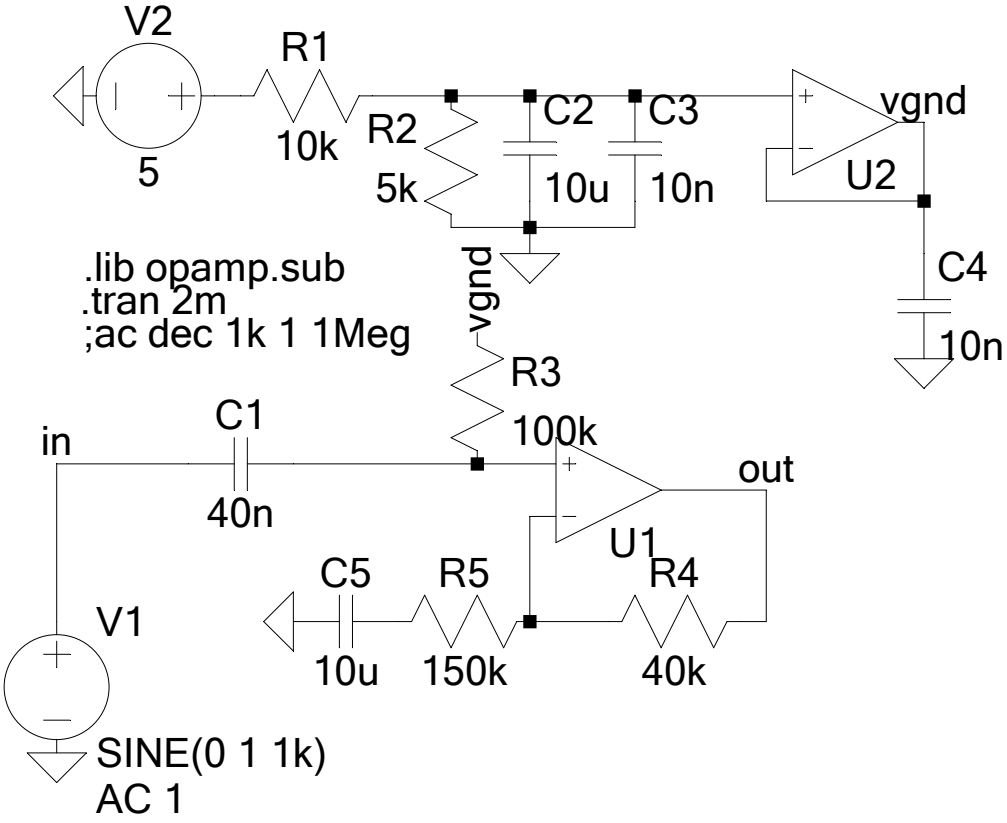


Figure B.4: LTspice model of DC Coupler

C Instruments

Table C.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 C.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