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

Published by (1):	DTU, Department of Electrical Engineering, Ørsteds Plads, Building 348, 2800 Kgs. Lyngby, Denmark `www.elektro.dtu.dk`
Published by (2):	KAIST, School of Electrical Engineering, E3-2, Daehak-ro 291, 34141 Daejeon, Republic of Korea `www.ee.kaist.ac.kr`
Timespan:	Saturday 1st October, 2022 ∼ Wednesday 12th April, 2023
Credits:	30 ECTS / 9 credits

Degree:	Master of Science
Field:	Electrical Engineering

Approval

This thesis has been prepared over six months at the Brain/Biomedical Microsystems Laboratory, School of Electrical Engineering, at the Korean Advanced Institute of Science and Technology, KAIST, and the Department of Electrical Engineering, Technical University of Denmark, DTU. This thesis is in partial fulfillment for the double degree Master of Science in Electrical Engineering (M.S.E.E.).

Jeppe Hinrichs - s163555

.....

Signature

.....

Date

Contents

List of Figures

List of Tables

List of Source Codes

Abbreviations

Notation	Description
ADC	analogue-to-digital converter
AM	amplitude modulation
BLE	bluetooth low energy
BP	band-pass
CMUT	capacitive micromachined ultrasound transducer
CPU	central processing unit
CT	computed tomography
CW	continuous-wave
DSP	digital signal processor
DTS	device tree source
DTSI	device tree source include
DTU	Danmarks Tekniske Universitet (Technical University of Denmark)
ELF	executable and linkable format
I/O	input/output
IC	integrated circuit
IoT	Internet of Things
KAIST	Korea Advanced Institute of Science and Technology
low-res	low resolution
MCU	microcontroller unit
MRI	magnetic resonance imaging
PSD	power spectral density
PW	pulsed-wave
PZT	lead circonate titanate
RAM	random access memory
RTOS	real-time operating system
SHA	sample-and-hold amplifier
SoC	system-on-a-chip
US	ultrasound

Glossary

Notation	Description
adiabatic	Any process that happens without heat gain or loss is considered adiabatic
Doppler effect	A change in frequency of a wave in relation to an observer who is moving relative to the wave source
dynamic range	The ratio between the largest and smallest values that a certain quantity can assume, in signal theory usually measured in decibels
flashing	Refers to the process of programming the non-volatile memory of a microcontroller
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
open-source	Refers to software whose original source code is publicly available and may be changed and distributed
piezoelectric	Material that deforms under electric excitation, or reversibly, electric charge that accumulates in response to mechanical stress
quantized	The process of mapping input values from a continuous set to a discrete smaller set with a finite number of elements
transcutaneous	Applied across the depth of the skin without invasive penetration
transducer	A device that converts electrical energy to kinetic energy, or kinetic energy to electrical energy

Nomenclature

Name	Unit	Description
		Input filter
V_{ref}	V	Reference voltage
C_{hp}	F	High pass filter capacitor
R_{hp}	Ω	High pass filter resistor
f_{hp}	Hz	High pass cut-off frequency
C_{lp}	F	Low pass filter capacitor
R_{lp}	Ω	Low pass filter resistor
f_{lp}	Hz	Low pass cut-off frequency
A_v	1	Amplification factor
		Modulator
R_1	Ω	AIM voltage divider resistor
R_2	Ω	AIM voltage divider resistor
R_{in}	Ω	AIM input resistor
R_{fb}	Ω	AIM feedback resistor
C_1	F	AIM capacitor
V_{in}	V	Input signal voltage
V_{span}	V	Voltage range of input signal
V_{pwm}	V	PWM signal
V_H	V	V_{pwm} high level voltage
V_L	V	V_{pwm} low level voltage
V_{out}	V	V_{pwm} voltage range ($V_H - V_L$)
V_{hys}	V	Hysteresis voltage
V_{hw}	V	Hysteresis width
V_{th_H}	V	Hysteresis threshold upper voltage
V_{th_L}	V	Hysteresis threshold lower voltage
V_c	V	PWM carrier voltage
V_{c_H}	V	PWM carrier upper voltage
V_{c_L}	V	PWM carrier lower voltage
f_{sw}	Hz	PWM signal frequency
D	1	PWM signal duty cycle
t_H	s	PWM carrier charge time

Continued on next page

Name	Unit	Description
t_L	s	PWM carrier discharge time
τ	1	PWM carrier charge constant
R_{th}	Ω	PWM carrier thevenin resistance
f_{idle}	Hz	PWM signal idle switching frequency
k_2	1	R_{fb}, R_{in} voltage divider
		Gate driver
D_{dt}	1	Dead-time circuit diode
R_{dt}	Ω	Dead-time circuit resistor
C_{dt}	F	Dead-time circuit capacitor
V_C	V	Dead-time circuit supply voltage
V_s	V	IC supply voltage
t_c	s	Charging circuit time
		Power stage
V_{DD}	V	Power supply voltage
Q_1, Q_2, Q_3, Q_4	1	Power stage switches
V_g	V	Gate driver signal
V_o	V	Output voltage
I_o	A	Output current
R_{BTL}	Ω	Speaker equivalent load resistance
		Output filter
R_f	Ω	Output filter single-ended load
C_{BTL}	F	Output filter differential capacitance
C_f	F	Output filter single-ended capacitance
L_f	H	Output filter inductance
ΔI_L	A	Output filter ripple current
Q	1	Output filter quality factor
f_c	Hz	Output filter cut-off frequency
ω_n	rads^{-1}	Output filter natural frequency
ζ	1	Output filter damping ratio

Continued on next page

Name	Unit	Description
		Shunt regulator
R_{sh}	Ω	Shunt current limiting resistor
I_K	A	Shunt cathode current
$I_{K_{\max}}$	A	Shunt maximum cathode current
$I_{K_{\min}}$	A	Shunt minimum cathode current
I_{su}	A	Shunt supply current
R_{A1}	Ω	Shunt adjust resistor 1
R_{A2}	Ω	Shunt adjust resistor 2
C_L	F	Shunt load capacitance
$V_{\text{ref}_{\text{IC}}}$	V	Shunt internal reference voltage

Reading comprehension

This section of the report will explain to the reader how to reference this document and explain the fundamental structure of the project as well as the report. Throughout the report, the reader will be assumed to be knowledgeable of basic circuit analysis and familiar with standard abbreviations typically used in electrical engineering. If not, readers can refer to the denotation section at the beginning of the report. It is assumed that the reader has a basic knowledge on the science of physics, electrical engineering, computing, and circuit analysis.

Please refer to Abbreviations, Glossary, and Nomenclature pages for explanations to terms found within the report.

Sources

Calculus expressions present in the report will typically have a reference explaining their origin. All references are prominently displayed with square brackets and a number, directing to the appendix in the last section of the report.

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. Potentially millions of people have been spared painful exploratory surgery through non-invasive diagnostic imaging. And thus, lives can be saved by early diagnosis and intervention through medical imaging. Advancements in scientific visualisation have in turn generated more complex datasets 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.

Since 2004, medical imaging has been reported to have been performed more than 5 billion times [8], and later numbers from 2011 show a general doubling and in particular, a tenfold increase in ultrasound examinations between 2000 and 2011 [22]. Recent data reveal that this trend of doubling has continued throughout the years 2010 to 2020 [34], 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, but inconsistent, resolution, cost-effectiveness, portability, and real-time interventional imaging. The downside of ultrasound is its limited penetration and restrictions for use in certain body parts. 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.

The cardiovascular system, which transports oxygen and nutrients to tissue, produces a complex flow pattern that causes velocity fluctuations. Several cardiovascular diseases are also known to cause abnormal blood flow. As mentioned above, ultrasound is a powerful tool for performing non-invasive imaging of the cardiovascular system [17], [20], 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.

1.1 Project scope

The aim of this project is to study the application of ultrasound in the context of blood flow measurements. Various scientific articles have been studied to gain knowledge of previous research [4]–[6], [9], [11], [13]–[15], [18], [19], [26], [27], [29]–[32]. In addition, textbooks [20], [22], [23] have also been instrumental in forming a solid knowledge base for the thesis. The desire is to build upon the vast knowledge already gathered by prominent researchers in the field of ultrasound systems for blood velocity estimation. Finally, using the knowledge gained, we designed and implemented an electronic device capable of performing these measurements using a novel approach. The system used in this project is called an Ultrasound Doppler flow-meter. Ultrasound Doppler flow-meters can be used to measure the velocity of blood flow in the human body. This is commonly done to assess the health of blood vessels and to diagnose and monitor conditions such as arteriosclerosis (hardening

of the arteries) and deep vein thrombosis (blood clots in the veins). To measure blood velocity with an ultrasound Doppler flow-meter, a handheld probe is placed on the skin over the area of interest, such as an artery or vein. The probe contains a transducer that emits high-frequency ultrasound waves and receives the reflected waves. The Doppler shift in the frequency of the reflected waves is caused by the movement of the blood cells, and it is proportional to the velocity of the blood flow. The probe is connected to a portable ultrasound machine, which processes the Doppler shift and displays the velocity of the blood flow on a screen. The machine can also produce a color-coded map of the blood flow, which allows the user to visualize the velocity of the blood at different points within the vessel. Ultrasound Doppler flow-meters are non-invasive and safe to use, and they provide a quick and easy way to measure blood velocity. However, they are not always accurate, especially in cases where there is a high degree of turbulence or when there are air bubbles or solid particles present in the blood. They are also limited in their ability to measure blood flow in small vessels or in deep tissues. The goals of the project are written in table 1.2.

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 for experimental testing. The fourth chapter explains the production of the hardware. 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.

Table 1.1: Comparison of medical imaging modalities [22]

Modality	**Ultrasound**	**X-ray**	**CT**	**MRI**
Topic	Longitudinal, shear, mechanical properties	Mean X-ray tissue absorption	Local tissue X-ray absorbtion	Biochemistry (*T1* and *T2*)
Access	Small windows adequate	2 sides needed	Circumferential around body	Circumferential around body
Spatial resolution	0.2 mm to 3 mm[a]	$\sim$ 1 mm	$\sim$ 1 mm	$\sim$ 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: 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 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 the systemic circulation is explained in vivo. Lastly, the various types of flow-meters are outlined with their strengths and weaknesses.

2.1 Ultrasound

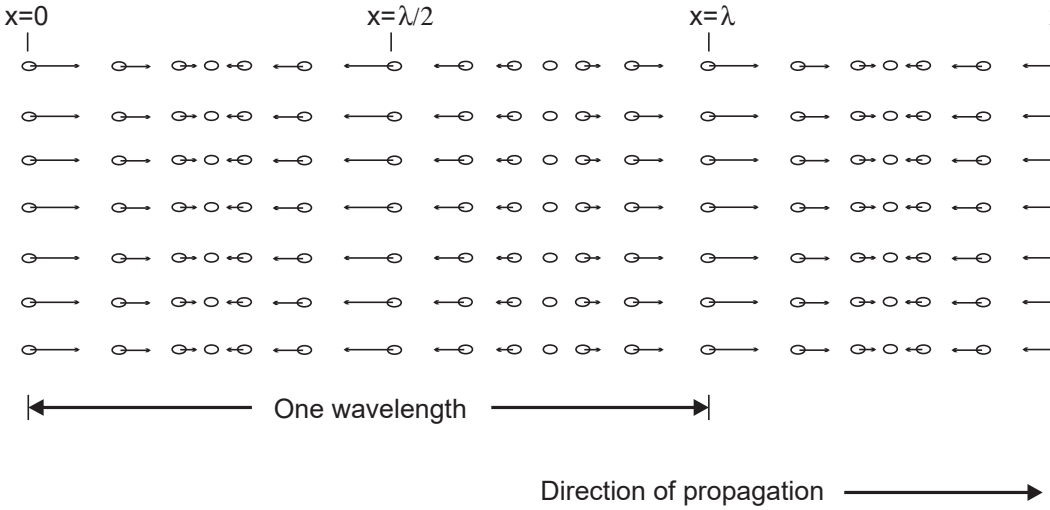

Figure 2.1: Particle displacement for a propagating ultrasound wave [20]

US is a technology that transmit 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 [22]. 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).

$$c = \sqrt{\frac{1}{\rho_0 \kappa_S}} \tag{2.1}$$

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

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

And propagates along the z-axis. ω is the angular frequency, k is the wave number and is expressed by $k = \omega/c = 2\pi/\lambda$, and $_0$ is the acoustic pressure amplitude. A spherical wave is expressed by eq. (2.3)

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

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)

$$u = \frac{p}{Z} \tag{2.4}$$

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).

$$Z = \rho_0 c \tag{2.5}$$

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 [20]

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	$1.38–1.81 \times 10^3$	2070–5350	$3.75–7.38 \times 10^6$
Brain	1.03×10^3	1505–1612	$1.55–1.66 \times 10^6$
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	$1.65–1.74 \times 10^6$
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 [20] 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).

$$P_s = I_i \sigma_{sc} \tag{2.6}$$

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).

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

Where R is distance to the scattering region [20]. This results in a spherical wave. A transducer with radius r gives the power P_r, presuming the attenuation and focus is neglected, and is expressed by eq. (2.8).

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

The backscattering coefficient, which characterizes scattering from a volume of scatterers, 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/cmsr. Backscattering 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 [3], [20].

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.

Table 2.2: Approximate attenuation values for human tissue [20]

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

The pressure of a wave propagating in z-direction decreases exponentially expressed by eq. (2.9)

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

Where $p(z = 0)$ 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).

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

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

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 [23].

2.1.3 Transducer

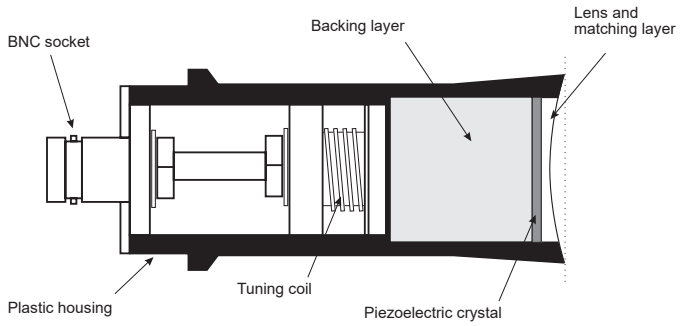

Figure 2.2: Single element ultrasound transducer construction [20]

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. The transducer 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 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

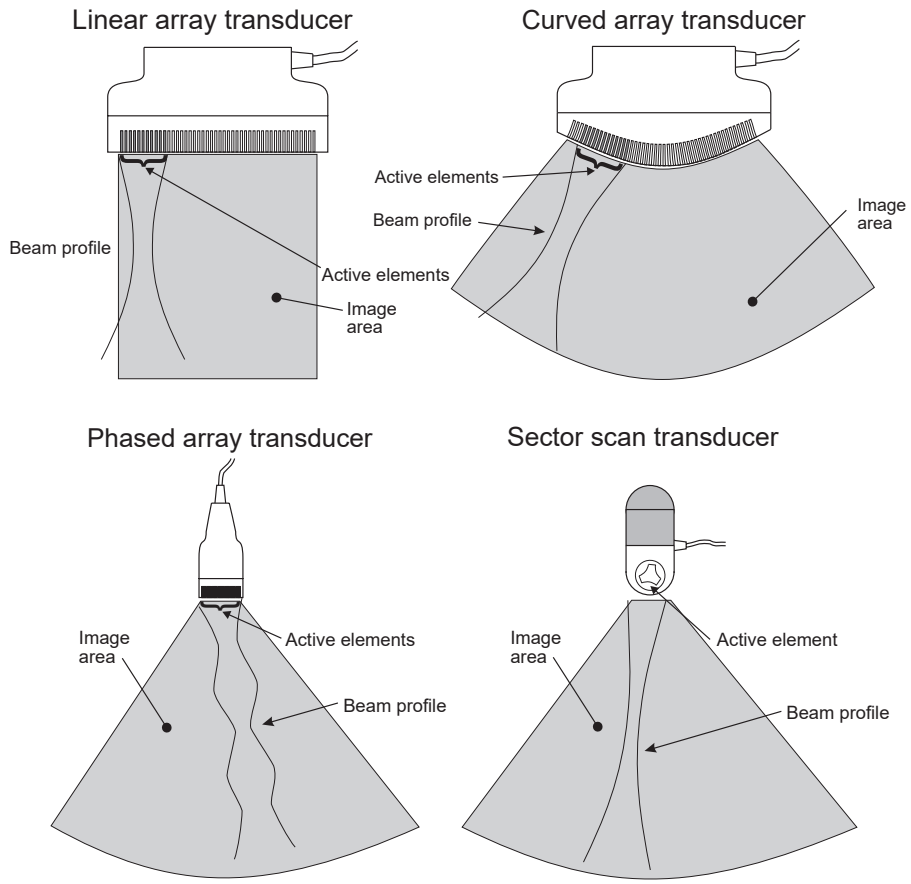

Figure 2.3: Transducer types for acquiring B-mode images [20]

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).

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

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

$$f' = \frac{c}{c+v} \cdot f_0 \tag{2.12}$$

Where Hvad mangl

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).

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

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).

$$f_d = \frac{2vf}{c} \tag{2.14}$$

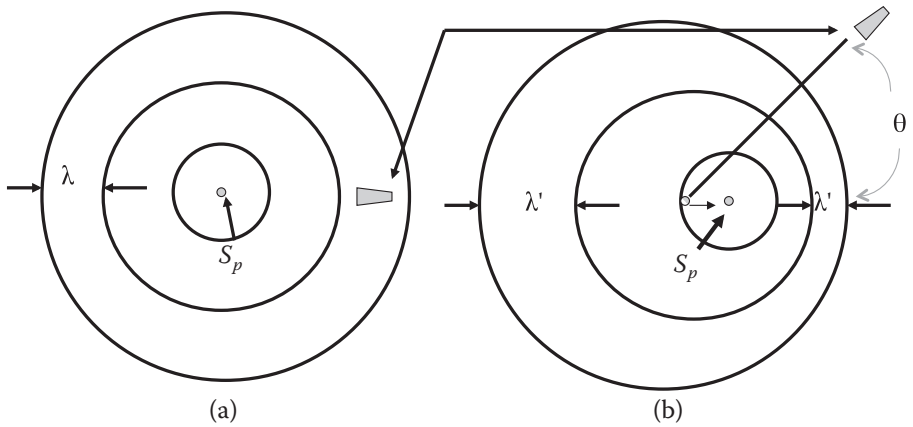

Figure 2.4: Doppler effect diagram. A stationary observer perceives a change in the frequency of a wave generated by a moving source toward the observer as a result of a wavelength shift from λ to λ'. In (a), the source is still. In (b), the source is moving at a velocity v. [23]

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.

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

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.

2.2 Flow Physics

The flow physics of the human circulatory system are sophisticated, and numerous nonstationary flow patterns emerge. 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 subcomponents, such as plasma and formed cellular elements that 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 venulae, 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

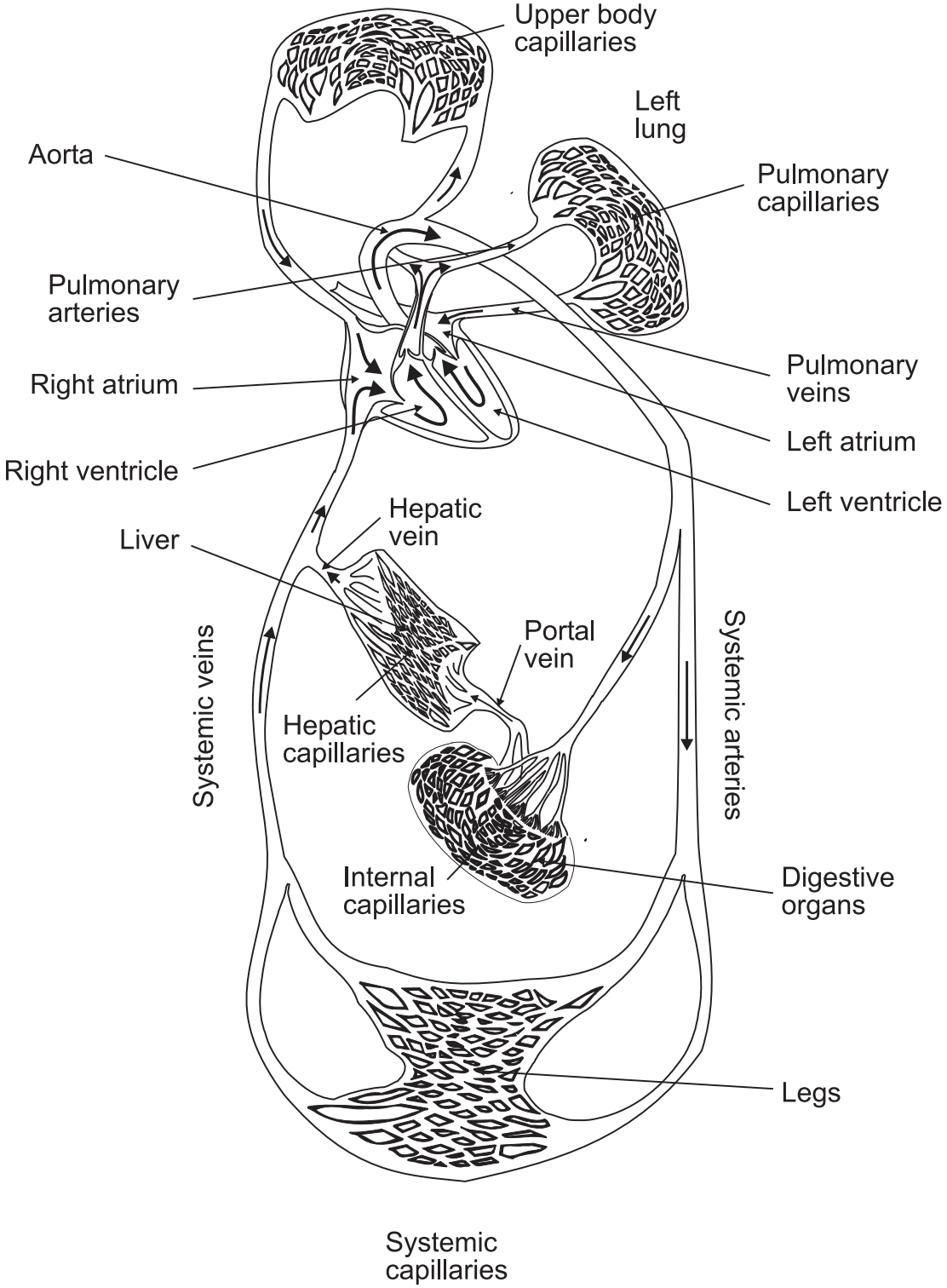

Figure 2.5: Circulatory system of the human body [20]

– Plasma, which forms the primary volume and 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 oxygenised 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).

$$F = \frac{\Delta P}{R} \tag{2.16}$$

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).

$$\Delta P = P_1 - P_2 \tag{2.17}$$

Where P_1 and P_2 are the blood pressures measured at each end of the blood vessel. Pressure has a 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.

2.3 Devices

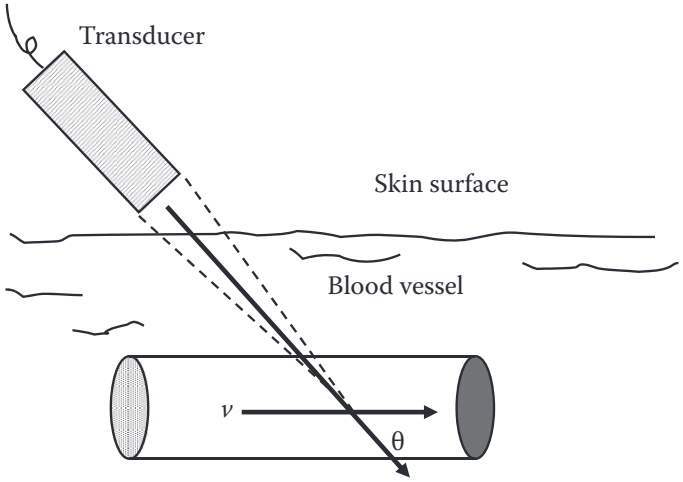

Figure 2.6: Diagram of *US* wave transmitted and reaching blood vessel with incident angle θ [23]

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

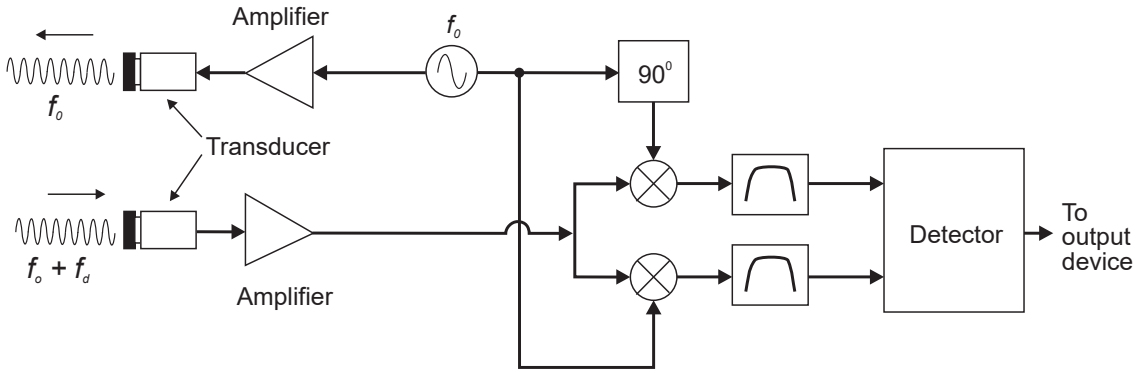

Figure 2.7: Block diagram of CW flowmeter [20]

The earliest non-invasive cardiovascular diagnostic technologies relied heavily on CW Doppler flowmeters. One of the earliest concepts for a device to estimate and study blood flow was proposed by Satomura, Yoshida, Mori, *et al.*[1] during the 1950s in Japan. 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) [20].

$$e(t) = \cos(2\pi f_0 t) \tag{2.18}$$

While receiving the backscattered signal on the second transducer expressed by eq. (2.19) [20].

$$r_s(t) = a\cos\left(2\pi f_0 \alpha(t - t_0)\right) \tag{2.19}$$

$$\alpha \approx 1 - \frac{2v_z}{c} \tag{2.20}$$

$$\alpha t_0 \approx \frac{2d_0}{c} \tag{2.21}$$

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

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

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).

$$m(t) = a\left[\cos(2\pi f_0 t) + j\sin(2\pi f_0 t)\right]\cos(2\pi f_0 \alpha(t - t_0)) \tag{2.23}$$

$$= \frac{a}{2}\Big\{\cos(2\pi f_0[(1-\alpha)t - \alpha t_0]) + \cos(2\pi f_0[(1-\alpha)t - \alpha t_0]) \tag{2.24}$$

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

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.

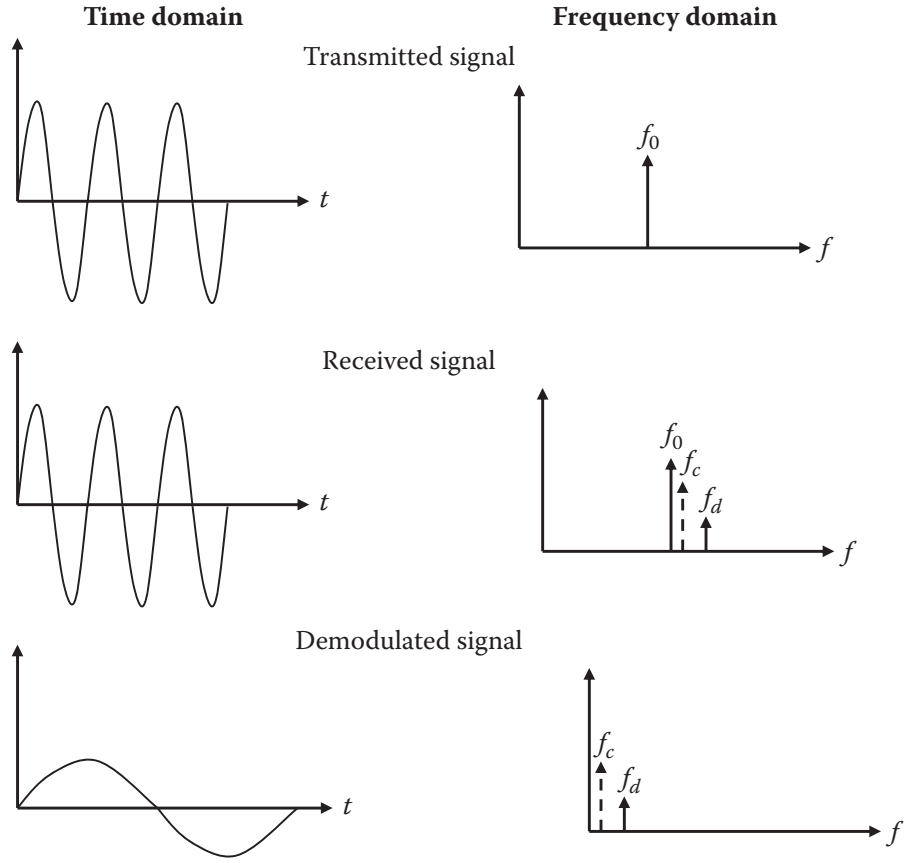

Figure 2.8: Doppler signals in time and frequency domain showing demodulation effects [23]

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.

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

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 2.3: Measured frequency shifts with a Doppler 3 MHz transducer at various velocities at a 45° incident angle [20]

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 2.3 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

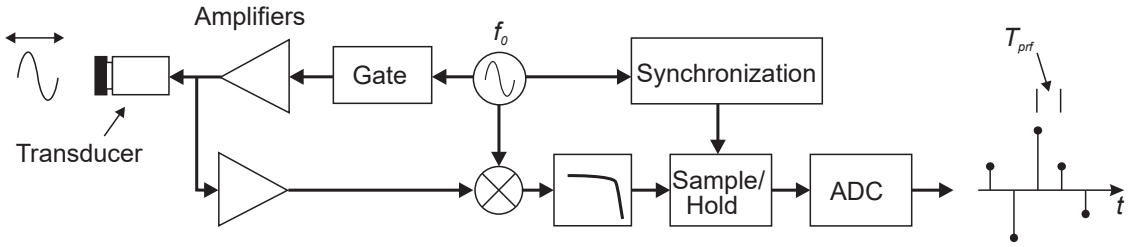

Figure 2.9: Block diagram of PW flowmeter [20]

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.

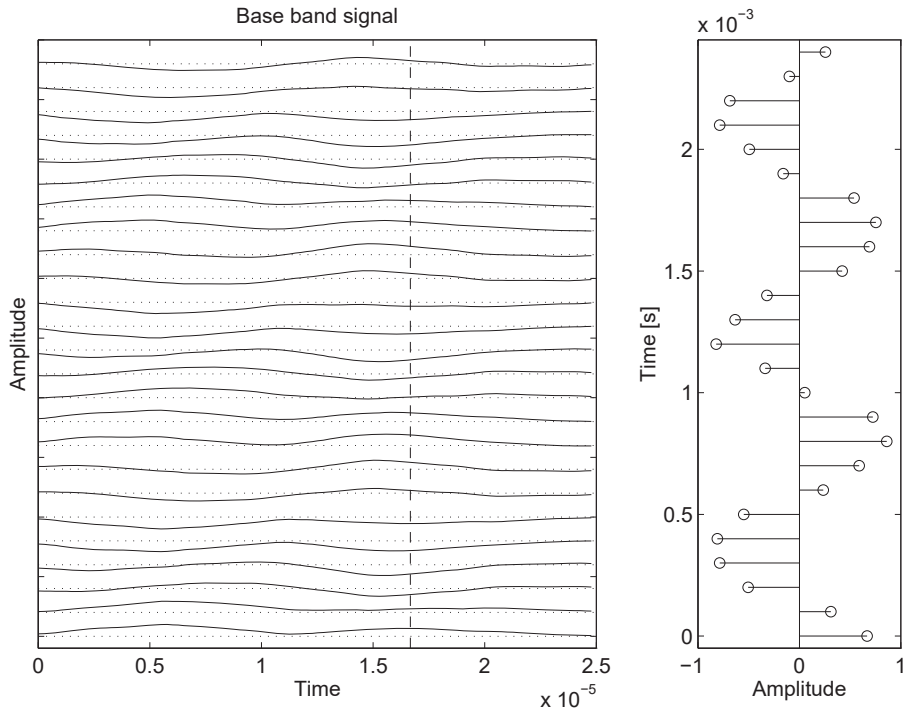

Figure 2.10: Sampling for a gate pulsed wave system with a single range. To depict the signals on the graph, a single pulse is emitted for each line, and the signals are displaced in amplitude. The sampled signal is displayed on the right. [20]

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 [20]. 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. (2.26).

$$d_0 = \frac{T_s c}{2} \tag{2.26}$$

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. The time shift of t_s is expressed as eq. (2.27).

$$t_s = \frac{2v_z}{c} \cdot T_{\text{prf}} \tag{2.27}$$

Where c is the speed of sound, and T_{prf} is the timespan between each pulse emission. Taking one sample from each line at a certain depth yields a sampled signal with a frequency proportional to the scatter velocity. 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 [7] and that signal is expressed by eqs. (2.28a) and (2.28b).

$$r(i) = a(i) \sin\left(2\pi f_p T_{\text{prf}} \cdot i\right) \tag{2.28a}$$

$$f_p = \frac{2v_z}{c} f_0 \tag{2.28b}$$

Where $a(i)$ is the amplitude, f_0 is the emitted frequency, and θ is the phase factor in the depth of interest. This technique improved the accuracy of the investigations of blood vessels and facilitated the display of velocity profiles. Furthermore, by employing two transducers or a multi-element transducer, duplex mode imaging (displaying both a B-mode picture and a blood velocity estimate) became feasible. Two-transducer systems are no longer utilised since it is easier to create a duplex picture with a multi-element transducer.

Hvor er θ i udtrykket? Omskriv fra Jensen2012

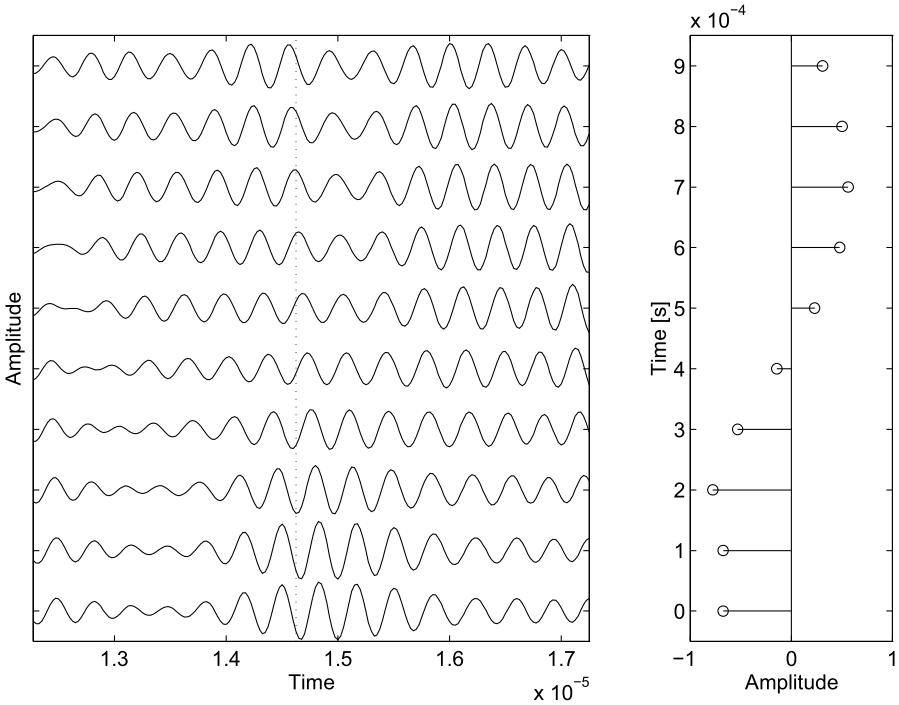

Figure 2.11: Simulated RF sampling of blood vessel. Left graph displays a segmented receiver line with a sample interval denoted by a dotted line, and the right graph shows the resulting amplitude of the sample

Figure 2.11 displays a signal that has been sampled using RF technology. As the scatterers in the image move away from the transducer, 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, using a sample/hold circuit. 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. (2.29).

$$|H_s(f)| = \left| a \frac{\sin(\pi f N T_{\text{prf}})}{\sin(\pi f T_{\text{prf}})} \right| \tag{2.29}$$

which for $f = 0$ is equal to aN and has its first zero at f_{prf}/N. The signal spectrum based on eq. (2.29) is shown in fig. 2.12.

When dealing with small blood vessels, the stationary echoes can be significantly stronger

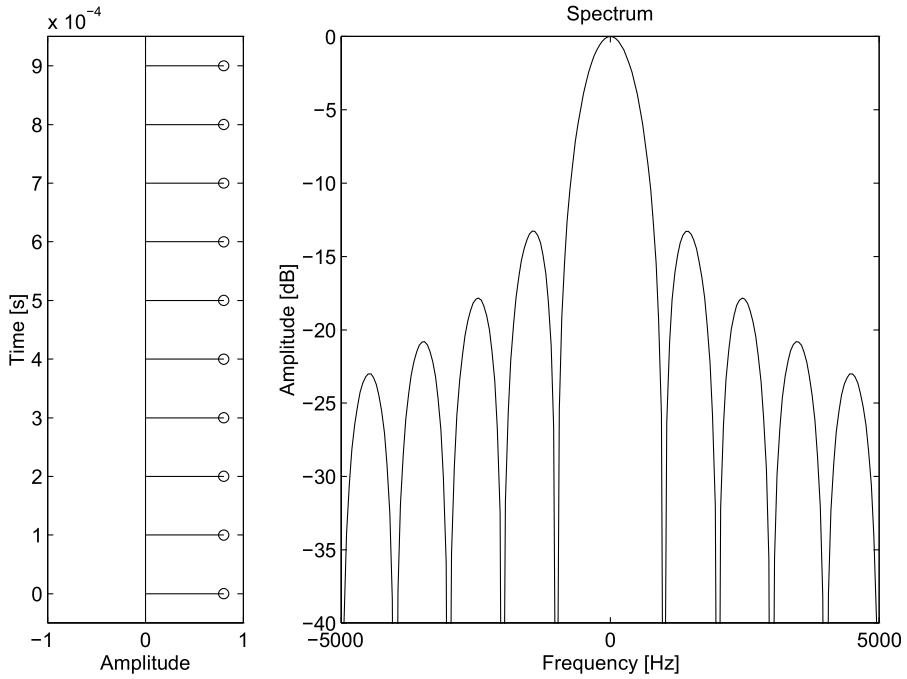

Figure 2.12: Left: Stationary tissue sampled signal. Right: stationary tissue sample spectrum

(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.

2.4 Blood Velocity Estimation

Once a backscattered signal is received, estimation of velocity can be obtained with multiple methods. In this section the most common methods to estimate blood velocity are mentioned with their strengths and limitations.

2.4.1 Spectral estimation

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. A sonography of an artery is displayed in fig. 2.13.

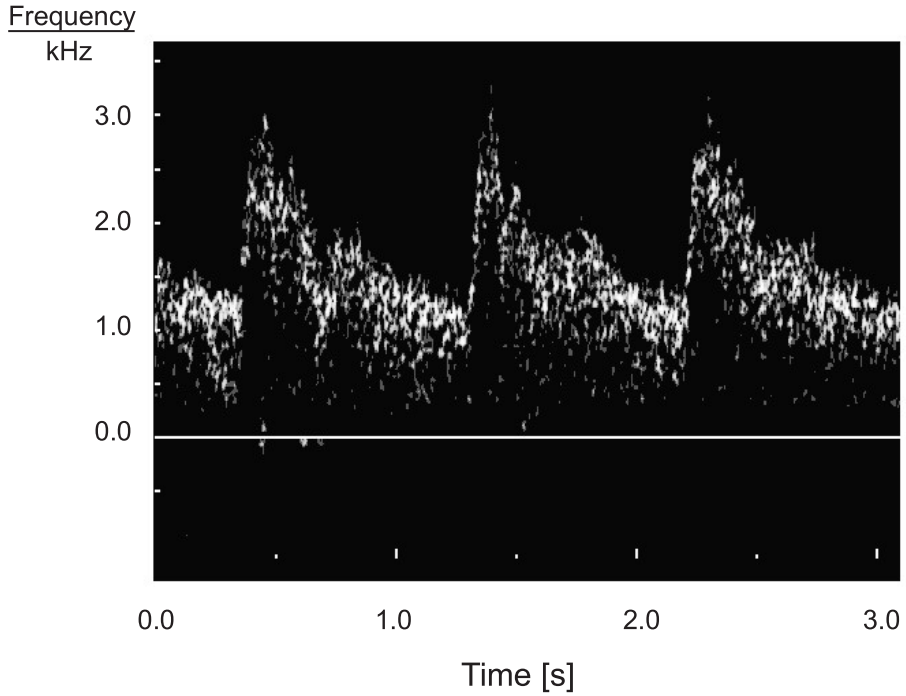

Figure 2.13: Arterial sonogram with time-frequency and Doppler shift [20]

3 Synthesis

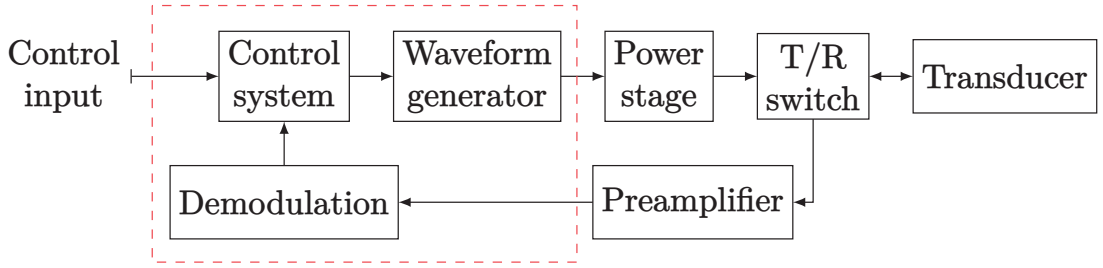

Figure 3.1: Simplified overview of the system with the demodulation and control system indicated with a red dashed rectangle

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 amplifying and demodulating 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 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. Differences between the two options are:

- Multitasking: RTOS allows for parallel execution that enable more complex applications.

- Portability: Standard APIs 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.
- among others.

3.1.1 Development Environment

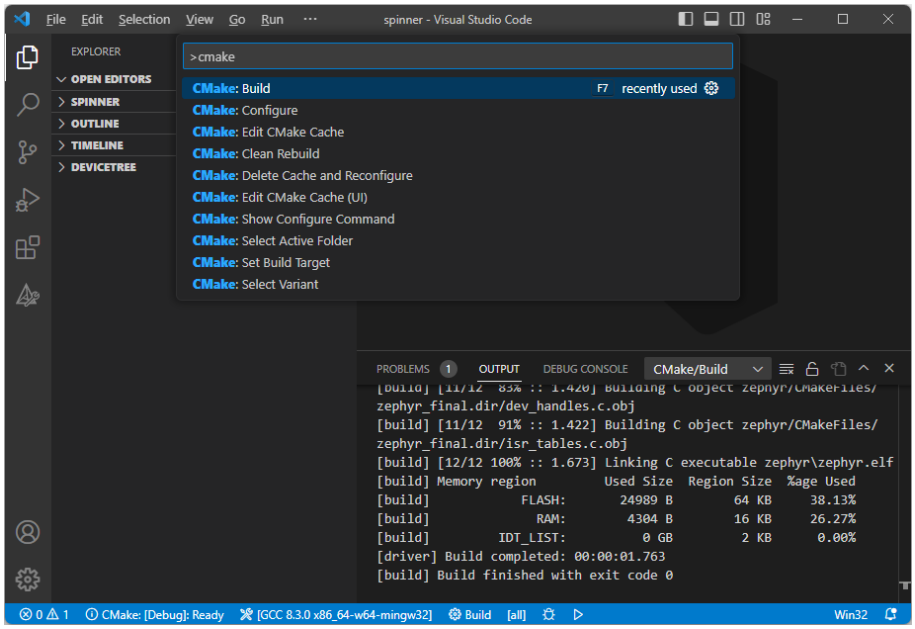

Figure 3.2: VSCode editor with CMake Tools active, displaying available tasks

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 [37] 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. 3.2. In addition to building and running the project, VSCode allows for debugging capabilities to debug code.

3.1.2 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 devicetree nodes and Kconfig to configure the set of build files for ninja. Seen in fig. 3.3 is the

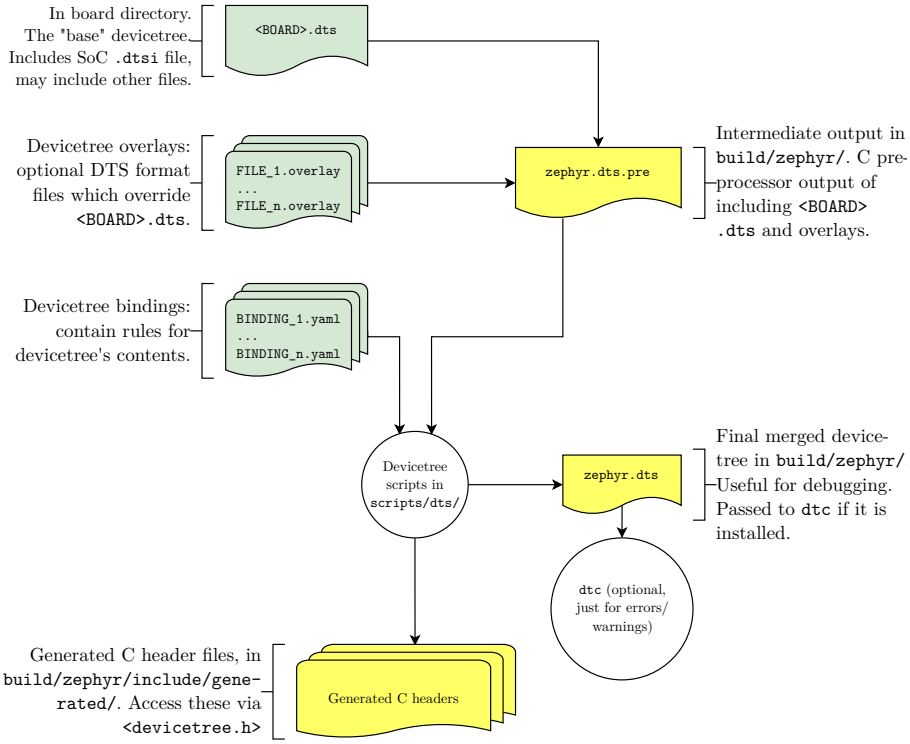

Figure 3.3: Devicetree input files (green) and output files (yellow) [36]

process in which the build system searches out devicetrees 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. 3.4. 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 [35]. 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 CMake's target concept 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.

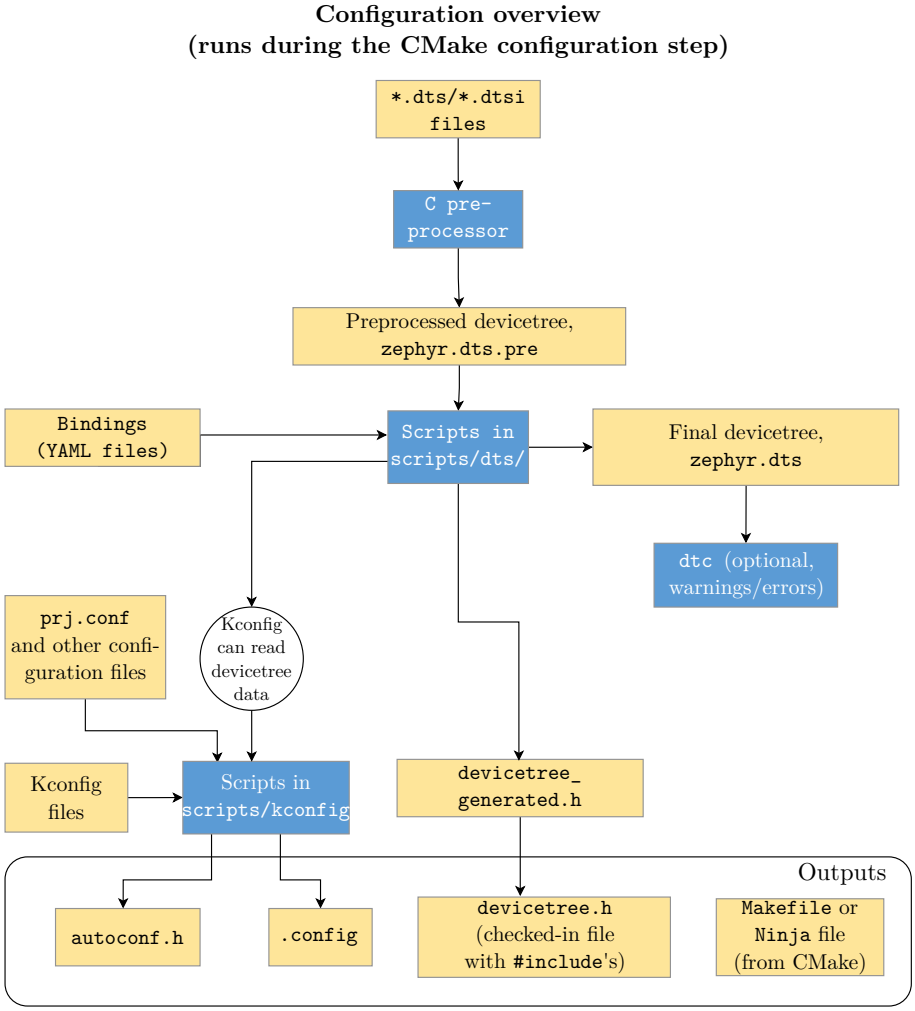

Figure 3.4: Configuration phase of a Zephyr application [36]

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. 3.5 the build system binds system call functions to implementations. Next, in fig. 3.6 the build system collects source files for various subsystems (decided by the configuration phase) and compiles into archives. The `gen_app_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

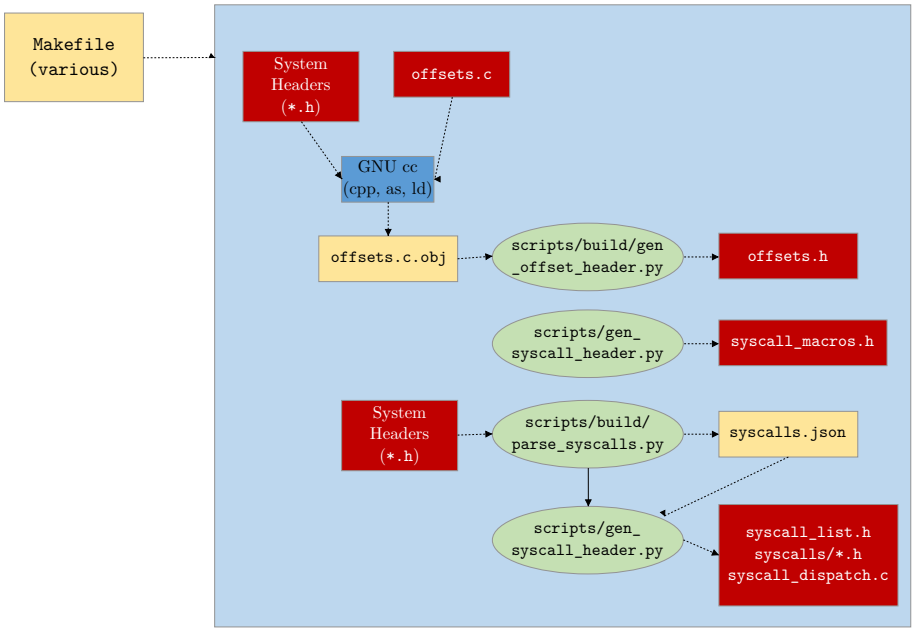

Figure 3.5: Flowchart of build stage I, pre-build

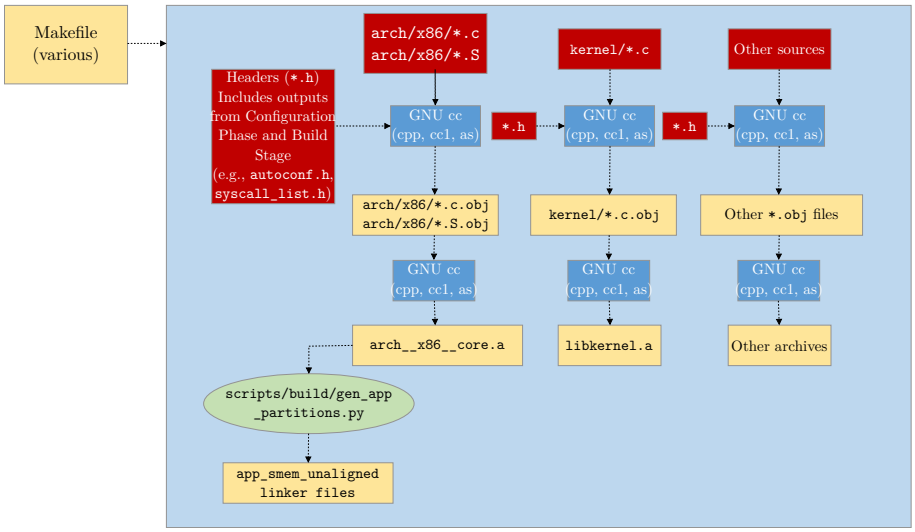

Figure 3.6: Flowchart of build stage II, generation and compilation

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

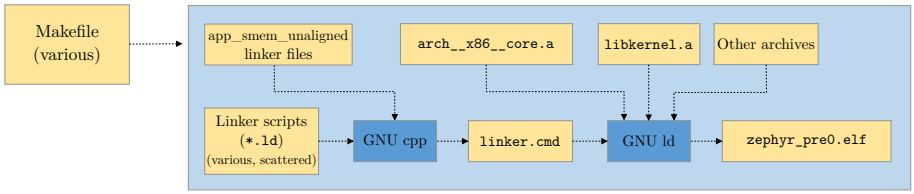

Figure 3.7: Flowchart of build stage III, intermediate binary

in fig. 3.7 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 3.8 the fixed size intermediate binary is

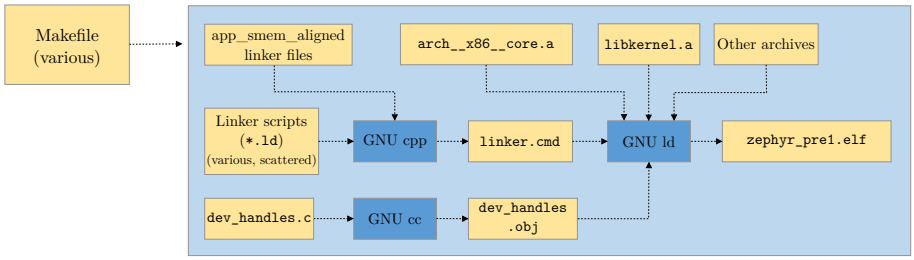

Figure 3.8: Flowchart of build stage IV, second intermediate binary

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

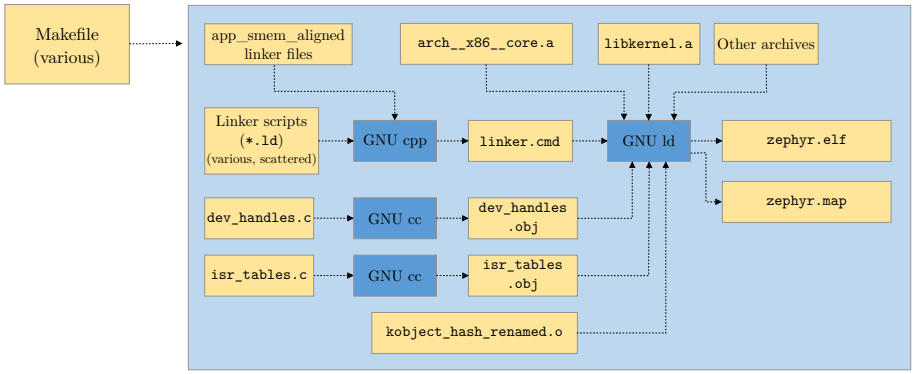

Figure 3.9: Flowchart of build stage V, final binary

fig. 3.9, 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. 3.10, using `GNU objdump`, the 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.

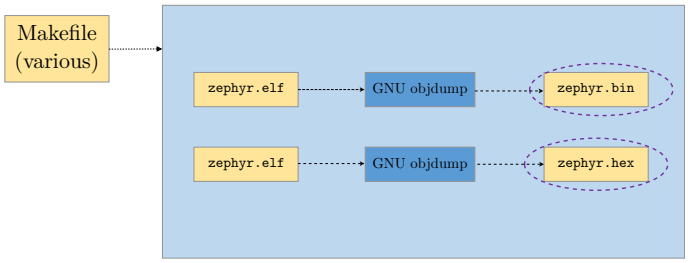

Figure 3.10: Flowchart of build stage VI, post-processing

3.2 Pulse-Width-Modulation Generator

Initially, a waveform 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 more accurate timer based PWM generation is required. 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.

3.3 Power Stage

Gate drivers [10], [28], [33]

3.4 Transmit/Receive Switch

The transmit/receive switch is a module that is based on an IC from Texas Instruments known as TX810[16]. The TX810 is an electronic device that can be used to switch transmit and receive paths of an ultrasound system. It can switch the transmit and receive paths for up to 8 different transducers (also known as probes) 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. This allows the user to perform multi-channel imaging, where multiple transducers are used simultaneously to capture images from different angles. 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

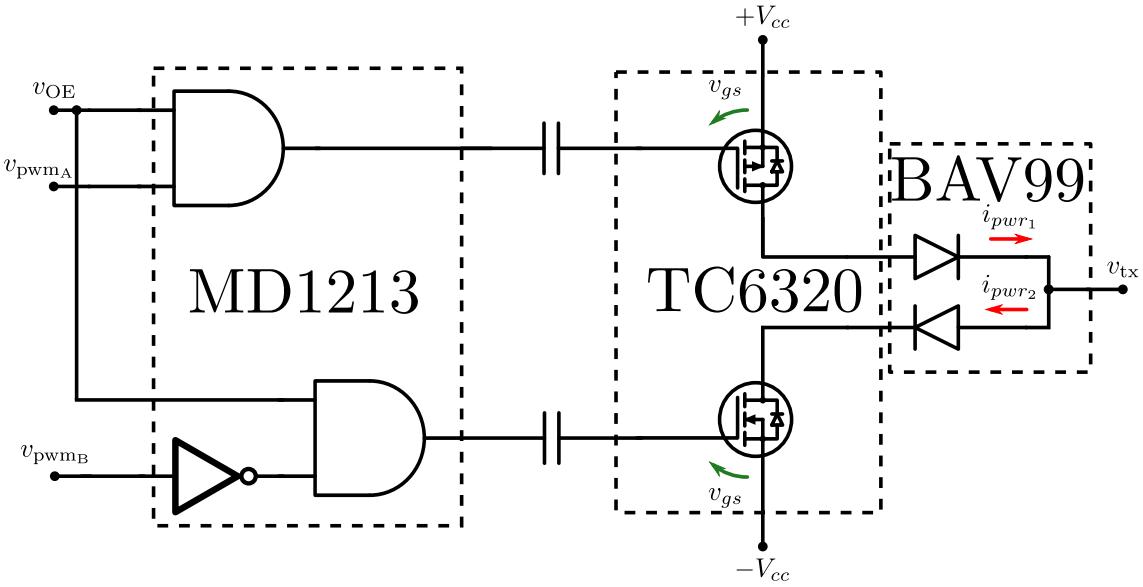

Figure 3.11: Circuit diagram of power stage

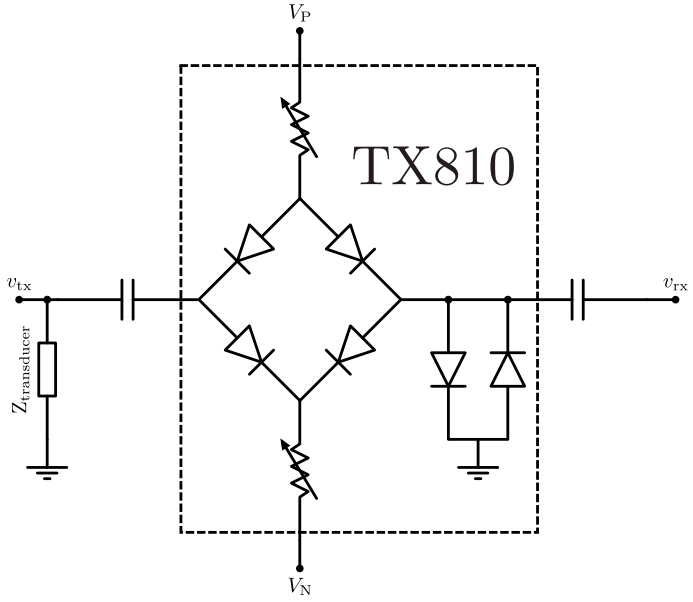

Figure 3.12: Circuit diagram of switch (per channel)

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 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 0mA (high-impedance mode). The TX810 does not put significant load on high-voltage transmitters when operating in the high-impedance mode. The device can also wake up from power-down mode in less than a ţs. These sophisticated programmable features enable systems to save a large amount of electricity.

The module is designed with three channels available. That means three channels can be used, either three separate transducers for multi-angle sonography, or a capacitive micromachined ultrasound transducer ($CMUT$) with three channels in a single angle.

3.5 Transducer

Various transducer types are used during the testing of this project. Initially, commercially available transducers were used in order to validate the experimental process of the pulse-echo and Doppler measurements. Commercially available transducers in this context means lead circonate titanate (PZT) piezoelectric transducers. PZT is a piezoelectric material that deforms when an electrical voltage is applied, and generates an electrical signal when it is mechanically deformed. PZT transducers are widely used for medical imaging due to their high piezoelectric efficiency, high bandwidth and durability.

At a later stage during the project, CMUTs are used. CMUT, is a newer technology that uses capacitance to convert between electrical and mechanical energy. The transducer consists of a flexible membrane that moves in response to an applied voltage, producing ultrasound signals. CMUTs have advantages in terms of their small size, high frequency, and ability to operate in a broad range of conditions, making them ideal for use in portable and high-resolution imaging devices. There are two types of CMUTs used:

Rigid CMUTs:

- Advantages:
  - Higher mechanical stability, leading to less noise and improved image quality
  - Can handle higher pressures, making them suitable for deeper imaging applications
  - Can have a larger surface area, providing higher sensitivity
- Disadvantages:
  - Less flexible, limiting their use in applications with complex geometries or limited access
  - Higher stiffness, leading to increased power consumption and difficulty in producing high-frequency signals

Flexible CMUTs:

- Advantages:
  - More flexible, allowing them to conform to complex geometries or be used in applications with limited access

  - Can be made thinner and lighter, making them suitable for portable or handheld imaging devices
  - Can be integrated into wearable devices or implanted into the body, enabling new imaging modalities
- Disadvantages:
  - Lower mechanical stability, leading to increased noise and reduced image quality
  - Lower maximum operating pressure, limiting their use in deep imaging applications
  - Smaller surface area, resulting in reduced sensitivity

In summary, the choice between rigid and flexible CMUTs depends on the specific requirements of the application and the trade-off between flexibility, sensitivity, and image quality. For this project, a wearable transducer is desired and therefore the flexible CMUT is preferred.

3.5.1 Impedance matching

Impedance matching refers to the adjustment of the output impedance of the transducer to match the input impedance of the receiving system, such as the ultrasound machine or amplifier. This ensures maximum transfer of energy from the transducer to the receiving system, leading to improved signal strength and reduced noise in the final image. In ultrasound imaging, the transducer is used to both transmit and receive ultrasound signals. The impedance of the transducer must be carefully matched to the environment it is operating in, to ensure efficient transfer of energy both ways. If the impedance of the transducer and the receiving system do not match, a portion of the transmitted energy will be reflected back to the transducer, leading to reduced signal strength and increased noise in the final image. This can also result in increased power consumption and reduced battery life in portable or handheld imaging devices.

3.6 Preamplifier

OPA487

The OPA847[12] is a high-performance, high-speed, voltage feedback amplifier with a bandwidth of over 1 GHz. It has a wide gain range, low distortion, and low noise, making it suitable for a variety of applications including video, RF, and high-speed data acquisition. The OPA847 has a single-ended input and a differential output, which allows it to be used in a variety of circuit configurations. It is available in a surface-mount package and operates over a wide supply voltage range.

AD8332

The AD8332[24] is a fully integrated, single-chip amplifier designed for use in a wide range of RF and microwave applications. It is a low-noise, high-linearity amplifier that can be used in a variety of configurations, including as a stand-alone amplifier or as part of a larger system. The AD8332 is a current-feedback amplifier that can operate over a wide frequency range, from 50 MHz to 4 GHz. It has a high level of integration, with all active components on a single monolithic IC. This makes it suitable for use in small form factor applications where space is at a premium. The AD8332 has a number of features that make it well-suited for use in RF and microwave applications. It has a high level of linearity, which allows it to amplify signals without introducing significant distortion. It also has a low noise figure, which makes it well-suited for use in low-noise applications. Additionally,

the AD8332 has a wide dynamic range, which makes it capable of handling a wide range of input signals without clipping or saturating. Overall, the AD8332 is a versatile and reliable amplifier that can be used in a wide range of RF and microwave applications. It is well-suited for use in a variety of systems, including communication systems, radar systems, and instrumentation systems.

3.7 Demodulation

To demodulate our signal using the process described in The AD8333[25] can be used to demodulate amplitude modulation (AM) signals. It contains all the necessary circuitry to detect and extract the information contained in an AM signal. In AM, the information is encoded in the amplitude of the carrier wave, which is modulated or varied in some way to convey the information. The AD8333 is able to detect these variations in the carrier wave and extract the original information from the signal. To do this, the AD8333 uses a process called envelope detection. It rectifies the input signal, which removes the negative portions of the waveform, and then filters the resulting signal to remove any remaining high-frequency components. This leaves only the envelope of the original modulated signal, which contains the encoded information. The AD8333 also includes amplifier and buffer stages to amplify the detected envelope and prepare it for further processing or application. It is often used in radio communication systems, industrial control systems, and other applications where an AM signal needs to be demodulated and the information contained in the signal needs to be extracted. Test af git push.

3.8 Sample/Hold

The sample/hold circuit is based on the AD783[21] sample-and-hold amplifier (SHA). It functions by converting an analog input signal into a sequence of quantized samples and holds these samples for a specified time. The SHA is made up of several internal stages, including an input amplifier, sample switch, sample-and-hold capacitor, and output amplifier. When the sample input of the AD783 is triggered, the input amplifier amplifies the input signal and the sample switch closes, allowing the signal to be stored on the sample-and-hold capacitor. Once the sample is taken, the sample switch opens, and the output amplifier outputs the stored value of the input signal. The hold input of the AD783 maintains the sample on the capacitor until the next sample is taken.

3.9 Pulse-Repetition and Wall Filter

3.10 Amplifier

3.11 Mixer

3.12 Adder

4 Production

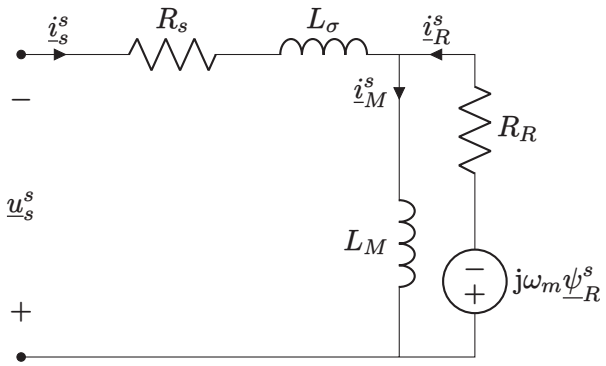

Figure 4.1: The nodes short, V, R and L are presented here, but there a lot more

4.1 Listings (code)

Listing 4.1 is a nicely formatted block of code. A listing will automatically continue on the next page if it encounters a page break. Many different programming languages can be highlighted. Check the `listings` package documentation for a list of supported programming languages.

```c
#include <stdio.h>
int main()
{
        printf("Hello, World!"); /*printf() outputs the quoted string*/
        if (n == 0 || n == 1){
                return n;
        }
        j = 0;
        for (i = 0; i < n-1; i++){
                if (arr[i] != arr[i+1]){
                        arr[j] = arr[i];
                        j++;
                }
        }
        arr[j++] = arr[n-1];
        return 0;
}
```

Code 4.1: Hello world in C

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 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.

[4] 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.

[5] 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.

[6] 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.

[7] P. Munk, "Estimation of blood velocity vectors using ultrasound," Technical University of Denmark, 2000. [Online]. Available: https://findit.dtu.dk/en/catalog/537f0d537401dbcc1200be04.

[8] 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.bmj.com/lookup/doi/10.1136/bmj.328.7439.578.

[9] 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.

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

[11] X. Xu, J. Yen, and K. K. Shung, "A low-cost bipolar pulse generator for high-frequency ultrasound applications," *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 54, pp. 443–447, 2 February 2007, ISSN: 0885-3010. DOI: 10.1109/tuffc.2007.259. [Online]. Available: http://ieeexplore.ieee.org/document/4107704/.

[12] *OPA847 Wideband VF Operational Amplifier*, Texas Instruments, Inc., 2008. [Online]. Available: https://www.ti.com/lit/ds/symlink/opa847.pdf.

[13] 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/.

[14] E. A. Aydn 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.

[15] 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.

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

[17] 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-esti.

[18] 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/.

[19] 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/.

[20] 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.

[21] *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.

[22] 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.

[23] K. K. Shung, *Diagnostic Ultrasound: Imaging and Blood Flow Measurements*, Second Edition. CRC Press, December 2015, ISBN: 978-1-4665-8264-4.

[24] *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.

[25] *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.

[26] 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.

[27] 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/.

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

[29] 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/.

[30] 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/.

[31] 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/.

[32] 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.

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

[34] 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 20102020 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.

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

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

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

A Source Code

Code A.1: main zephyr

```c
#include <zephyr/kernel.h>
#include <zephyr/device.h>
#include <zephyr/drivers/uart.h>
#include <stm32f4xx_hal.h>
#include <soc.h>

#include <string.h>
#include "main.h"
//#include "hal_msp.h"

TIM_HandleTypeDef htim1;
/* queue to store up to 10 messages (aligned to 4-byte boundary) */
K_MSGQ_DEFINE(uart_msgq, MSG_SIZE, 10, 4);
static const struct device *const uart_dev = DEVICE_DT_GET(UART_DEVICE_NODE);
static int rxFlag = 0;

/* receive buffer used in UART ISR callback */
static char rx_buf[MSG_SIZE];
static int rx_buf_pos;

void main(void)
{
  HAL_Init();
  HAL_MspInit();
        //SystemClock_Config();
  MX_TIM1_Init();
        MX_GPIO_Init();
  char tx_buf[MSG_SIZE];

        if (!device_is_ready(uart_dev)) {
                printk("UART device not found!");
                return;
        }

        /* configure interrupt and callback to receive data */
        int ret = uart_irq_callback_user_data_set(uart_dev, serial_cb, NULL);

        if (ret < 0) {
                if (ret == -ENOTSUP) {
                        printk("Interrupt-driven UART API support not enabled\n");
                } else if (ret == -ENOSYS) {
                        printk("UART device does not support interrupt-driven API\n");
                } else {
                        printk("Error setting UART callback: %d\n", ret);
                }
                return;
        }
        uart_irq_rx_enable(uart_dev);

        print_uart("Hello! I'm your echo bot.\r\n");
        print_uart("Tell me something and press enter:\r\n");
        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
        HAL_TIMEx_PWMN_Start(&htim1, TIM_CHANNEL_1);
        TIM1->CCR1=(int)htim1.Init.Period/2;
```

```
55          while (1) {
56            //if (k_msgq_get(&uart_msgq, &tx_buf, K_MSEC(1)) == 0) {
57      if(rxFlag == 1) {
58      k_msgq_get(&uart_msgq, &tx_buf, K_MSEC(1));
59                  print_uart("Echo: ");
60                  print_uart(tx_buf);
61                  print_uart("\r\n");
62      rxFlag = 0;
63          }
64                  HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
65                  k_msleep(SLEEP_TIME_MS);
66          }
67  }
68
69  void Error_Handler(void)
70  {
71    /*__disable_irq();
72    while (1)
73    {
74      printk("ERROR!\r\n");
75    }*/
76  }
77
78  void serial_cb(const struct device *dev, void *user_data)
79  {
80    //Read characters from UART until line end is detected. Afterwards push the data to the message queue.
81          uint8_t c;
82
83          if (!uart_irq_update(uart_dev)) {
84                  return;
85          }
86
87          if (!uart_irq_rx_ready(uart_dev)) {
88                  return;
89          }
90
91          //read until FIFO empty
92          while (uart_fifo_read(uart_dev, &c, 1) == 1) {
93                  if ((c == '\n' || c == '\r') && rx_buf_pos > 0) {
94                          //terminate string
95                          rx_buf[rx_buf_pos] = '\0';
96
97                          //if queue is full, message is silently dropped
98                          k_msgq_put(&uart_msgq, &rx_buf, K_NO_WAIT);
99
100                         //reset the buffer (it was copied to the msgq)
101                         rx_buf_pos = 0;
102                 } else if (rx_buf_pos < (sizeof(rx_buf) - 1)) {
103                         rx_buf[rx_buf_pos++] = c;
104                 }
105                 //else: characters beyond buffer size are dropped
106         }
107   rxFlag = 1;
108 }
109
110 void print_uart(char *buf)
111 {
112   //Print a null-terminated string character by character to the UART interface
113         int msg_len = strlen(buf);
114
115         for (int i = 0; i < msg_len; i++) {
```

```
                uart_poll_out(uart_dev, buf[i]);
        }
}

void HAL_TIM_MspPostInit(TIM_HandleTypeDef* htim)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(htim->Instance==TIM1)
  {
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    //TIM1 GPIO Configuration
    //PA7     ------> TIM1_CH1N
    //PA8     ------> TIM1_CH1
    GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_8;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF1_TIM1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = GPIO_PIN_14;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF1_TIM1;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
  }
}

static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    //GPIO Ports Clock Enable
          __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();

    //Configure GPIO pin Output Level
    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

    //Configure GPIO pin : B1_Pin
    GPIO_InitStruct.Pin = B1_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

    //Configure GPIO pin : LD2_Pin
    GPIO_InitStruct.Pin = LD2_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);
}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
```

```
177
178   // Configure the main internal regulator output voltage
179   __HAL_RCC_PWR_CLK_ENABLE();
180   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
181
182   // Initializes the RCC Oscillators according to the specified parameters in the RCC_OscInitTypeDef stru
183   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
184   RCC_OscInitStruct.HSIState = RCC_HSI_ON;
185   RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
186   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
187   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
188   RCC_OscInitStruct.PLL.PLLM = 8;
189   RCC_OscInitStruct.PLL.PLLN = 100;
190   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
191   RCC_OscInitStruct.PLL.PLLQ = 4;
192   if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
193   {
194     printk("HAL_RCC_OscConfig\r\n");
195     Error_Handler();
196   }
197
198   // Initializes the CPU, AHB and APB buses clocks
199   RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
200                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
201   RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
202   RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
203   RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
204   RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
205
206   if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
207   {
208     printk("HAL_RCC_ClockConfig\r\n");
209     Error_Handler();
210   }
211 }
212
213 static void MX_TIM1_Init(void)
214 {
215   TIM_ClockConfigTypeDef sClockSourceConfig = {0};
216   TIM_MasterConfigTypeDef sMasterConfig = {0};
217   TIM_OC_InitTypeDef sConfigOC = {0};
218   TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};
219   htim1.Instance = TIM1;
220   htim1.Init.Prescaler = 0;
221   htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
222   htim1.Init.Period = 19;
223   htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
224   htim1.Init.RepetitionCounter = 0;
225   htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
226   if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
227   {
228     printk("HAL_TIM_Base_Init\r\n");
229     Error_Handler();
230   }
231   sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
232   if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
233   {
234     printk("HAL_TIM_ConfigClockSource\r\n");
235     Error_Handler();
236   }
237   if (HAL_TIM_PWM_Init(&htim1) != HAL_OK)
238   {
```

```
239       printk("HAL_TIM_PWM_Init\r\n");
240       Error_Handler();
241     }
242     sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
243     sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
244     if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
245     {
246       printk("HAL_TIMEx_MasterConfigSynchronization\r\n");
247       Error_Handler();
248     }
249     sConfigOC.OCMode = TIM_OCMODE_PWM1;
250     sConfigOC.Pulse = 0;
251     sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
252     sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
253     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
254     sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
255     sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
256     if (HAL_TIM_PWM_ConfigChannel(&htim1, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
257     {
258       printk("HAL_TIM_PWM_ConfigChannel\r\n");
259       Error_Handler();
260     }
261     sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
262     sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
263     sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
264     sBreakDeadTimeConfig.DeadTime = 1;
265     sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
266     sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
267     sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
268     if (HAL_TIMEx_ConfigBreakDeadTime(&htim1, &sBreakDeadTimeConfig) != HAL_OK)
269     {
270       printk("HAL_TIMEx_ConfigBreakDeadTime\r\n");
271       Error_Handler();
272     }
273     HAL_TIM_MspPostInit(&htim1);
274
275 }
```

B Bill of Materials

Table B.1: Bill of Materials for the entire system

Component	Value	Footprint	Classification	Description
		Transmitter		
C1	1n	0603	X7R50V	Preamp
C13	1u	RAD-0.3in	Film 40V	Preamp
C14	1.5n	0603	X7R50V	PI controller
C15	NC	0603	X7R50V	PI controller
C2,C3,C4	100n	0603	X7R16V	Decoupling
C5,C11	1u	0603	X5R6.3V	Decoupling
C6,C7	100n	0805	X7R50V	Decoupling
C8	1500u	RAD-0.3in	63Vdc	Decoupling
P10	3-pin	Male	250V	Controller bypass
P11,P12	10-pin	Female	250V	InterPCB con.
P2,P3	1-pin header	Male	250V	Measurements
P4,P5	2-way screw	NA	300V 15A	Power,Output
P6	2-pin header	KK254	500V 4A	Audio in
P7	4-pin header	Female	250V	5V Power
P8,P9	BNC	PCB	500V	Audio in, Output
R2	16.2k	0603	1%	Preamp
R1,R3	2k	0603	1%	Preamp
R4,R7	4.75k	0603	1%	Voltage ref
R5	33.2k	0603	NA	PI controller
R6	0 (short)	0603	NA	PI controller
R9	0 (short)	0603	NA	PI controller
R10	NC	0603	NA	PI controller
R8	2.49k	0603	1%	Voltage ref
U1	OPA2365	SOIC-8	NA	Preamp, PI
U2	TLV431A	SOT-23	1%	Voltage ref
		Receiver		
C1,C11,C15	100n	0603	X7R16V	Gate driver
P1,P2	10-pin header	Male	250V	InterPCB con.
C2,C16	150n	0603	X5R10V	Gate driver
C3,C10	100p	0603	NPO50V	Decoupling

Continued on next page

Table B.1: Bill of Materials for the entire system (Continued)

Component	**Value**	**Footprint**	**Classification**	**Description**
L1,L2	1.768u	Radial	NA	Output filter
U1,U3	LM5113	WSON-10	NA	Gate driver
R20	15m sense	1210	1% 1W	Output filter
R1,R6	500	3213	NA	Gate driver
Q1,Q2,Q3,Q4	BSZ097-N10NS5	TSDSON-8	NA	Power stage
R2,R5,R8,R10	5	0603	1%	Gate driver
R4,R9	0	0603	1%	Gate driver
D1,D2	Diode	NA	85V 0.25A	Gate driver
C5,C6,C9,C17	10u	1210	X7R50V	Decoupling
C12,C13,C14	680n	1210	X7R100V	Output filter
		Microcontroller		
C18,C20,C27	100n	0603	X7R16V	Decoupling
C28,C29	100n	0603	X7R16V	Decoupling
U6	LT1999	MSOP-8	NA	Current acq.
U4	AD8274	MSOP-8	NA	Volt acq.
U2	LT1711	MSOP-8	NA	AIM
C4	NC	0603	X7R16V	AIM
C7	NC	0603	X7R16V	AIM
C8	NC	0603	X7R16V	Cur. acq.
CA1	1.5n	0603	X7R50V	AIM
R14	8.66k	0603	1%	AIM
R18	16.2k	0603	1%	AIM
R23	1.5k	0603	1%	AIM
R3,R11	120k	0603	1%	Voltage acq.
RA1	2.2k	0603	1%	AIM
RA2	20k	0603	1%	AIM
RA3	2.74k	0603	1%	AIM
RA4	0	0603	NA	AIM

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
SMT solder paste dispensing	Fisnar	SL101N
Reflow oven	Mistral	260
DMM	Meterman	38XR

Table C.2: List of instruments used for testing

Function	**Manufacturer**	**Model**
Power supply	RIGOL	DP832A
Function generator	Keysight	33500B
DMM	Fluke	1587
Frequency response analysis	N4L	PSM1735
Audio analysis	Audio Precision	APx500
Audio analysis filter	Audio Precision	AUX-0025
Control loop analyzer	OMICRON Lab	BODE 100
Control loop injection transformer	OMICRON Lab	B-WIT 100
Passive load	Unknown	Power resistor 4 Ω
Passive load	Unknown	Loudspeaker 4 Ω
Active cooling	ebm	**Test of bold**