FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Object detection in protected area using dToF sensors in automotive environment
Student:	Bc. Petr Moucha
Supervisor:	Ing. Jiří Andrle
Study program:	Informatics
Branch / specialization:	Design and Programming of Embedded Systems
Department:	Department of Digital Design
Validity:	until the end of summer semester 2022/2023

Instructions

Design and realize a setup performing object detection in a protected area. The setup will consist of a dedicated model of a protected area and VL53Lx family dToF sensor connected to SPC58x automotive MCU. Create new or adapt existing PCB with SPC58x MCU and connect it with one or more dToF sensors integrated inside the model. Implement SW stack for MCU platform capable to control dToF sensor(s) and capture data. Implement detection algorithm on MCU platform based on sensor output data. Evaluate the performance of detection using a defined set of objects (pen, coin, credit card). Object presence inside a protected area will be indicated by an optical or acoustic signal.

Master's thesis

OBJECT DETECTION IN PROTECTED AREA USING DTOF SENSORS IN AUTOMOTIVE ENVIRONMENT

Bc. Petr Moucha

Faculty of Information Technology
Department of Digital Design
Supervisor: Ing. Jiří Andrle
January 5, 2023

Citation of this thesis: Moucha Petr. *Object detection in protected area using dToF sensors in automotive environment.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

List of Code listings

First, I want to thank my supervisor Jiří Andrle, for the guidance and invaluable assistance he provided from the start to finish of this project. It also goes without saying that my gratitude extends to the entire team at STMicroelectronics for constantly sharing their incredible insights and expertise over a cup of coffee. I'm also indebted to my good friends. Especially to Michal Flodr for his advice on the UX design of the desktop application and Kryšpín Varyš for final grammar and sanity checks. I look forward to the next decade of our friendship and sincerely hope there'll be many more to come! Last but not least, I thank my partner and my family for their patience and constant support.

Declaration

In Prague on January 5, 2023 ....

Abstract

This work describes the implementation of a device that aims to guard a defined area inside a car against the intrusion of unwanted objects. The protected area is covered with *direct Time-of-Flight* (dToF) sensors from STMicroelectronics, which are qualified for automotive applications. The device has been designed from the ground up to be a self-contained unit that includes not only the protected area itself but also all the electronics needed to read the sensor data and evaluate the presence of objects. The basic detection algorithm has been implemented on a PowerPC microcontroller and its results are signaled in real time by an LED. All integrated circuits and other electrical components were soldered on custom PCBs designed specifically for this thesis. The device can optionally be connected via USB cable to a desktop user interface that can graphically represent the sensor data, but in addition, it also allowed for easier development of a more advanced version of the detection algorithm. This external variant was also used for final testing, which showed that the system could correctly respond to the presence of most spatially significant objects, but for example a coin and other objects with low reflectivity could not be reliably detected over the entire protected area. Several external factors were also discovered that further negatively affect the algorithm and should be taken into account in future versions.

Keywords object detection, dToF, Time-of-Flight, optical sensors, depth sensing, automotive industry, PCB design

Abstrakt

Práce popisuje realizaci zařízení, jehož cílem je hlídat vymezenou oblast uvnitř automobilu před vniknutím nežádoucích objektů. Oblast je snímána pomocí *direct Time-of-Flight* (dToF) senzorů od firmy STMicroelectronics, které jsou kvalifikovány pro využití v automobilovém průmyslu. Zařízení bylo od základu navrhnuto tak, aby se jednalo o samostatně fungující jednotku, jejíž součástí bude nejen chráněná oblast, ale i veškerá elektronika, která je potřeba k vyčtení dat ze senzorů a vyhodnocení přítomnosti objektů. Základní detekční algoritmus byl implementován na PowerPC mikrokontroléru a jeho výsledek je v reálném čase světelně signalizován LED diodou. Všechny čipy a další elektrické komponenty byly osazeny na tištěné spoje navržené speciálně pro tuto práci. Zařízení je volitelně možné přes USB kabel připojit k desktopovému uživatelskému rozhraní, které dokáže graficky reprezentovat data ze senzorů, ale mimojiné také umožňilo snadnější vývoj pokročilejší varianty detekčního algoritmu. Tato externí varianta byla využita i pro závěrečné testování, které ukázalo, že systém dokáže správně zareagovat na přítomnost většiny prostorově významných předmětů, ale například minci a další objekty s nízkou reflektivitou nebylo možné spolehlivě detekovat po celé ploše oblasti. Byla také odhalena řada externích faktorů, které algoritmus dále negativně ovlivňují a měly by být zohledněny v budoucích verzích.

Klíčová slova detekce objektů, dToF, Time-of-Flight, optické senzory, měření vzdálenosti, automobilový průmysl, návrh tištěného spoje

List the abbreviations

ADC	Analog-to-digital convertor
API	Application programming interface
ASCII	American Standard Code for Information Interchange
CAD	Computer-aided design
CAN	Controller Area Network
CMOS	Complementary Metal-Oxide-Semiconductor
DC	Direct current
DCR	Dark count rate
dToF	Direct Time-of-Flight
EEPROM	Electrically erasable programmable read-only memory
EMC	Electromagnetic compatibility
ESD	Electrostatic discharge
FoV	Field of View
GPIO	General-purpose input/output
GPU	Graphics processing unit
GUI	Graphical user interface
I^2C	Inter-Integrated Circuit
IC	Integrated circuit
IDE	Integrated Development Environment
iToF	Indirect Time-of-Flight
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
MCU	Microcontroller unit
OEM	Original equipment manufacturer
PCB	Printed circuit board
PTC	Positive temperature coefficient
QE	Quantum efficiency
RGB	Red-Green-Blue
SNR	Signal-to-noise ratio
SPAD	Single-photon avalanche diode
SPI	Serial Peripheral Interface
TDC	Time-to-digital converter

ToF	Time-of-Flight
TVS	Transient-voltage-suppression
UART	Universal asynchronous receiver-transmitter
USB	Universal Serial Bus
VCSEL	Vertical-cavity surface-emmiting laser

Introduction

In this work, a novel use case for the *direct Time-of-Flight* (dToF) sensor in the automotive environment was explored. This type of sensor can accurately estimate distance by sending a laser pulse of infrared light at the target and measuring the time it takes for it to reflect back to the source. In the simplest case, the reflected light is captured with a single-zone detector. It is also possible to create a pixel array with a more substantial resolution that can measure arrival times at multiple points to create detailed height maps of the scene. However, there is no guarantee that the reflected light will return along the same straight path from which it originated. The pixels also have a field of view that slowly diverges with increasing distance, allowing other light sources to cause unwanted interference. Therefore, the measured distance is always a value extracted with statistical methods from aggregate data gathered in the entire field of view.

The main goal is to find out how usable the single-zone automotive dToF sensors from STMicroelectronics are in situations where only a subset of the natural field of view is of interest. This subset will be referred to as a protected area because the purpose of the sensors is to detect if the area is occupied by foreign objects. The protected area will have a rectangular shape defined with a physical frame. Naturally, the frame and also the surface will become the default targets captured by the sensors, so the question is whether additional objects placed *only* within the protected area can cause a detectable change in the distance reading and other measured quantities. It will also be important to find the best physical sensor placement for limiting the influence of objects that are in the field of view but not in the protected area and should therefore be ignored.

It is fundamental to understand the underlying technology, so the first chapter contains a brief introduction to different depth-sensing systems and also explains the strengths and weaknesses of the optical Time-of-Flight sensors. This is followed by the description and features of the automotive dToFs from STMicroelectronics. Chapter 2 then goes through the application requirements and explains the reasoning behind the selected approach to the realization of a physical model of the protected area. The Chapter 3 explains in detail every part of the electrical system that forms the backbone of this thesis. The biggest focus is on the schematic of the control board PCB that operates the sensors and evaluates the presence of foreign objects. Important parts of the routing process are also touched upon, as well as the assembly and soldering. The Chapter 4 delves into the structure of the firmware for the automotive MCU that is part of the control board. It is explained how diffent peripherals are used and controlled and what is necessary to communicate with the board from other devices. The Chapter 5 describes the design and manufacturing of the enclosure that houses all electrical components and creates the protected area. It includes the description of individual parts and instructions for assembly. The Chapter 6 mentions the technology used for the creation of graphical desktop application, but its main purpose is to explain how the application is compiled and how it is used. The last chapter evaluates the product of all these distinct disciplines and presents the results of object detection tests along with other relevant observations.

Chapter 1

Depth sensing technology

The ability to accurately judge distance from an object or even calculate a full depth map of a scene has many useful applications in the automotive industry, smartphones, or cameras. This chapter summarizes different approaches to solving this task using optical sensors.

1.1 Cameras

Output from a typical camera sensor can be successfully utilized to detect objects and assess distances. Even a single camera with the help of a machine learning can not only detect if an object is present in the scene, but can also determine the type of object, recognize specific people and understand complex situations. However, a lot of processing is needed to obtain this information, and in practice a powerful GPU is mandatory. This comes at the cost of higher power consumption and a more complex system in general. When it is only useful to know if something enters the scene and detailed identification is not necessary, the detection of objects by their motion is a much less demanding alternative and can even be implemented into the CMOS sensor itself [1]. This can be great for security cameras, where it is useful to filter out uninteresting footage when the scene remains static.

In either case, critical information will be missing, that is, the distance between the camera and the detected objects. There is only so much that can be extracted from a single image, and although a neural network can estimate distance from cues such as line angles and perspective, this task is inherently not deterministic, and the overall scale of objects is difficult to judge [2]. For example, an image with a toy car could be ambiguous because the neural network expects a car to have a certain scale. The relative positions between the toy car and other objects in the photo can still be guessed, but the absolute distance from the camera would probably be incorrect.

For distance measurement and depth mapping, the better approach is to use *stereoscopy*, which is two cameras that each capture a different point of view, analogous to our eyes that can naturally perceive depth with a similar arrangement. If the pitch of the cameras is known, the distance of an object that can be identified in both images is calculated by a relatively simple triangulation [3]. Problems arise when there are not enough unique features in the scene that can be reliably matched between perspectives, so a projector may be added to project easily distinguishable patterns that will help the algorithm; this is known as *active stereoscopy* [3]. Another technique called *structured light* also employs the projector, but this time only one camera is necessary. The difference is that the pattern is something regular, for example, a grid of dots, and when it gets projected onto an object, the distortion of the pattern is captured by a camera and can be used to determine the distance in each point [4]. In either case, the algorithms that perform the calculations are very demanding, so neither stereoscopy nor structured light are suitable for low-power real-time applications.

1.2 Time-of-Flight

The idea of Time-of-Flight systems is very simple: a signal is sent at the target, and the distance can be determined from the time it takes for it to reflect back to a detector. RADAR and SONAR systems that have been around for a long time work on this principle and utilize radio and sound signals, respectively. However, the focus of this thesis is primarily on the optical relatives and the underlying technology. Passive components like simple cameras rely only on the light that is already in the scene, and therefore are very sensitive to ambient light and cannot work reliably in the dark without an additional illuminator. Bright light sources such as direct sunlight can easily saturate the sensor and, because of the limited dynamic range, objects might be completely lost in the image. In contrast, the optical Time-of-Flight systems are active because they only rely on light that they actively produce. They can work in complete darkness or in strong ambient light, even though there are still some limitations as well.

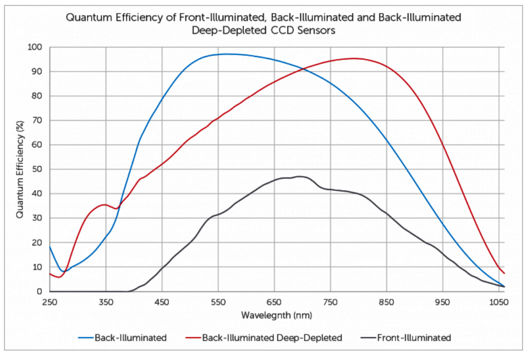
Every optical Time-of-Flight sensor has two components: an emitter and a receiver/photodetector. Today, the emmiter is usually a *vertical-cavity surface-emmiting laser* (VCSEL) with a laser beam that is directed perpendicularly to the surface of a chip. Compared to an older *edge-emmiting laser* (EEL), it is among other things more efficient at low power, the laser beam diverges more slowly, and most importantly, it is easier and cheaper to manufacture, as it can be tested right on the wafer, thus the production is more efficient [5].

The emmiter typically generates infrared light that is invisible to the human eye, but the exact wavelength depends on a number of factors. One of them is *quantum efficiency* (QE), a measurement of the effectiveness with which photons can be converted to electrons; a higher QE means that the device is more sensitive to incoming light. In the visible spectrum, the QE of some cameras can exceed 90 %, but reaching similar figures is generally more difficult as the wavelength increases [6]. The decreased sensitivity that comes with usage of an infrared light is compensated for by the better resistance to ambient light, thanks to a significant absorption of certain infrared wavelengths of solar radiation by the elements in the atmosphere. This relationship between quantum efficiency and solar radiation absorption is shown in Figure 1.1. The three most common VCSEL wavelengths, all of which lie in the dips caused by atmospheric water vapor, are the following:

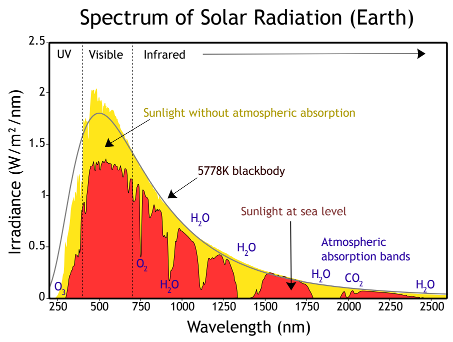
- 850 nm – has a faint red glow and is useful in night vision applications, as the image quality and range are usually better than for higher wavelengths.
- 905 nm – used in automotive industry, has better detection of wet or water-based objects than 940 nm [7].
- 940 nm – common in multiple consumer applications

Higher wavelengths, like 1340 nm, belong to an even more advantageous region of Figure 1.1, where solar radiation is almost completely blocked and thus ambient light is less of a concern. Sensors that operate in this short-wave infrared band (SWIR) exist and can provide lower noise and even better eye safety [9], so the power output and consequently the range can be much higher. However, they can no longer be sillicon-based, and much more expensive materials such as Germanium (Ge) or Indium Gallium Arsenide (InGaAs) need to be used instead [9, 10]. This dramatically increases cost, so this trade-off is not warranted in most common cases.

Figure 1.1 Relationship between a typical QE curve and solar radiation



(a) Quantum efficiency in CCD sensors [6]



(b) Solar radiation absorption [8]

In addition to the choice of wavelength, the illumination scheme is probably the next most important factor to consider on the emmiter side of the sensor. The simplest is the *uniform/flood illumination*, where the laser beam is guided through an optical diffuser to cover the entire FoV [11, 12]. The resolution that can be achieved is determined by the resolution of the detector, but the maximum range is limited by the SNR at each pixel or SPAD and long distance measurement requires higher power output. *Spot illumination* attempts to overcome this issue by using single or multiple focused beams [13, 11, 9]. In areas where the reflected beam hits the receiver, the SNR is much better than in the previous case, but the resolution now depends on the number of laser beams [11]. Alignment is also a concern, as the position of the beam shifts with distance of a target due to a parallax effect (more drastically for closer objects), but if the illuminated pixels are identified correctly, the performance can be better for the same power output compared to flood illumination. The last option is to use *scanning* of the scene by a moving laser beam, which requires a more complicated optics for steering of the beam, but the advantage is that higher resolutions can be achieved with a lesser demand on the overall resolution of the receiver [14].

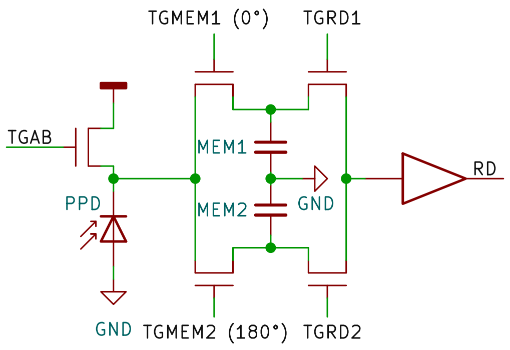
When it comes to receiving reflected infrared photons, several technologies can be used, and they all have strengths and weaknesses, so it ultimately depends on what is most important for the specific application and which type of ToF sensor is used.

1.2.1 Indirect Time-of-Flight

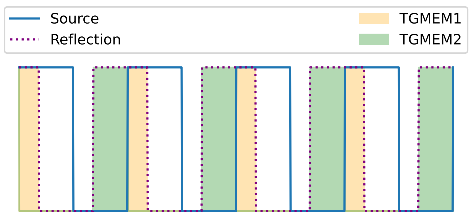
In iToF sensors, the time of a signal's round trip is not measured directly, but instead derived from the phase shift of the obtained signal. For this to work, the emmitter has to send a continuous wave modulated in frequency (FMCW) or amplitude (AMCW), of which the amplitude modulation is more common [9]. Ideally, the signal would be a sine wave, but in practice the square pulses are used because the design of a driver that controls the emmiter is less complex.

The receiver is made of a photodiode, which is a simple component that produces an electrical current when exposed to light. A variant called a *pinned photodiode* (PPD) is the basis of modern CMOS sensors and can also be utilized in ToF systems [15, 16]. The schematic in Figure 1.2 illustrates a single iToF pixel, with TGMEM signals that control the storage of the energy induced by the photodiode. TGMEM switching needs to be synchronized to the same frequency as the emitted signal, so that each TGMEM captures

Figure 1.2 Principle of Indirect Time-of-Flight



(a) Pixel architecture



(b) Capture of the reflected wave with TGMEM switching

signal in the distict part of the period. In this exact case, the period of the signal is divided by 180°– the duty cycle for both TGMEM signals will be 50%, but when TGMEM1 is high, TGMEM2 is low and vice versa. This setup can easily be extended to more capture nodes that divide the period into finer segments. The incoming light is integrated with the TGMEM switching until enough charge is acummulated in memory nodes, and then each node is read. The ratio of charges then determines the phase shift and, consequently, the distance to an object.

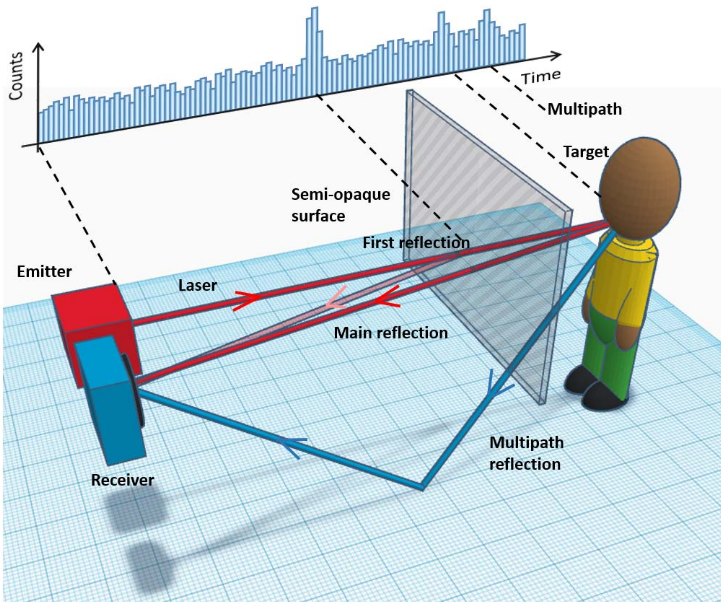
This architecture is relatively simple and easily scalable for higher resolutions, as it is based on the technology present in CMOS imagers [15]. Therefore, iToF sensors are very popular for face recognition, gaming, or gesture control. The downside is that achieving long distance ranging is complicated without sacrificing spatial resolution in the process [9]. The typical range with flood illumination is a few meters; for longer distances, spot illumination is mandatory. The frequency of the emmited pulses is also a key factor, as measurements of objects that are farther than the length of the emmited wave will be ambiguous. Schemes with multiple frequencies are sometimes used to provide wrap-around detection with lower frequencies and high depth resolution with higher frequencies [17]. Multiple frequency modulation can also help reduce the impact of multiple-path interference caused by bouncing of light from objects other than the point of interest [9].

1.2.2 Direct Time-of-Flight

Going back to the original idea of Time-of-Flight, the dToF sensor can precisely determine when the reflected pulse arrived relative to when it was emitted. The problem with sending a single pulse is that it may not reflect back, because the target is either too far or has low reflectance, and a completely unrelated pulse may also trigger incorrect detection. Therefore, in practice, a series of pulses is transmitted over some integration period, and the arrival time of incoming photons is used to build a histogram, where each bin represents the number of photons in a specific time interval [18, 19]. The objects in the scene will correspond to the peaks in the histogram, and the position of the peak can be used to calculate the distance according to a simple Formula 1.1, where c stands for the speed of light.

$$d = \frac{t_{peak}}{2} \times c \tag{1.1}$$

Figure 1.3 Illustration of a relationship between histogram and the actual scene [22]



The photodetector in dToF is typically nowadays a *single-photon avalanche diode* (SPAD), which is created by biasing a diode beyond its breakdown voltage. In this state, a single photon can cause an avalanche reaction in the SPAD that increases the current flow through the diode. Because a quenching resistor is connected in series with the SPAD, the current will also increase the voltage across the resistor, which will consequently result in a drop in voltage across the SPAD [20]. With lower voltage, the avalanche event is slowly suppressed and the SPAD is brought back to its default state. *The duration of this operation, during which the SPAD is unable to detect photons, is called dead-time, and is typically a few nanoseconds long* [18]. The current spike is easily detected and converted by a *time-to-digital converter* (TDC) to a digital pulse that can be processed and counted towards a relevant histogram bin. Although the high sensitivity, fast reaction time, and low timing jitter[1] of SPADs are particularly advantageous for dToF [22], they can be utilized in iToF sensors as well [23].

The key metric of SPAD is the photon detection probability (PDP), which is an SPAD equivalent for quantum efficiency, i.e. the sensitivy of the sensor [18, 24]. As the name implies, it would be incorrect to assume that a single photon is guaranteed to cause an avalanche event in SPAD, but there is still a tight correlation between the two. Avalanche events that are not caused by photons might also occur, and this is usually referred to as a *dark count rate* (DCR) [21]. DCR along with ambient light form a noise ground floor that is the limiting factor to a maximum range of the sensor because at some point the returning signal is too weak to stand out. On the other hand, it is also possible to saturate the sensor with a signal that is too strong and that will limit the minimum detectable range.

The captured histogram must be post-processed with suitable algorithms that correctly identify peaks belonging to target objects. The scenario shown in Figure 1.3 highlights how some of the objects affect the shape of the histogram. Unwanted peaks might occur due to multipath reflection or crosstalk between emitter and receiver, but if the post-processing is successful and the cause of every peak is correctly identified, challenging tasks such as identifying multiple sufficiently spaced apart objects or objects behind semi-opaque surface are more straightforward compared to iToF [22].

[1] *the uncertainty between actual and measured photon arrival time* [21]

The downside of using SPADs is that a readout process is more challenging than that of a photodiode because TDCs are more complex, require precise timing, and have relatively high power consumption. Current SPAD sensors also have a low fill factor[2] of at most 5% in 2D grid arrangements without microlenses, making scalability to high-resolution pixel arrays challenging [25, 26, 27]. Although this technology is continuously improving due to advances in 3D stacking processes, the only viable way to achieve high spatial resolution with SPAD dToF is to use scanning instead of flash or spot illumination, which increases cost and complexity. Therefore, dToF is still mostly used in low-resolution/long-range applications. Almost everyone is familiar with the common usage in most smartphones, where the dToF sensor detects if the phone is close to an ear during calls so that the screen can be turned off and accidental interactions can be prevented.

1.3 Automotive dToF from STMicroelectronics

The physical sensor used throughout this thesis was a prerelease engineering sample provided directly by STMicroelectronics. As such, it was not available for purchase at the time and the only relevant public source of information was the VL53L4 dToF product page. This is because the sample shares most of the important characteristics with the VL53L4, except that it is designed to obtain automotive certifications. The VL53L4 exists in two variants: VL53L4CD and VL53L4CX. The CD variant is primarily intended for proximity sensing and its maximum range is limited to 1300 mm, while LX is capable of distinguishing even multiple objects spanning up to 5 meters. Each variant has its own specific software driver, but the overall control scheme is very similar. They also come in the exact same package, and the PCB layout recommendations are the same as well. The internal architecture of the automotive dToF is likely to be closer to VL53L4CD, because some features specific to the LX could not be accessed. Properties listed in the rest of this section are common to all three variants of dToF sensors (CD, LX, and automotive), unless explicitly stated otherwise.

Figure 1.4 VL53L4CX package [28]

The emitter is the VCSEL with wavelength 940 nm, and the receiver is an SPAD array with 18°wide FoV. The sensor contains a 32-bit embedded MCU that handles the readout of SPADs and the consequent post-processing and distance estimation. I^{2}C interface is used to send these data to a host system. I^{2}C is a very common multi-master bus that only needs two wires – *synchronous data* (SDA) and *synchronous clock* (SCL) – connected with pull-up resistors to voltage representing logic high. The master node, usually configured as open-drain output, is then able to send information by pulling these two lines low according to protocol, so that the slave node may read the data frame. Every node on the bus must have its unique address (typically specified by the manufacturer), which is sent with every data frame. The default address is `0x52`, but a scenario with multiple sensors on the bus is also supported, so there must be a way for the host system to modify this address.

That is where the XSHUT pin comes in. It can be controlled by the host, and pulling it low disables the sensor firmware so that it will not respond to I^{2}C requests. When the XSHUT of each sensor is connected to the host MCU, all sensors can be disabled except one, and that will mean that the I^{2}C command that changes the address will

[2]measure of how much of a sensor die is dedicated to a light sensing

Figure 1.5 VL53L4 reference schematic [29]

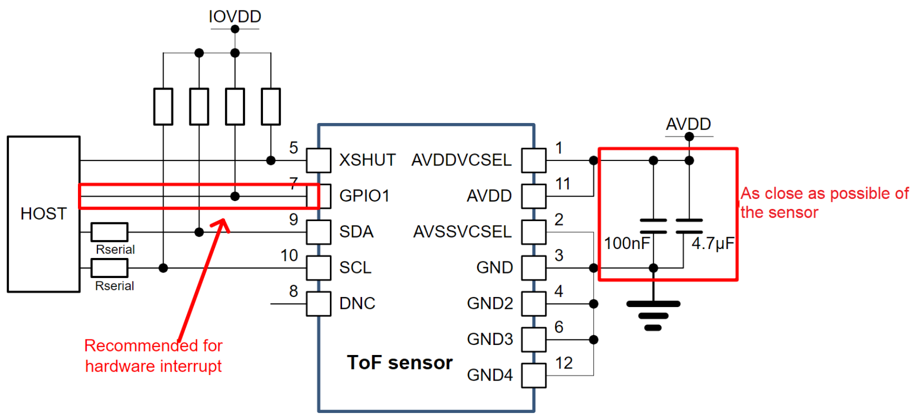
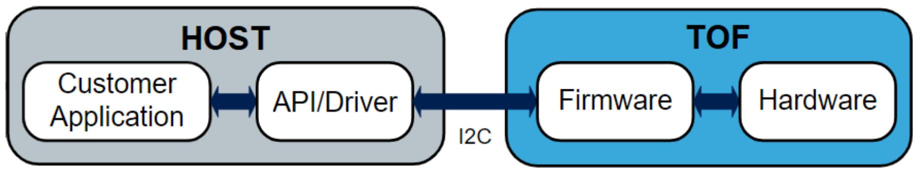


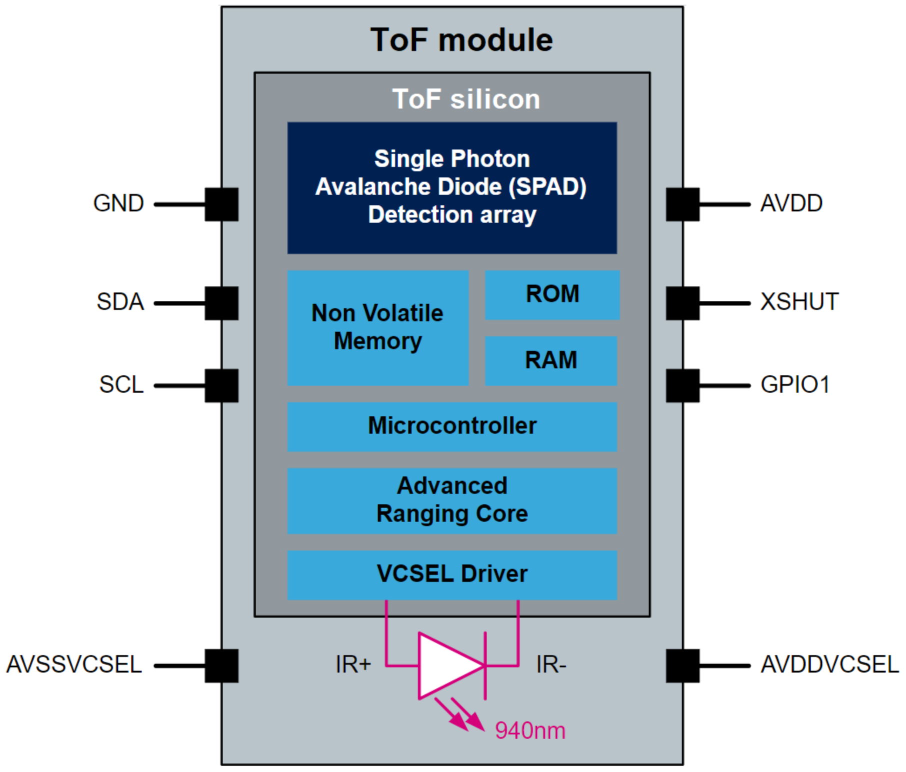
Figure 1.6 VL53L4 system functional description [28]



only be applied to that particular sensor. This process is repeated until every sensor has its unique address. If this feature is not needed, the XSHUT may simply be hardwired through a pull-up resistor to the IOVDD, so that the firmware always boots automatically when the power is present. However, the XSHUT should not be left floating, to avoid leakage current.

With every sensor addressable, the firmware initialization procedure may be started. An optional calibration and configuration may be performed, and then the main control loop is started with a function that activates the VCSEL and starts the ranging process. When the result is ready, the GPIO1 pin is pulled low and can be used as an interrupt signal for the host MCU. After the interrupt is detected, the host MCU reads the results and clears the interrupt, which automatically starts the next range cycle. If the interrupt cannot be used, it is possible to obtain the result by polling. The functional equivalents of all these commands are implemented in both the CD and LX software drivers. The drivers, especially the CD variant, are also not very demanding, so even a low performance system can easily handle the control procedure. Because the simpler CD sensor only measures the distance to a single target, it actually has some features that can further reduce the load on the host system. For example, the interrupt can be raised only if the measured distance is above or below some threshold set by the user. There is also a possibility of specifying a range with two thresholds and waiting for a measurement that lies inside or outside this range.

Figure 1.7 VL53L4CX block diagram [28]



Chapter 2

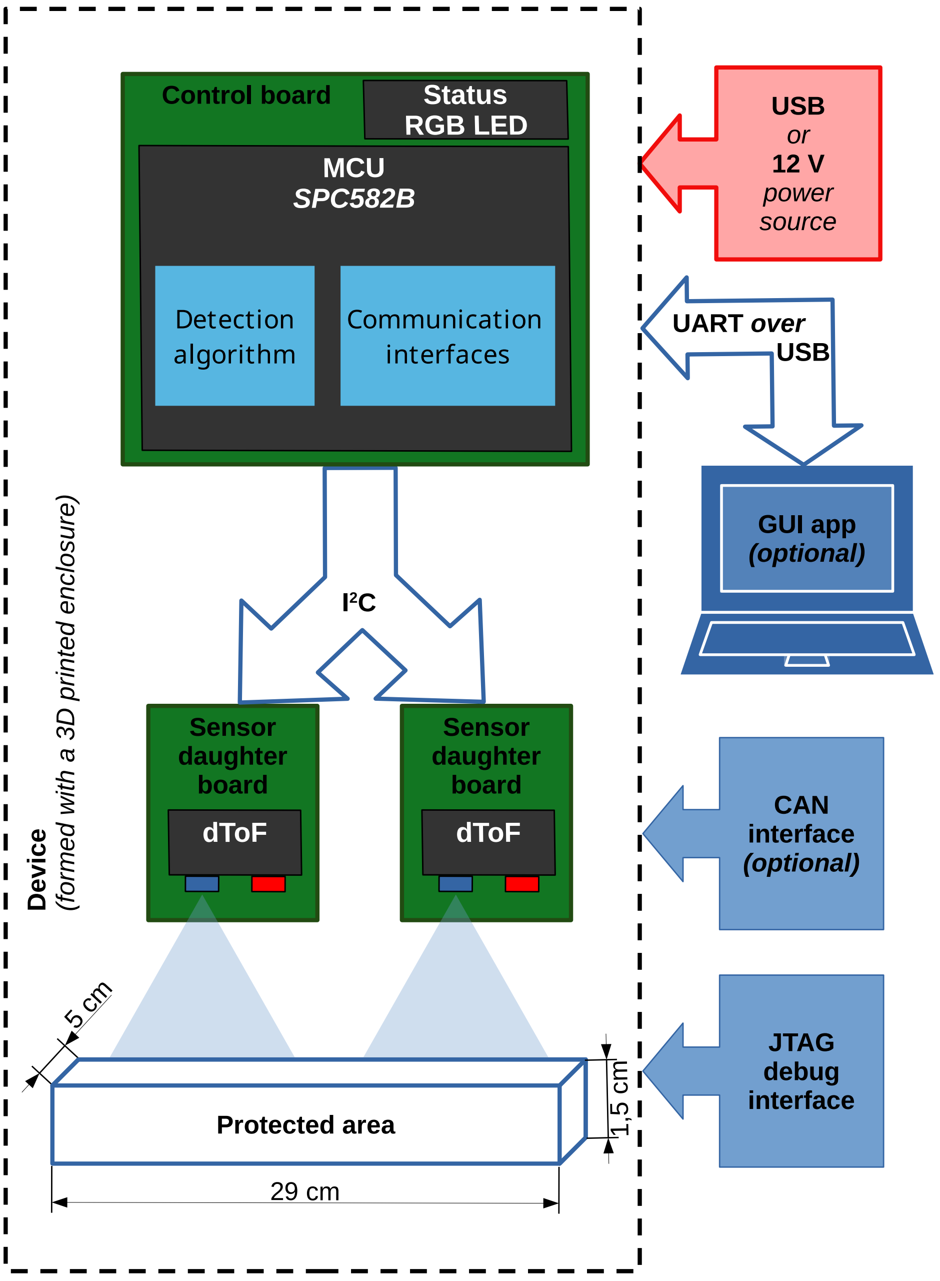
Application requirements

The automotive application this thesis deals with requires detecting an object's presence in a predefined rectangular area with dimensions $29 \times 5 \times 1.5$ cm. It is not important to distinguish what type of object is present only if there is something in the guarded volume that would be considered an obstacle. The exact specifications of the area and detection model come from a large European carmaker that wants to implement a similar system in its products to solve an in-cabin task that cannot be further specified. Currently available state-of-the-art solution is based on the beam break principle. There is an array of light sources on one side of the area and light detectors on the other side. If an object enters the area, the link between the transmitter and receiver is broken and the object is detected. However, this solution is limited by the resolution of an array, and some objects can still enter the scene without being detected. It is also a complex and very expensive system, so the car manufacturer wanted to explore simpler and cheaper alternatives.

The idea is to use one or multiple automotive direct Time-of-Flight sensors, which are based on the commonly available VL53L4 sensor that functions basically the same way, with the difference being that it does not have automotive certifications. Currently, there is no publicly available information about the price of an upcoming automotive variant, as it is still in development, but it should be in single units of US $, so the cost savings should be significant. Naturally, the low price of the sensing component is meaningless if it becomes overshadowed by the price of the rest of the system. Fortunately, the MCU that will read the sensor data does not have to be excessively powerful and expensive, as most of the processing is already done by the sensor itself. The measurement results are gathered via an I^2C bus, and the MCU's responsibility is only to implement a detection algorithm, read the sensor data, and transmit the information about the presence of the object to the user.

Although detection with dToF sensors would satisfy complexity and price requirements, it is still unclear whether this technology can perform well in this use case. The beam break solution covers only the guarded area, and the probability of false positives is low. Similarly, cameras can recognize the boundaries of the area and perform the detection only in that portion of the image. But when the dToF sensor is used, there is no way to truly limit the detection to only a certain portion of the scene. Each sensor naturally has its field of view that should be taken into account when the sensor is positioned to cover most of the protected area, but it is not possible to isolate detection to an arbitrarily shaped volume within this FoV because the light refracts and returns from different directions. Objects within the FoV that are not inside the area can cause a reading that would be indistinguishable from the actual obstruction, because the only information available to a sensor is the intensity of photons and the time of their arrival, not their direction.

Figure 2.1 Block diagram of the setup for object detection in protected area using dToF sensors



This means that a development setup that can be used to easily evaluate the optimal positioning of the sensors and realize the detection algorithm is needed. To match the conditions specified by the carmaker, the surface of the protected area has to be made out of glass, and the surrounding 1.5 mm tall frame out of a black plastic. The position of the sensors within the frame must be adjustable in all axes and firmly fixed once it is set. Because the frame should only contain an opening for a sensor and be otherwise uninterrupted, moving mechanisms are not viable, as they would require a gap in the frame larger than just the sensor, and it would also be harder to keep such mechanism in a set position. Therefore, adjustments will be made by creating different versions of the sensor housings.

The block diagram of the application can be seen in Figure 2.1. The objective is to create a standalone and easily transportable device from the protected area model. As was already mentioned, sensors must be embedded in the area frame, but the device will also integrate a board with an automotive microcontroller and all the cables needed for connecting it to the sensors. The microcontroller will be programmed to process the measured data and determine if a foreign object entered the area. The detection algorithm will work automatically after boot and no user action will be needed to obtain the detection status, which will be clearly indicated visually with a colorful LED. Finally, the device will expose several connectors to access different communication interfaces and to connect a suitable power source. The communication interfaces will optionally be used to transmit the sensor data to a desktop application that will visualize them in a graphical interface. Creating a simple way to capture and analyze measured data is going to be very helpful for future development and for anyone who wants to implement a similar object detection device.

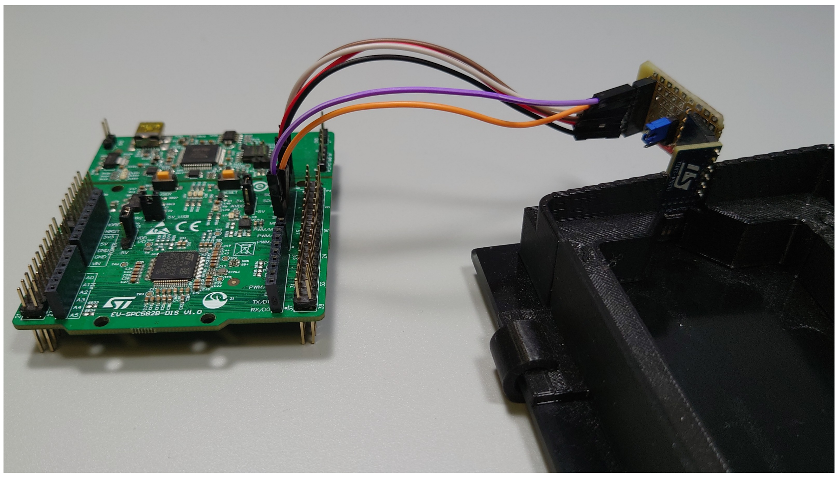
To evaluate the performance of the detection algorithm, a minimal defined set of objects outlined in the assignment should be considered: pen, coin, and credit card. This list will be extended with three extra items; a parking ticket to demonstrate the system's ability to detect very thin objects and normatively sized black and white cubes to test the influence of the object's reflectivity.

2.1 Preliminary testing

The detection algorithm must be implemented on the automotive PowerPC MCU from the SPC58 lineup, so it was important to get familiar with the development process for this platform. The commercially available SPC582B-DIS development board proved to be a good starting point. It has the same interface as the very popular STM32 NUCLEO development boards, so it worked with the P-NUCLEO-53L4A1 shield, which has one VL53L4CD sensor onboard and has the ability to also connect with two more sensors mounted on sattelite boards. The NUCLEO shield was soon left out, and one of the satellite boards was directly connected to headers on the SPC582B-DIS board, because it was easier to test it in different positions and it is much smaller than the shield. The development board is supported in the SPC5Studio IDE, so the startup configuration was not a concern and it was only necessary to create working UART communication and integrate the VL53L4CD ULD driver[30], which exposes a simple interface to take care of the low level I^{2}C commands. The driver is platform-independent and only requires the definition of functions that read and write data on I^{2}C bus. Attention must be paid to how these functions are implemented, especially those that read or write multiple bytes, because the PowerPC architecture is big endian while the VL53L4 expects data as little endian.

With a working code that reads the current distance and intensity of the returned light and forwards it to a serial device connected to one of the UART peripherals of the MCU, some basic tests could be performed. A simple plastic frame with slot for the sattelite

Figure 2.2 Preliminary testing with SPC582B-DIS Discovery board and VL53L4 Satellite board



board was 3D printed and mounted on a sheet of glass, and the measured distance was evaluated under different conditions. First, it was clear that a single sensor would not be able to provide sufficient coverage and that at least one more sensor on the opposing side would have to be present. Another observation was that the detection of small objects seemed problematic, especially those with low reflectivity. Lastly, according to expectations, objects that are in the sensors FoV but are not inside the protected area could cause the same reading as an actual object. Unfortunately, in each case, even slight changes in the position of the sensor could significantly alter the measured values and detection capability, so these results could not be properly quantified without a proper housing that would prevent the satellite board from wobbling in its slot. However, even without the specialized housing, the main purpose of this rudimentary setup was fulfilled: to ensure that the sensor can work with the automotive MCU and that a better understanding of what is needed from the area enclosure is gained.

2.2 Hardware specifications

The VL53L4 satellite board connected with jumper cables to the SPC582B-DIS worked well, but was definitely not suitable for use in the final product. One issue would be that the the satellite board is not intended for automotive use, but since the package of the VL53L4 is the same as for the automotive dToF, the original sensor could simply be desoldered from the PCB and replaced with an automotive variant. The real problem is that using a generic development board for a specific use case becomes complicated as more components are added, and connecting everything by individual jumper cables that do not possess any form of locking mechanism is also not very reliable and scalable. The finished device should be able to handle a reasonable amount of physical stress from user handling, transportation, and other factors. Unfortunately, even slight movement of cables or touching a contact could interfere with or completely interrupt communication between the sensor and the processor. Having a lot of individual PCBs connected with a lot of cables would also not look good and would increase the space requirements dramatically. Clearly, a more integrated system is necessary, and designing a custom PCB is the most important step in achieving that.

During the planning stage, many different features and requirements for the final product were considered. The board must be able to communicate with the PC in some way and the simplest solution is to use a USB port with a UART bridge. However, programming of the onboard MCU will be handled by a separate 10-pin JTAG connector. Although it would be preferable to have just one connector for everything and the SPC58 MCUs technically can have their firmware binary flashed over UART or even CAN, the advantage of JTAG is that it can also be used for debugging [31]. Choosing a solution like FTDI's FT233HPQ or FT2232HL, which can provide both JTAG and UART bridge over single USB cable, was not practical because of a significantly higher price and lack of availability at the time.

Additionally, the 9-pin D-Sub connector will be present for interfacing with a CAN bus. Although it is technically not necessary for this thesis, it is safe to assume that in the future the device might be used in conjunction with some other automotive device or tested inside a car. For the same reason, the board should have the option to be powered from 12 V with a barrel jack connector. USB will also be able to provide power, so the device can function without an external power adapter (the power source should be selectable using a jumper, switch, or other means). The mandatory bright LED that will indicate the detection status will be accompanied by smaller LEDs and buttons to allow for simple controls of the board without any master device.

Automotive grade components will be prioritized, unless an industrial part provides better performance, relevant features, or significant cost savings. These compromises would, of course, not be possible in a finished product, but the industrial components that are involved would not appear in a real car anyway. This includes features such as a USB-C connector and UART transciever or even addressable RGB LEDs because the object detection status would probably be only relayed over the CAN bus to a gateway and then to some dashboard controller, as it is unlikely that the board itself would handle the signalization to the user.

Chapter 3

Hardware design

This chapter follows the entire process of creating a highly integrated PCB that can reliably control multiple Time-of-Flight sensors over I^2C bus. First, the electrical schematic and the reasoning behind the selection of certain components are described. This is followed by an explanation of the critical aspects of PCB layout, component placement, and routing. The closing sections are dedicated to the manufacturing process and assembly of the board.

Altium Designer was chosen as the PCB design software mainly because it is a de facto standard in the industry and, therefore, makes potential future collaborations easier. Although it comes with many great features, one big disadvantage is that it is not free and open source, and viewing design files without a valid license might be difficult. For that reason, detailed PDF exports and also Gerber files used for fabrication are included in the attachments. Importing to free alternatives like KiCAD or Eagle is also possible and generally good enough for reference purposes.

It is important to inform the reader that the manufactured board shown in some of the photographs and used throughout this thesis is version 1.0 and, as such, unfortunately contains some imperfections. Although they were not serious to the point that the board would have to be completely discarded, it would be confusing to show here 1.0 schematics that include these mistakes. Therefore, all schematic and board-view snippets come from version 1.1. Important differences will be discussed.

3.1 Electrical schematic and component selection

The schematic is divided into 3 sheets, each dedicated to some logical part of the board: the MCU (with some peripherals), power distribution and dToF sensor interface. This section follows the same structure.

3.1.1 Power inputs

Designing a reliable way to power the board is essential, especially in an automotive environment. That is why the primary 12 V power input is converted to 5 V by an automotive grade DC-DC buck converter. The exact model is A7985A and it is manufactured by STMicroelectronics. It can provide up to 2 A of current, which gives us a lot of headroom. The same can be said for the input voltage, which can range anywhere from 4.5 to 38 V.

This means that even 24 V input can technically be accommodated, although the efficiency of output would not be optimal in such a case. The reason is that, in general, the rest of the components around the switching converter should be adjusted for most

Figure 3.1 A7985A schematic

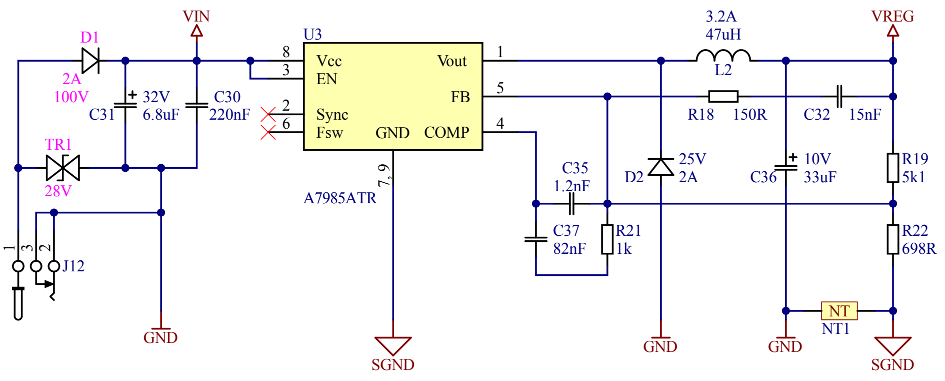
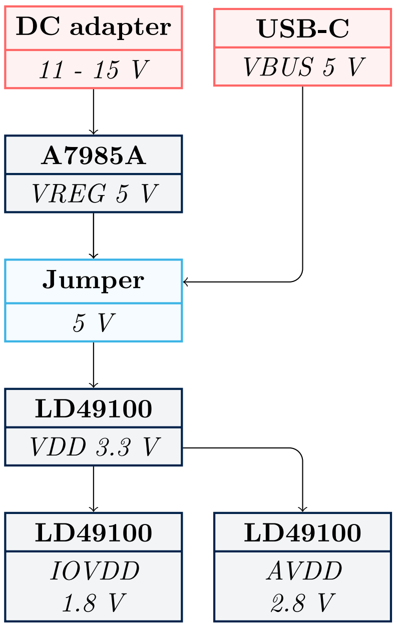


Table 3.1 List of A7985A components

Ref	Type	Reference component	Selected component
U3	IC	*A7985A* - STMicroelectronics	
C31	Capacitor	*20TQC8R2M* 8.2 uF / 20 V / 20 % Panasonic	*32SEQP6R8M* 6.8 uF / 32 V / 20 % Panasonic
C30	Capacitor	220 nF	
C36	Capacitor	*TPSB3360060250* 33 uF / 6.3 V / 20 % AVX	*10SVP33M* 33 uF / 10 V / 20 % Panasonic
L2	Inductor	*MSS1278H-473KED* - 4.7 uH / 4.4 A - Coilcraft	
D2	Diode	*STPS120MF* - 1 A / 20 V	*STPS2L25* - 2 A / 25 V
C37	Capacitor	68 nF	82 nF
R21	Resistor	1.1 kΩ	1 kΩ
C35	Capacitor	1.2 nF	
C32	Capacitor	15 nF	
R18	Resistor	270 Ω	150 Ω
R19	Resistor	5.1 kΩ / 1 %	
R22	Resistor	698 Ω / 1 %	

common operating conditions. If these conditions are defined too broadly, unnecessary spending or suboptimal performance are likely to occur. Finding the right balance is not easy, so thankfully, STMicroelectronics provides an online tool called eDesignSuite that can produce an optimized reference schematic for a multitude of power management components.

■ **Figure 3.2** Power component hiearchy diagram



In this tool, the input voltage was chosen between 11 and 15 V, so even an unregulated car battery voltage that can vary depending on the remaining power is taken into account. On the output side, the original intention was to select a 3.3 V output voltage that would be directly supplied to the MCU and other components, but ultimately a 5 V voltage was chosen because it was needed to power a CAN bus transceiver. Finally, the output current was set to 1 A, mainly because it is a maximum value that the LD49100 regulator can provide for the derived power rails.

The generated schematic was further tweaked and the final version is shown in Figure 3.1. One key difference is the addition of reverse polarity and overvoltage protection, consisting of the shottky diode `D1` and the TVS diode `TR1`, respectively (both highlighted in purple). The overall structure of the schematic is the same; only some of the components were swapped for alternatives.

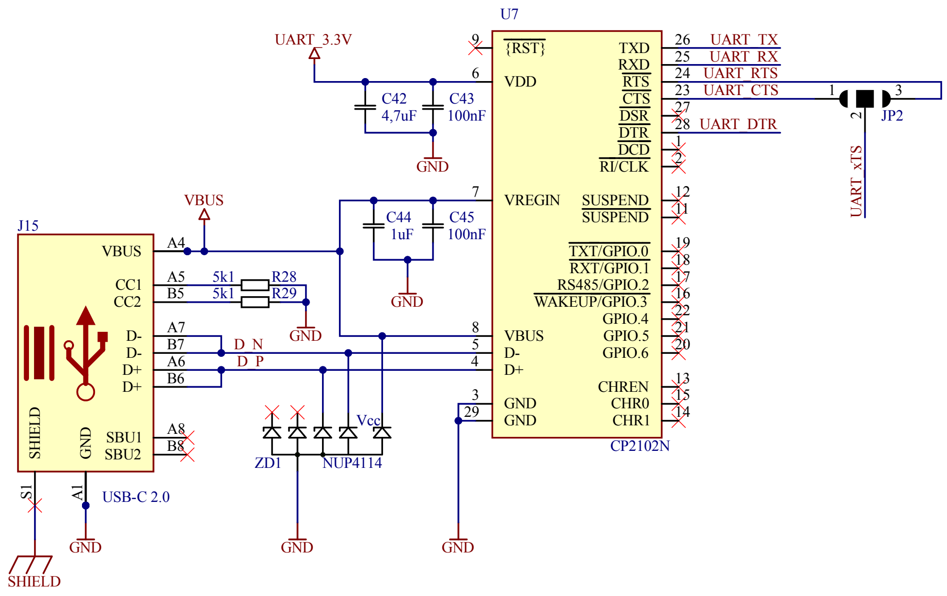
A detailed comparison of the reference and selected components is listed in Table 3.1. The input and output capacitors had to be replaced because the suggested parts were not available at the time. The diode `D2` was unfortunately ordered before the decision was made to use the 5 V output, so the recommended part for the 3.3 V version was kept. Changes in the compensation net were made to obtain nicer and more typical values, but only the resistor `R21` was selected manually – the rest was calculated automatically by the online tool to reflect this change. All of these modifications had a marginal effect on the expected efficiency, which decreased only by 0.7% from 89.5% to 88.8%.

Another 5 V power source, selectable with a jumper, is the USB-C connector used in conjunction with the CP2102 UART bridge[1]. This part of the schematic (shown in Figure 3.3) is based on the typical connection diagram in the CP2102 datasheet [32], which provides a minimal working example, as well as several optional variations, of which two were implemented. The first is the addition of the 4.7 uF capacitor `C42`, which is required if the internal 3.3 V regulator is used to power other components. This is intended to help in a situation where the USB portion of the board is used standalone in some other application.

The second optional addon is the `ZD1` ESD diode connected to the VBUS and data lines. It protects the board to some extent against short voltage spikes, but a PTC fuse could also have been added to provide overcurrent protection. This omission should be rectified in the next version of the schematic. Unfortunately, another more serious error was made: an incorrect connection of the RX and TX lines. Figure 3.3 already shows the correct arrangement, but the manufactured board had to be fixed by crossing the RX/TX lines with cables.

[1]In the schematic, UART bridge is labeled as CP2102N instead of CP2102. This is because CP2102N is a designation for a newer version of this UART bridge and is fully compatible with the legacy CP2102

Figure 3.3 USB-C and UART bridge schematic



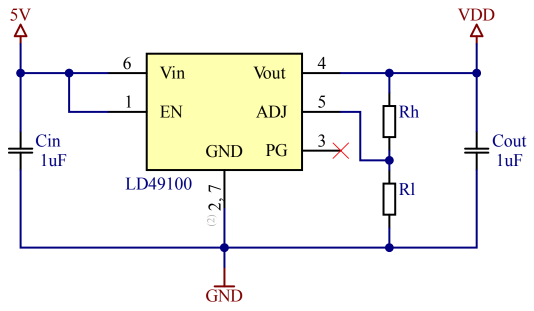
USB-C receptacle has a reduced pin count, which means that it does not offer USB 3.0 functionality, but it is easier to solder than the full 24-pin version. Since CP2102 is a USB 2.0 device that is typically used with a microUSB connector, the lack of high-speed pins is not a problem. Replacing microUSB with USB-C is not difficult; the only thing that requires extra attention are the CC1 and CC2 pins, which advertise to an upstream port what voltage and current the device requires. Typically, these pins are pulled down with 5.1 kΩ on USB 2.0 devices, and that is also the case here.

The last component in this section is the automotive grade LD49100 low dropout linear regulator that can lower the input voltage with fewer components and less output noise than switching regulators. However, its efficiency depends on the difference between input and output voltages – a higher difference means that more power is lost as heat. Therefore, A7985A is first used to reduce 12 V to 5 V as effectively as possible, so that the input of LDOs is not too distant from their output. In total, three LDOs are used. One converts 5 V to 3.3 V, which then gets converted by remaining two to 1.8 and 2.8 V to power dToF and its communication. The full power component is illustrated in Figure 3.2.

An adjustable version of LD49100 was chosen for this board, since one part can be used multiple times, reducing the complexity of the assembly process and making adjustments to output values possible. The output voltage is determined by a resistor divider whose values are calculated as $V_{out} = V_{adj}(1 + R_h/R_l)$ with $V_{adj} = 0.8$ V [33]. Fixed voltage variants for 1.8 and 3.3 V also exist and can be used as a drop-in replacement. If this is the case, the reference schematic for the adjustable version in Figure 3.4 must be changed so that R_h is shorted with 0 Ω resistor and R_l is left unpopulated [33].

LD49100 also has a very useful enable feature that allows the MCU to completely turn off its output. Both IOVDD and AVDD are controlled in this way, so the connected dToF sensors can be truly reset if some serious error is detected or simply to save power. The enable pin of the VDD is connected to 5 V, so it is permanently on, but it could also have been used as a simple manual power switch instead. This option will be considered in future iterations because currently it is not possible to shutdown the system without unplugging the cable or disconnecting the jumper. Grounding of the enable pin would not stress the hypothetical switch nearly as much as breaking the current path, so a cheaper part with a lower current rating could be used.

Figure 3.4 Adjustable LD49100 schematic



3.1.2 Sensors

The board must be able to communicate with up to four dToF sensors connected by a cable, and thus a suitable connector is necessary. Such a connector needs to provide a secure connection, but also be reasonably compact and easy to use. Ultimately, the SignalBee DF51K 8 pin double row connector from Hirose was selected as a great compromise between size, quality, and price. It boasts a strong and clicky locking mechanism that has been proved to be capable of withstanding accidental disconnections. Contacts are organized in a standard 2 mm grid layout, so finding an alternative should not be an issue in cases where this exact connector cannot be used. 2 mm spacing might be less common than 2.54 mm, but it provides a noticable space saving without making the socket contacts too small and too hard to crimp. The only downside is that this connector does not have automotive certification.

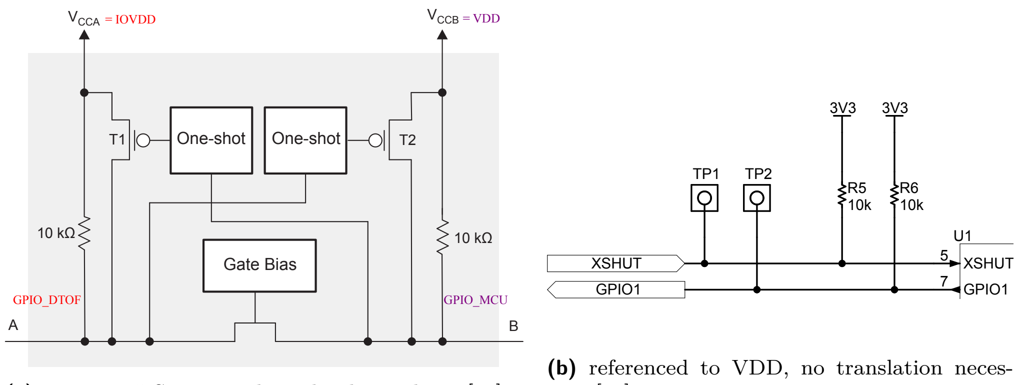
However, connecting the sensors to the board is only a small part of the problem. The bigger part was to ensure that the MCU and the sensors could communicate with each other, because the sensors are powered by a voltage different from that of the MCU, as stated in Section 3.1.1. The goal was to prevent any problems with heat or power consumption because the 3.3 V of the VDD power rail is right at the upper limit of the specified operating conditions, so both IOVDD and AVDD were lowered to their typical datasheet values[29]. As a consequence, some form of voltage-level translation is necessary.

Two different components were used to facilitate translation between VDD and IOVDD. The first is the PCA9306 from NXP, which translates the I^2C bus signals. It can handle speeds up to 400 kHz, which is below the maximum of 1 MHz that the dToF sensor can support in *fast mode plus* and therefore can be a limiting factor in some cases. Fortunately, in this application, I^2C speed is not a bottleneck, so a regular fast mode is good enough. The pull-up resistors for SDA and SCL are externally connected on each side and their values were chosen based on the *VL53L4CD NUCLEO expansion board* schematic [34].

The rest of the sensor's GPIO signals are translated by three TXS0104E 4-bit bidirectional voltage-level translators from Texas Instruments, which exist in both industrial and automotive versions. Their main advantage is that the 10 kΩ pull-up resistors are already integrated at each port, so the board space occupied by these translators is compensated for by the reduced number of resistors (see Figure 3.5). All TXS0104E translators have their ports permanently enabled according to the datasheet [35]. Additional automotive-grade ESD protection is also present on the sensor side to further improve the integrity of the signals and prevent any damage to the system.

The sensor interface can theoretically be simplified in future versions because only six wires are strictly necessary to communicate with the dToF sensor. IOVDD can be left out because all pull-ups are part of the main board and the sensor does not need it as a reference nor a power source. It was included for potential future versions of the daughter board that might have more than just a sensor itself, or a completely different sensor altogether.

Figure 3.5 Connection of dToF GPIO pins



(a) using a TXS0104E voltage-level translator [35]

(b) referenced to VDD, no translation necessary [34]

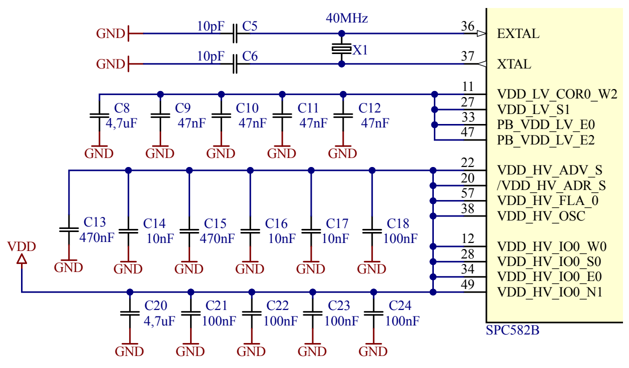
The same applies to IO0-IO3 signals, which could be used to control additional circuits, for example, an LED indicator that will highlight which sensor is currently ranging or a temperature sensor. If these redundancies were left out, 6 pin connectors and only two TXS0104E translators would be required.

3.1.3 MCU

The most vital component is, without question, the SPC582B automotive MCU. It has a single 32-bit PowerPC core, 1 MB of flash for code, 64 kB for data, and multiple peripherals including UART, CAN, I^{2}C, and SPI[31]. It can be powered from 3.3 or 5 V and is generally available in three packages:

- eTQFP64 with 64 pins
- eTQFP100 with 100 pins
- QFN48 with 48 pins

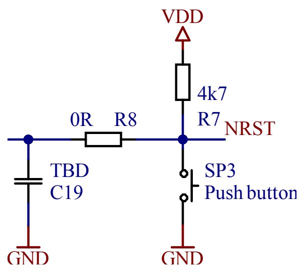
Figure 3.6 Clock crystal and debouncing capacitors of the SPC582B



Since the board already contains several components in QFN packages, it would be preferable to use a compact and estetically pleasing QFN48. Regrettably, it's pin count is too low, so a larger eTQFP64 had to be selected. The upside is that the eTQFP64 can be hand-soldered, and it is easier to find soldering errors and repair them.

Not much is needed to get the MCU up and running, namely a clock crystal, a JTAG connector, and a network of decoupling capacitors. These components were selected based on the recommendations from the ST's automotive team, whose preffered schematic contains fewer capacitors than the official minimal working example available online [36], and therefore takes up less board space and is easier to route. In addition to capacitors, the essential components include a 36 MHz crystal and a 10 pin JTAG connector with 50 mil pitch, which is more compact than a full-size 14 pin 100 mil version in other automotive boards, so an adapter might be necessary to connect to a debugger.

Figure 3.7 Button with optional debouncing

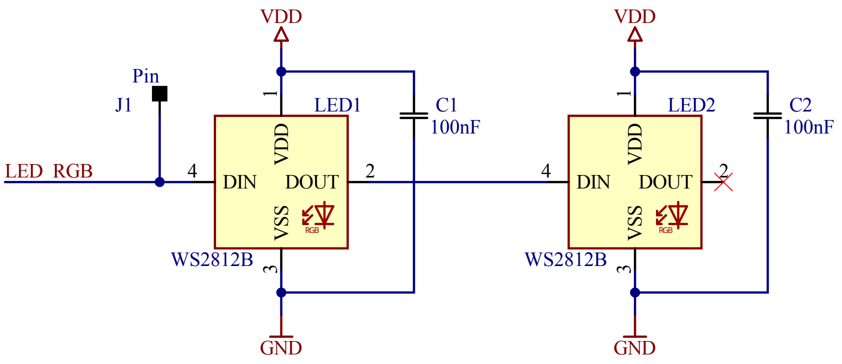


Three buttons are present on the board: a reset button and two general purpose buttons that should allow for some basic controls in cases when the board is working standalone. All of the buttons are active low and can be debounced in hardware by changing the values of capacitor and series resistor. For a reset button, that would be `C19` and `R8` shown in Figure 3.7, but all buttons are wired in the same way. By default, no debouncing is used, so a series resistor is shorted and a capacitor is not populated. Unfortunately, both general purpose buttons were mistakenly connected to GPIO pins that do not support external interrupts and must be polled by the firmware to detect a press.

For visual indications, two simple 0805 LEDs are available, but they are only useful for signaling error states because they are small and when the board is mounted inside a 3D printed enclosure (refer to Chapter 5), they will not be visible unless the whole device is turned upside down. As a result, a much more powerful and versatile WS2182B RGB addressable LED will indicate whether the object is present inside a scene or not. The schematic shows four of these LEDs, but the intention is to provide an option to mount two of them on either front or back of the PCB, depending on how the enclosure is to be illuminated. Technically, both sides can be occupied, but they cannot be controlled separately.

This is because each side is organized as a daisy chain pair (see Figure 3.8) and both pairs would be controlled by the same signal coming from pin 48 of the MCU. This pin is internally connected to an SDO signal from the SPI peripheral, which can be used to generate a series of pulses according to the required protocol. Other SPI signals are not used. Daisy chaining feature of WS2812B allows for a simpler schematic that scales well, unlike a conventional RGB LED where each color channel has to be controlled separately by a PWM signal for each LED. So, two conventional LEDs would take up 6 MCU pins, while WS2812B needs just one no matter the number of LEDs. Limiting factors are only an available power and a refresh rate of the chain.

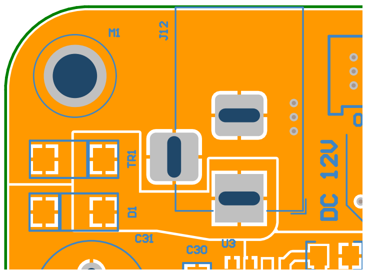
Figure 3.8 Pair of daisy-chained WS2812B RGB LEDs



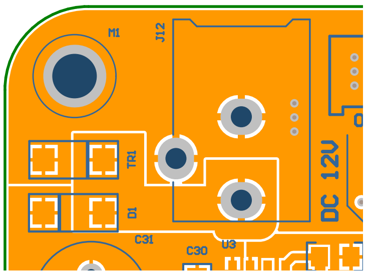
3.2 PCB design

The form factor of the board had to be carefully considered so that the design of an accompanying enclosure was not too complicated. Therefore, the board has a rectangular shape with three M3 mounting holes. The dimensions of the rectangle, as well as the position and spacing of the mounting holes and RGB LEDs, are of nice values rounded

Figure 3.9 Barrel jack footprint comparison



(a) Flat pins



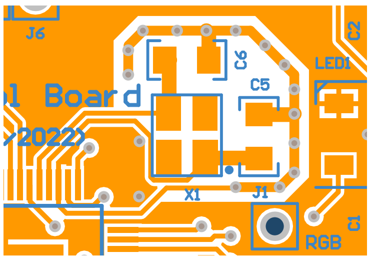
(b) Round pins

to the nearest millimeter to make things easier, and all important connectors are placed around the edges so that they can be exposed to the user through cutouts in the walls of an enclosure. The board has four layers organized in a typical fashion, with the inner ground plane, the inner power layer, and the outer layers mostly used to route the data signals.

At first sight, the most noticeable feature is the USB interface, which is separated from the rest of the board by large grooves. Therefore, the complete removal of the USB portion of the board is an option that can be useful in certain cases. Most likely, this would occur during evaluation, when the board undergoes different measurements and even stress tests that put the system outside of normal operating conditions. So, either intentionally or accidentally, things like short circuits can happen and put anything connected to a USB port in danger. The preferable solution is to use optically isolated or a wireless, e.g. Bluetooth, UART interface that would not expose the expensive equipment to such failures. Although integration of either feature would be possible, it would require a considerable amount of board space and increase cost.

Even after the USB interface is removed, it can function standalone as any other USB to UART converter. In this mode of operation, the `P1` header should be mounted, as it serves as the main interface with the same order of pins as in typical Arduino compatible UART programmer. To use the power coming from the USB interface, the `P2` header with appropriate jumper must be present to select between 3.3 and 5 V; otherwise, the VDD pin is disconnected. It is important to pay attention when using the version 1.0 of the board, because it has `P2` labels swapped, so moving jumper to a position labeled 5 V puts 3.3 V on the output VDD pin and vice versa. Auto-reset during firmware flashing, feature common in some generic development boards, is also supported thanks to the exposed DTR and RTS pins. CTS may also be used instead of RTS and can be selected with a solder bridge on the bottom side.

Figure 3.10 Oscillator layout



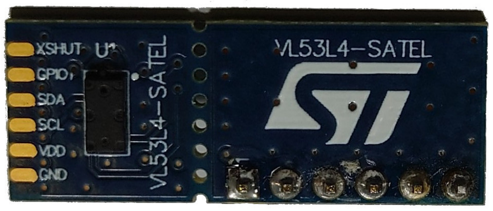
Another distinct area belongs to a buck converter in the upper left corner of the board, where it is sorrounded by LD49100 regulators to keep all power components nearby and maintain a clear and logical partitioning of the board space. Efforts were made to minimize interference due to buck converter switching activity, which is exemplified by the removal of copper under coil `L2` and the separation of the signal and the power ground. Furthermore, the copper regions and traces that carry power were appropriately sized to meet current requirements. Unfortunately, version 1.0 of the PCB also contained an inconvenient footprint for the 5.5mm x 2.1mm barrel jack meant for parts with round pins, which are less common than parts with wide flat pins. These two footprints, shown in Figure 3.9, are incompatible.

Lastly, the MCU is on the right and all of the decoupling capacitors are placed underneath it on the bottom side, as close to the power pins as possible. This was the most sensible arrangement because 0805 capacitors are relatively large and would obstruct access to the signal pins if they were to be placed next to the chip itself. On the contrary, the crystal oscillator is placed on the front side near the MCU according to the recommendations contained in the design guidelines, which state that oscillator traces should be as short as possible to optimize performance and minimize EMC susceptibility [36]. Electrical noise isolation is even further improved thanks to a guard ring around the oscillator. Details of its implementation are shown in Figure 3.10, where it can also be seen that there is no copper under the oscillator area. As a consequence, no high-speed signal traces that might cause interference are present in the layers below, and parasitic capacitance should also be reduced.

The test points that simplify diagnostics are spread across the board; to name a few the I^2C and CAN bus can be diagnosed on either side of their respective transcievers, GPIO signals have test pads on the bottom layer below each connector, and each power rail can also be checked. One glaring omission is the lack of test points for the power rail enable signals, as there is no other easy way to probe them and they are critical for the operation of the board.

3.3 Sensor daughter board

Figure 3.11 Commercial VL53L4 satellite board



The daughter board for the automotive dToF sensor was also designed from the ground up. Solutions such as SATEL-VL53L4CD with VL53L4CD dToF (Figure 3.11), which is an industrial equivalent, are sold by STMicroelectronics and would work with the control board without problems. However, they lack one feature that limits the flexibility of measurement setups. In some asymmetric arrangements, for example, when the sensors are on the same side of the protected area (that is, they are not in the opposing corners, as illustrated in Figure 3.12), the position of the transmitter and receiver is not the same relative to their field of view. Therefore, it is likely that the sensors would behave differently even in the optimal case where there are no objects in the background, there is no manufacturing disparity, and the ambient light is consistent and uniform.

As a workaround, it would be possible to rotate the board 180°, but that would also move the solder pads to a different place, so two different enclosures would have to be designed. As a proper solution, the custom 2-layer board shown in Figure 3.13 supports mounting of a dToF sensor on either side, so the transmitter and receiver positions can be selected during the soldering process. Decoupling capacitors will typically be mounted on the side opposite the sensor to make space for a cover glass. The most common cover glass model is the IR012C0-PM3D-A066 manufactured by Hornix Optical Technology and, for this reason, its footprint is highlighted on the silkscreen for reference. In cases where cover glass is not used, the capacitors may also be placed next to the sensor. Cables can be soldered on either side.

Figure 3.12 Simplified illustration of a difference in FoV for asymmetric arrangement

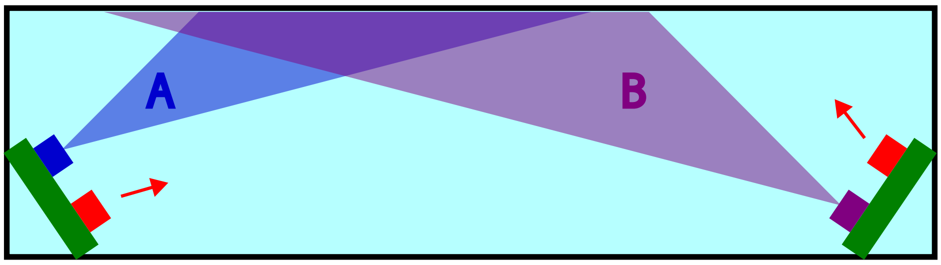
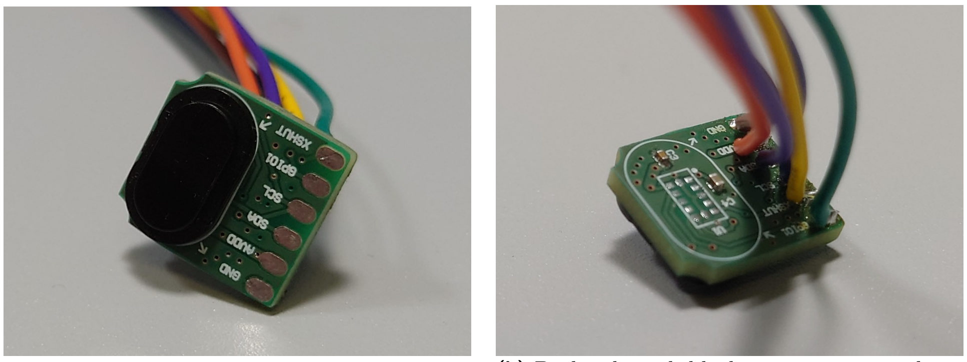


Figure 3.13 Custom reversible PCB for an automotive dToF sensor



(a) Front side with Hornix cover glass

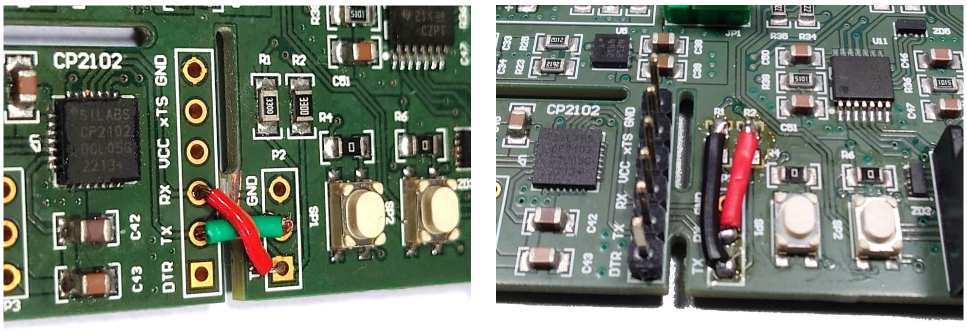
(b) Back side with blocking capacitors and unused alternative location for the sensor

3.4 Manufacturing and assembly

The boards were manufactured in the Czech Republic by the company Pragoboard. The POOL service manufacturing process suitable for prototype boards was chosen because it is their most affordable option and no additional features were necessary. The core material and the copper thickness could not be adjusted, but since the application does not rely on high-frequency signals and is not particularly demanding when it comes to power requirements, the default 18 µm copper foil in the outer layers and 35 µm in the inner layers (in the case of a 4-layer main PCB) did not pose a limitation. ITEQ-158 and ISOLA IS400 core materials were used for the sensor and the main board, respectively, and the total width of the PCB is equal to 1.5 mm in both cases. Matching stainless steel stencils were also produced for the precise application of solder paste.

Multiple ICs that are not easy to solder by hand were used, for example, ESD protection on sensor communication lines or the CP2102 UART transciever. So, from the start, the idea was to apply solder paste with the help of a stencil and then assemble the boards using a semi-automatic component placer. Since it is usually viable to apply solder paste only on one side, at least unless a dedicated stencil holder machine is available, most of the mandatory components were placed on the front side of the PCB during the design phase. The component placer allows the user to position each component gently and precisely, without smearing the applied solder paste, a feat that is very difficult to achieve if the components are placed by hand using tweezers. Smearing of a solder paste is dangerous because small solder balls can form outside of the individual pads, and they can later move and cause short circuits. The low-melt Sn42Bi58 solder alloy was used because there

Figure 3.14 Fix of the RX/TX wiring error



(a) The first option is to cut RX/TX traces on the bridge between UART module and the rest of the board. Then they can be reconnected at the two UART pin headers in the correct order.

(b) Alternatively, to avoid cutting into the PCB, resistors R1 and R2 that act as a failsafe against a short circuit can be removed, and their pads can be used instead of the P1 header.

is a lower risk of damaging sensitive components or the board itself by excessive heat, especially when it is necessary to fix soldering errors with a hot air gun.

With solder paste and components present, the boards were first baked at the appropriate temperatures in the reflow solder oven, and then some components such as connectors and pin headers had to be soldered separately by hand. All of the shortcomings of the first prototype that were critical to a board's function also had to be addressed. The barrel jack pins had to be filed to fit into insufficiently sized holes and the wrongly connected RX and TX signals of the UART transciever had to be crossed with the wires using one of the methods shown in Figure 3.14.

The finished board was visually checked for soldering errors before being powered on and then carefully tested to verify that each component worked as expected. An issue that manifested during testing was that the jumper that selects between USB and 12 V power source causes an undesirable behavior when the board is powered with USB and jumper is either not present at all or is set to a 12 V input without any power adapter connected. In that situation, because the RX and TX pins of the UART transciever are pulled high, the voltage passes through the UART pins of the MCU to a VDD power rail and because it is the only voltage source at that point, it is used to power everything connected to the VDD. However, these pins are not meant to provide power and their voltage is also less than the 3.3 V expected for a VDD. Fortunately, neither the board nor the CP2102 were damaged, but this undefined behavior is still not desirable and should be fixed in later versions.

Ideally, power would be automatically sourced from the 12 V supply if it is connected. One way to do this is to directly check the presence of the voltage on the positive pin. Most likely an operational amplifier that makes a comparison with some known reference voltage will be used for this purpose. The result can then be connected to a gate of a PMOS transistor or a dedicated load switch IC that will block USB power if a 12 V power adapter is used. Alternatively, the third pin of the barrel jack behaves like a mechanical switch that is normally closed and connected to a ground, but upon insertion of the plug it is opened. Therefore, it can also act as a control signal that switches power sources, with the difference that the board will not react to disconnection of the power adapter from the wall outlet. In either case, automatic supply switching would fix the aforementioned issue because the user would not have to manually reposition the jumper every time the USB becomes the sole power source. Not only would the behavior be deterministic, but user comfort would also be improved, as there is one less action that needs to be done when using the board.

Figure 3.15 Assembled dToF Control Board 1.0

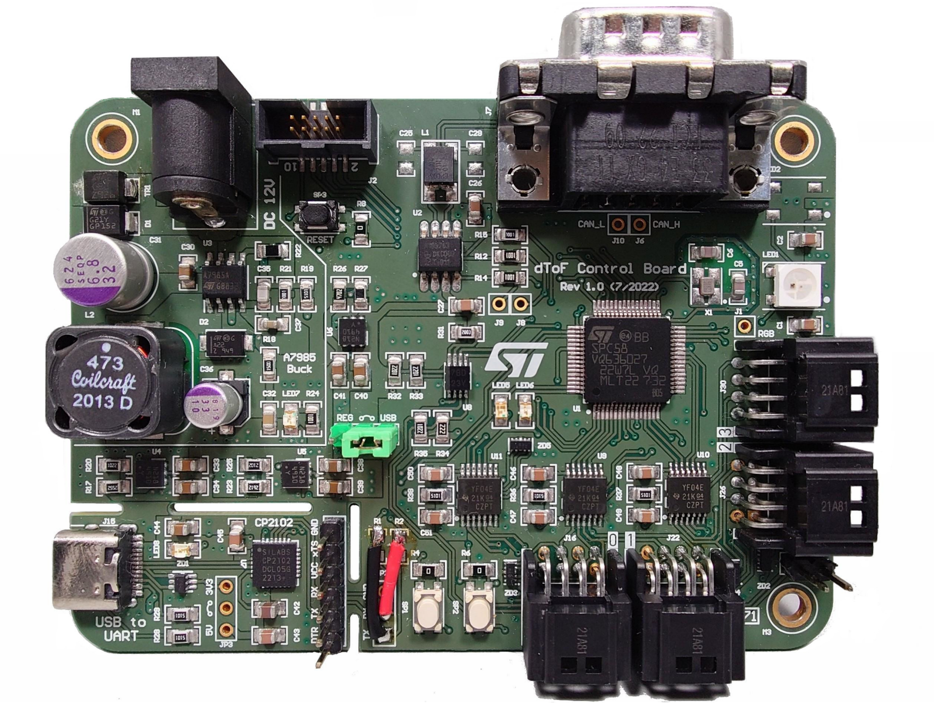
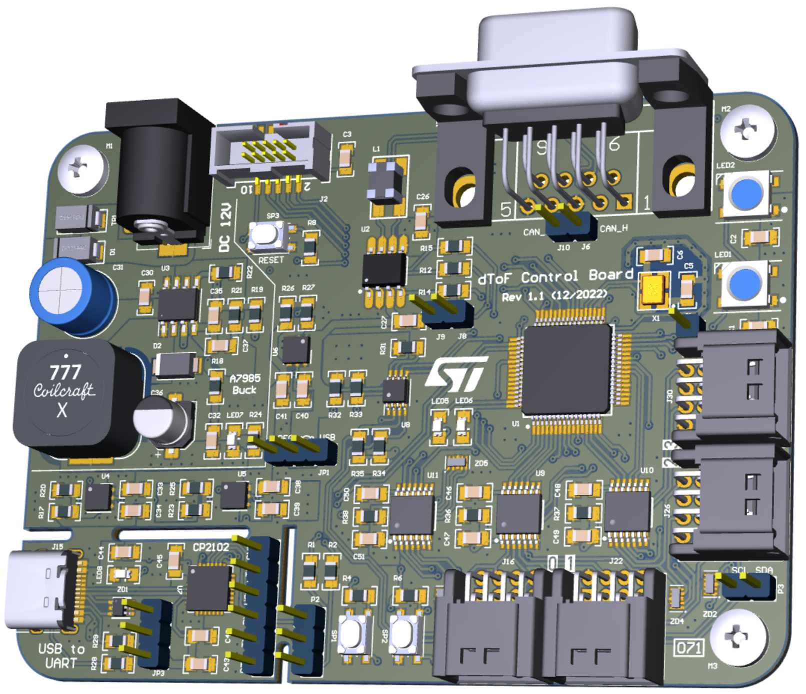


Figure 3.16 3D render of the dToF Control Board 1.1



Chapter 4

Design and implementation of firmware

There is not much community support for automotive MCUs like SPC582B; therefore, the best resource is the datasheet, guidelines, and tools provided directly by the manufacturer. For that reason, SPC5Studio 6.0, an official and free Eclipse-based IDE with a hardware configuration wizard and other useful features, was chosen to develop the firmware. Conveniently, many examples of projects are included with simple demonstrations of how to use different peripherals, timers, and other hardware features. The hardware-specific code is distributed and updated directly through the IDE, but once it is downloaded, the project can be built using a generated Makefile without any other software but the compiler compatible with the PowerPC architecture.

The first step to create a new project is as simple as choosing the exact MCU model in the creation wizard, in this case SPC582B60E1. Different aspects of the MCU are set through *components*, so only the required functionality is included in the source code. The firmware source depends on the following components:

SPC582Bxx Platform Component RLA Specifies the exact model of the MCU and is created by default

SPC582Bxx Init Package Component RLA Dependency of other essential components

SPC582Bxx Board Initialization Component RLA Allows changing the board name and assignment and settings of pins

SPC582Bxx Clock Component RLA Specifies oscillator parameters and clock sources for different peripherals

SPC582Bxx IRQ Component RLA Specifies handling and priority of external and internal interrupts

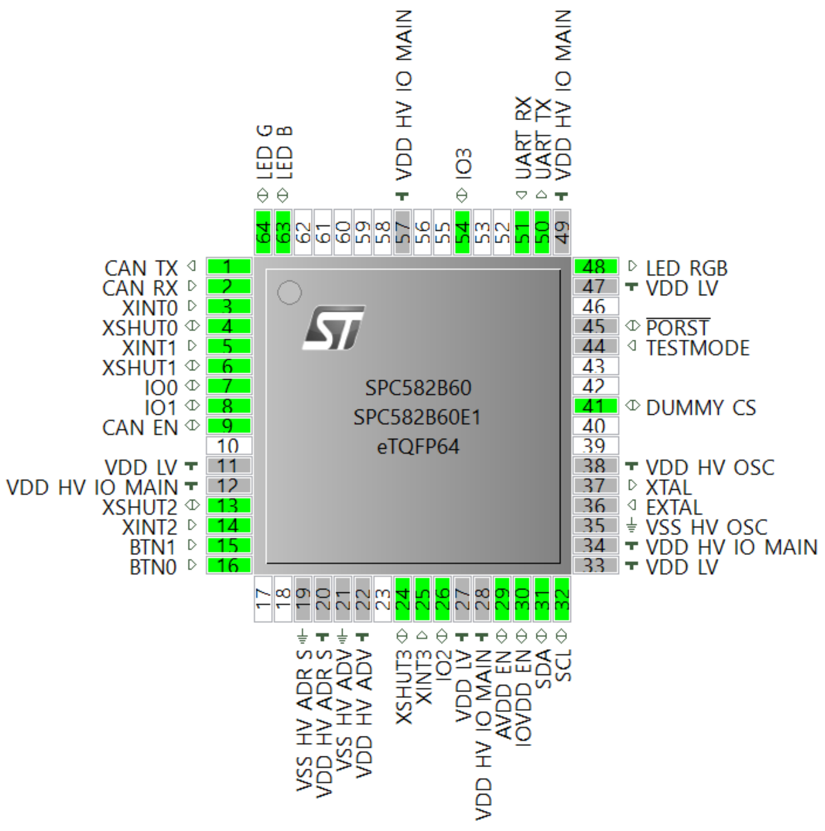
SPC582Bxx OSAL Component RLA Sets system timer parameters

SPC582Bxx Low Level Drivers Component RLA Important component used to configure communication peripherals, system timers, and DMA channels

Eeprom Emulation Driver Used for EEPROM emulation in flash for persistent storage

Flash Driver Dependency of the *Eeprom Emulation Driver*

Figure 4.1 SPC5Studio PinMap editor



Each component presents itself as a list of curated options that allow for granular configuration of the system even without the knowledge of internal registers. This is especially helpful when defining the parameters of individual pins, because the interface (shown in Figure 4.1) displays the MCU package, where the position of each pin can be quickly compared to the PCB layout, and its parameters are set by clicking directly on its physical position. Making changes to the rest of the system was similarly straightforward. The clock tree configuration did not need adjustments at all, because the default value for the crystal is 40 MHz and matches the physical component. The same goes for the system timer and EEPROM emulation. Changes in the *Low Level Drivers* and *IRQ* components will be discussed in more detail in the relevant sections.

The component settings are stored in the *configuration.xml* file placed in the project root. Before starting the build process, SPC5Studio checks for changes in this file, and it will generate the *components* folder that contains assembly and C source files with the necessary interfaces and configurations. In cases where some functionality is missing or does not work as expected, the generated components can be modified by the user. The SPC5Studio will update the *patch.xml* file accordingly so that any changes are remembered and reapplied during component regeneration. Components were not changed in this project, so compatibility issues with future versions of SPC5tudio are less likely to occur. The starting point of the firmware is in the *main.c* file that calls the main application loop. All user-defined code and external libraries are stored in the *source* folder and organized according to Table 4.1.

Table 4.1 Firmware source code structure

	Main API
platform	Frequently reused platform-specific code and platform initialization
app	Main application loop, interrupt handling and shared global structs
sensor	Sensor interface, abstracts the access to LA and LX drivers
meas	Functions that process sensor data and determine object presence
msg	Implementation of simple message protocol for recieving commands from GUI over UART
cmd	Defines functionality for valid command bytes accepted by the message protocol
	Additional modules
vl53l4	VL53L4 Ultra Lite driver
vl53lx	VL53LX Bare driver with histogram readout
argb	WS2812B Addressable RGB LED driver
printf	MIT licensed printf/sprintf implementation for embedded systems [37]

4.1 Platform specific code

The *platform* header file mainly contains common macros and constants used throughout the rest of the application, such as the number of available sensor ports and peripheral IDs. Platform-specific functions and function-like macros provide access to some peripherals, and the idea behind this was to simplify the porting of the firmware to a different platform by making the *platform.h* and *platform.c* only files that need to be changed. However, this goal proved to be difficult to achieve because, for example, the reading from the UART peripheral works asynchronously and `uart_read(*buffer, len)` only sends a request to wait for the arrival of a specific number of bytes and quits immediately. When the full message is received in the buffer, the interrupt is fired, and it is the responsibility of the application to process it. Therefore, it is challenging to contain this kind of functionality in a single function. Thankfully, adapting simpler functionality, like blocking delay that waits for a certain amount of microseconds/milliseconds, or readout of the current value of system timer should be more straightforward.

4.2 Application loop

The first function called by the `main()` is the `platform_init()` that initializes components provided by the SPC5Studio – this includes the UART, I^{2}C and SPI peripherals, timers, and EEPROM emulation drivers. It also enables interrupts and turns on the power rails for the sensors by setting the EN pin of the IOVDD and AVDD LDOs. The `app_run()` is called next and it performs initialization of the application settings and, most importantly, finds which sensors are connected and assigns at most two (in order based on their port number) to available measurement slots. Port numbers are visible on the PCB and can also be determined by checking the boot message that lists which ports are occupied. The number of measurement slots can be changed by increasing the

Table 4.2 Interrupt table

IRQ source	Pin (eTQFP64)	Handler
EIRQ 0	3	`dtof0_irq_handler`
EIRQ 10	5	`dtof1_irq_handler`
EIRQ 8	14	`dtof2_irq_handler`
EIRQ 13	25	`dtof3_irq_handler`
UART RX	51	`uart_rx_handler`
PIT0 Ch1 (1 Hz)	–	`wd_irq_handler`
PIT0 Ch2 (12 Hz)	–	`btn_irq_handler`

`SENSOR_MEAS_CNT` macro. If more sensors are connected, they will not be scheduled for ranging. The application will not continue if no sensor is connected.

The command that starts ranging is sent to the first available sensor, and the main loop that waits for any of the interrupts described in Table 4.2 is entered. One of the interrupts will come from the sensor after it completes the ranging and the only action inside the interrupt handler is storing the sensor's port number to the `dtof_irq_id` variable. When it is determined in the main loop that `dtof_irq_id` contains valid port nummber, these actions are performed:

1. Ranging for the sensor that caused the interrupt is stopped.
2. Ranging is started for the next available slot.
3. Results are evaluated with `meas_eval_obstacle()`, unless the evaluation is done externally through the GUI.
4. Indication light is changed to red if an object is present and green if the scene is empty.
5. `dtof_irq_id` is set to -1, and the watchdog is reset.

Interrupt handlers for events generated by timers and processed UART read requests also set just one variable that is then regularly checked in the main loop. This ensures that all actions are performed sequentially in correct order, and handlers are short and have as little performance impact as possible. One timer is used as a watchdog that can be used as a safeguard in situations where the ranging interrupt does not come in an expected time window, possibly because the sensor is disconnected or the cable is damaged. The watchdog value is reset with each successful sensor readout.

Other timer polls buttons, because GPIO pins to which they are connected unfortunately do not support interrupts. With each tick of the timer, the current button value is shifted into an integer variable that represents a short history of pin states. If the pressed state is present for a sufficient number of ticks, the action can be performed. This is done as a debouncing countermeasure, a phenomenon where due to mechanical imperfections a button might oscillate between values during state transitions, and this can cause unwanted press detections if only an immediate state is considered. Currently, buttons are used to pause/resume ranging for each sensor slot, but a more fitting use case will probably be assigned in the future.

Board settings, all data captured by sensors, and evaluation data are stored in the following global variables:

`Board_t board;` Contains an array of structs that represent each sensor port. Another array represents the measurement slots and is expected to contain pointers to elements in the previous array. The number of slots assigned is also stored here.

`Measurement_t meas;` Contains raw data captured from sensors, as well as data processed by the detection algorithm.

`Setting_t settings;` Contains board settings and parameters for the evaluation algorithm. This struct is designed to be saved into flash blocks that emulate EEPROM persistent storage.

The evaluation algorithm contained within the firmware is very simple, as it only compares the distance reported directly by the sensor to a fixed threshold stored in the `settings` variable. If any sensor detects something closer to this threshold value, the object will be detected. This is not ideal because histogram knowledge is not used in any way, but it made sense to delegate the experimental detection algorithm to a desktop application because it is easier to debug and a lot more tools and libraries are available. If a reliable algorithm is ever developed, it will be ported directly to a firmware. For now, only basic functions that calculate different parameters of the histogram are implemented.

4.3 Sensor interface

The first attempts to control the dToF sensor were done with a relatively simple VL53L4CD ULD driver, which was already tested with a VL53L4 satellite board and was supposed to work with the automotive variant as well. This driver is divided into API functions and platform functions. The API includes functions for reading the result, clearing the interrupt, and also for calibration and configuration. It is a well-designed and compact library, so it was initially used directly within the main application. The programmer only needs to implement platform functions that read and write data via the I^2C bus. The measured data are stored into a struct where two parameters are especially important - distance in millimeters and signal intensity. These numbers are aggregated by the firmware inside the sensor and do not provide a complete picture of the scene. As the distance of an object from the sensor increased, these two numbers became more and more ambiguous without all the underyling data used in their computation.

Interestingly, it was discovered that the driver intended for VL53L4CX dToF, which uses the histogram to distinguish multiple objects over larger distances, also works with the automotive variant. It is not clear whether the performance is the same as with a proper VL53L4CX, but it was still possible to obtain individual histogram bins along with aggregated information about distance and signal intensity. However, this driver is much more complicated and is structured differently than the VL53L4CD ULD driver, so to allow usage of both in the application, a universal interface was created that abstracts both of these drivers, and it can be decided before compilation which one should be used.

Each sensor is represented by a **`struct`** in Listing 4.1. Some information is common between the CD and LX drivers, such as the need to know where the XSHUT pin is connected and what is the port number of the sensor. The difference is that the LX driver uses a custom object that represents the I^2C address of the device, whereas the CD driver simply stores the address as a byte. When it comes to ranging parameters contained within the `params` member, the LX also needs additional information about the preffered distance range, but only `VL53LX_DISTANCEMODE_MEDIUM` and `VL53LX_DISTANCEMODE_SHORT` can be used. This is probably the main limitation of the automotive dToF, as the `VL53LX_DISTANCEMODE_LONG` seems to be exclusive to a genuine VL53L4CX. Nevertheless, both drivers need to know the timing budget and the inter-measurement period. The timing budget specifies the duration of active ranging, when the VCSEL is active, and

Code listing 4.1 Wrapper representing a status of dToF sensor

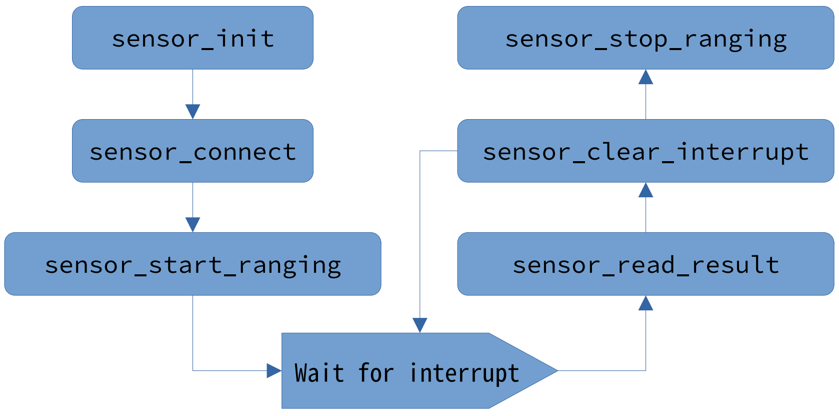
```
typedef struct
{
    uint8_t id;///<Internal ID of the sensor (port number)
    SensorState_t state; ///<Current sensor state

    struct
    {
        #ifdef SENSOR_USE_LXDRIVER
            VL53LX_Dev_t lx;
        #endif
        uint8_t addr; ///<Current I2C address
    } dev;

    uint16_t xshut_pin; ///<XSHUT pin number
    uint16_t xshut_port; ///<XSHUT port number
    SensorRangingParams_t params; ///<Ranging parameters
} Sensor_t;
```

the valid values are in an interval from 0 to 200 ms. Longer ranging will result in better accuracy and longer maximum distance, while shorter ranging saves power [30]. The inter-measurement period specifies the time between starts of two consecutive rangings. When it is 0, the VCSEL is always active, but it can also be as high as 5 seconds. Larger values will reduce power consumption because VCSEL is active less frequently and also because the sensor goes to sleep mode if the timing budget ends and there is still time left until next ranging [30]. Lastly, a `SensorState_t` is an **`enum`** type with four different states for when the sensor is disconnected, connected but idle, connected and ranging, or in an invalid state.

4.3.1 Functions

The sensor API was designed according to the common dToF ranging procedure shown in Figure 4.2. All functions require a pointer to a sensor wrapper `Sensor_t` as the first parameter.

Figure 4.2 Sensor API workflow



- **void** sensor_init (Sensor_t* sensor, **uint8_t** id, **uint8_t** dev, **uint16_t** xshut_pin, **uint16_t** xshut_port, SensorRangingParams_t* params);
  - Initializes the sensor wrapper with values passed as parameters and guarantees correct behavior regardless of the backend driver. It is not recommended to initialize `sensor` by assigning values to individual members.
- **void** sensor_connect (Sensor_t* sensor_array, **const int** cnt);
  - Here, the array of sensors and the number of sensors in the array are expected. Consider that one of the sensors needs to be initialized and that all of the sensors have the same I^{2}C address after boot. The `sensor_connect` then must be able to disable the rest of the uninitialized sensors so that the command that changes the I^{C}C address is accepted only by a single sensor. When this function ends, all initialized sensors will have a desired address, their state will be set to `SENSOR_STATE_IDLE`, and ranging parameters will be applied.
- **void** sensor_disable (Sensor_t* sensor);
  - Disables the sensor with an XSHUT pin set to low and waits 50 ms.
- **void** sensor_enable (Sensor_t* sensor);
  - Enables the sensor with an XSHUT pin set to high and waits 50 ms to give it time to enter SW standby.
- **void** sensor_reset (Sensor_t* sensor);
  - The shorthand for `sensor_disable` and `sensor_enable` called in succession
- **void** sensor_start_ranging (Sensor_t* sensor);
  - Starts ranging and sets the sensor state to `SENSOR_STATE_RANGING`
- **void** sensor_stop_ranging (Sensor_t* sensor);
  - Stops ranging and sets the sensor state to `SENSOR_STATE_IDLE`
- **void** sensor_toggle_ranging (Sensor_t* sensor);
  - Ranging is stopped if the sensor state is `SENSOR_STATE_RANGING` and started otherwise.
- SensorRet_t sensor_read_result (Sensor_t* sensor, SensorResult_t* result);
  - Reads the measured data into a `result` parameter. The return type is an **`enum`** with error states. If the data are not ready or are not valid, the `SENSOR_RET_ERR` value is returned. The `SENSOR_RET_OK` is returned otherwise. This can be useful if an interrupt cannot be used and polling is preferable.
- **void** sensor_clear_interrupt (Sensor_t* sensor);
  - Clears the interrupt and resumes ranging.
- **uint16_t** sensor_id (Sensor_t* sensor);
  - If the sensor is connected and working, this function should always return `0xEBAA`, even before any initialization is performed.
- **uint8_t** sensor_isconnected (Sensor_t* sensor);
  - Checks if the sensor is connected and if the initialization with `sensor_connect` was performed

4.3.2 Result type

Code listing 4.2 Type representing a sensor data

```
typedef struct
{
        int32_t max; ///<Maximum value in histogram
        int32_t sum; ///<Sum of histogram values
        int32_t bins; ///<Number of used bins
        int32_t hist[SENSOR_HIST_SIZE]; ///<Histogram data
} SceneData_t;

typedef struct
{
        uint32_t distance_mm; ///<Distance
        uint32_t signal_rate_kcps; ///<Signal rate
        uint32_t ambient_rate_kcps; ///<Ambient rate
        uint32_t number_of_spad; ///<Number of SPADs
        uint32_t sigma_mm; ///<Sigma (standard deviation)
        SceneData_t scene; ///<Histogram of the scene
} SensorResult_t;
```

The data from sensors are stored into a `SensorResult_t` type shown in Code Listing 4.2. The distance, signal rate, and ambient rate are self-explanatory, but both drivers also report two other interesting metrics. The first is the number of SPADs that were used for the measurement. This number will change with the intensity of the signal and closer targets will require less active SPADs. Sigma refers to a typical notation of standard deviation and its purpose is to describe how precise is the measured distance. Lower values mean that the confidence in a measured value is higher.

The histogram is part of the `scene` member that also stores a number of bins and simple metrics that can be easily calculated during the histogram readout from the LX driver and could be useful for the detection algorithm. For now, only a maximum value and summation of bins are calculated, but in the future this might change. If the CD driver is used, the number of bins is always zero, indicating that the histogram is not available.

4.4 Message protocol and command interface

To communicate with some other device via the UART, some protocol must be agreed upon. The most influential choice is whether to convert data to strings and send them as ASCII characters or whether they should be transmitted in their binary form. The advantage of the first approach is that the communication is easier to read and understand by the human, even if all they have is a data dump captured by a logic analyzer or an oscilloscope. The downside is that the conversion to characters creates a lot of redundancy and is harder to parse by a machine on the receiving end. Moreover, if data are sent with as little overhead as possible in binary form, faster communication speeds can be achieved. In practice, both approaches are used in this project. The seldom debug and error messages are transmitted as strings because they must be understandable to the user under any circumstances. The compact implementation of `printf` is used for this purpose, because it is a useful and familiar interface for C programmers. Every `printf` call blocks until

Figure 4.3 Structure of the message frame

Header start	Command	Data size	Stop	Data start	Data	Stop
`0x2A`	1 byte	1 byte	`0x2D`	`0x2B`	*Data size* bytes	`0x2D`

the full string is transmitted so that fast subsequent calls do not result in data loss due to buffer rewrites.

Because of the way the UART driver provided by SPCStudio is implemented, receiving the data into the application has more caveats and the binary approach made more sense. It was already mentioned that to read data, a request to UART peripheral is made and an interrupt is invoked when the transfer is complete. The key part of the request is the number of bytes to read, which means that at any point, the size of the incoming message must be known in advance. This is a frustrating requirement for any text-based protocol, so the decision was made to control the firmware exclusively through a graphical application that will handle the communication in binary form.

Figure 4.3 shows the structure of a single frame according to a simple binary protocol that was implemented for the purpose of this application. Every frame is split into a header and data. Each part separately must contain a correct start and end byte; otherwise, the message is deemed invalid. The header and data need a different start byte, but the end byte is the same for both. The header contains two bytes of useful information: the command byte and the data length. If the target device supports the required command, the header will be acknowledged with a `0x41` byte. The response with a `0x45` byte will be used if the command is not recognized. The data portion of the message may only be sent after the valid acknowledge byte is received to ensure that the next UART read request has already started and that the data will not be lost. The maximum data length is limited to 255 bytes, which is good enough to transport a full histogram for a single sensor, but in hindsight, it would probably have been better to use at least 16 bit integer to allow larger messages.

When the full message is received, the device will send a response in the same format. The header will contain the command byte of the incoming message, and the data length will correspond to the size of the data produced by the command. The only difference is that there is no waiting for acknowledge after the header and the data are sent immediately, because the GUI buffers incomming data and the confirmation is not necessary.

Commands themselves are defined as functions that share a prototype from Code Listing 4.3. One reason for this is to separate them from the parsing logic so that the same command function can be used with a different protocol or even peripheral. This would be important if in the future the CAN bus was used as another input channel. Therefore, each command only requires a pointer to the input data, the size of the input, and a pointer to the buffer where the command output will be stored. The size of the output is passed to the caller through a return value. Each command assumes that the output buffer is allocated and is sufficiently large. In practice, the **`uint8_t`** `msg_out_buffer[MSG_MAX_LEN]` buffer that can accommodate the largest 255-byte message is always used by the message parser that calls the command function.

Another advantage of having commands that share the same prototype is that they

Code listing 4.3 Command prototype

```c
typedef uint8_t (*CommandFunc_t)(uint8_t* in, const uint8_t in_len,
                                 uint8_t* out);
```

Table 4.3 Table of implemented commands

Name	Command byte	Params (bytes)	Response (bytes)
`cmd_read_sensor`	'A' `0x41`	*1:* Port number	*4:* Distance (mm) *4:* Ambient rate (kcp/s) *4:* Signal rate (kcp/s) *4:* Number of SPADs *4:* Sigma (mm)
Reads the last captured measurement. Items in the response are encoded as **`uint32_t`**. If the port is not active, the response will be *Empty*.			
`cmd_set_evalmode`	'D' `0x44`	*1:* `0x0` onboard `0x1` remote	*Empty*
Sets the evaluation mode.			
`cmd_set_eval`	'E' `0x45`	*1:* `0x0` empty `0x1` object `0x2` fake object	*Empty*
Sends remote evaluation result. Has no effect if the evaluation mode is set to *onboard*.			
`cmd_read_hist`	'H' `0x48`	*1:* Port number	*96:* 24 histogram bins
Reads the last captured histogram. Each bin is encoded as **`uint32_t`**. If the port is not active or the histogram is not available, the response will be *Empty*.			
`cmd_set_dist`	'S' `0x53`	*1:* `0x0` read `0x1` write *2:* Distance (mm) only if *write*	*2:* Distance (mm) or *Empty* if *write*
Can be used to either read the current distance treshold for the *onboard* algorithm, or to set a new one and store it into persistent memory. Distance is encoded as **`uint16_t`**.			

can be easily accessed by function pointers. The pointer to every command is stored in the `cmd_links` array and every position in this command table corresponds to a single command byte. Therefore, the message parser simply calls the function stored in the correct position when the full message is received. This process should be faster than the comparison of command bytes, and the code is also easier to maintain because, to add a new command, the programmer simply writes the function with a correct prototype and adds its pointer to a free position in the command table. The index in the command table is not the same as the command byte, and an offset is applied so that the first position matches the capital 'A' according to ASCII table. This means that every command byte is a printable character, which can simplify debugging. The full list of commands currently implemented is in Table 4.3.

4.5 ARGB LED

Figure 4.4 WS2812B protocol

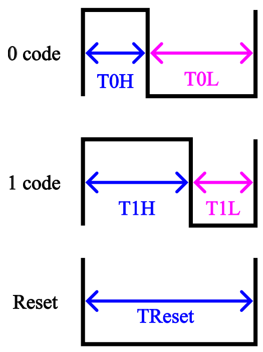


Table 4.4 WS2182B timing requirements

Interval	Period	Tolerance
TxH + TxL	1.25 µs	±600 ns
T0H	0.4 µs	±150 ns
T0L	0.85 µs	±150 ns
T1H	0.8 µs	±150 ns
T1L	0.45 µs	±150 ns
TReset	$\geq$ 50 µs	—

The WS2812B LED accepts information about color as a series of bits encoded according to Figure 4.4. Each bit has a fixed period and always starts with high level and then goes to low level. If the duration of high level is longer than the duration of low level, the bit has value 1, and if it is shorter, the bit is 0. The exact ratio of high to low is specified in the datasheet for each bit value, as well as the timing tolerance. Each LED requires 24 uninterrupted bits that carry 8-bit brightness values for green, red, and blue colors (in this exact order). Pulling the input line low for at least 50 µs activates the color as the received bits are loaded into a driving circuit.

To demonstrate the full control sequence, consider a configuration with two daisy-chained LEDs. The input buffer of WS2812B acts like a shift register, and after the first 24-bit frame is received and another is sent in succession (without a reset signal in between them), it will start shifting the received bits to the next LED in chain. Therefore, the first 24 bits will end up in the second LED, and at that point, the reset signal may be used to apply the color and finish the sequence. As a consequence, even when only one LED is required to change color, the color of all preceding LEDs in the chain must also be rebroadcast, and so the MCU must store the current status of the daisy chain in a buffer.

The protocol could be implemented with software control of the GPIO pin, but the more optimal and less resource-intensive way was to utilize the SPI peripheral. SPI traditionally requires four signals: clock (SCK), data in (SIN), data out (SOUT), and chip select (CS). The signal that is connected to the data input of the LED is the SOUT – SIN is of no use because no data is coming from the LED and CS is only useful in real SPI communication where there is a need to specify target device. The data sent over the SPI are organized into n-bit frames and the duration of each bit is determined by the frequency of the SCK.

Code listing 4.4 Procedure that changes one color in the ARGB buffer

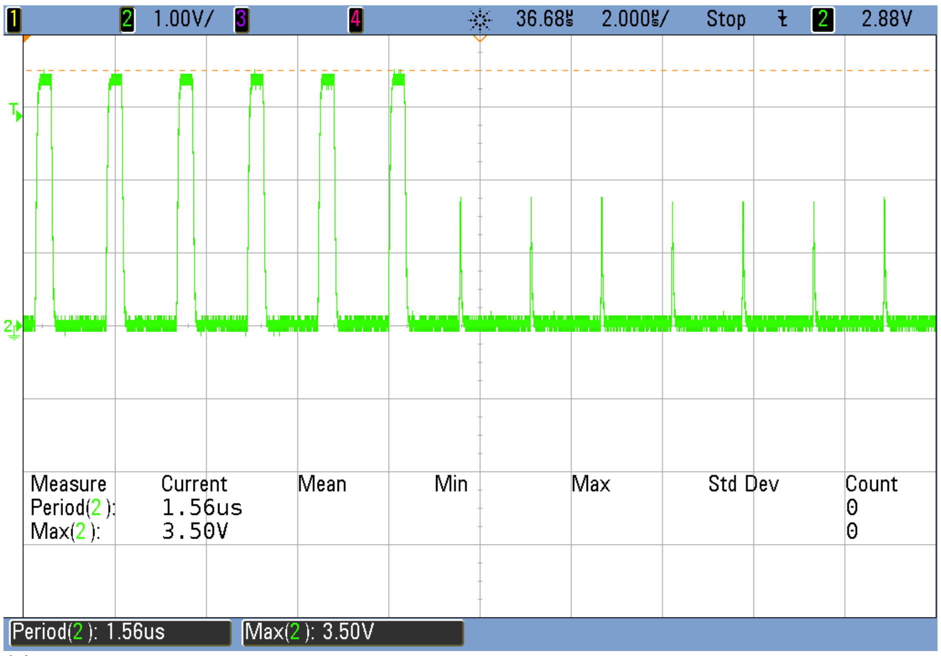
```
void _argb_set (const uint8_t num, const Color_t color)
{
    unsigned long color_cp = color;
    int idx = (ARGB_NUM_LEDS - num - 1) * ARGB_FRAME_SIZE
                + ARGB_FRAME_SIZE - 1;

    for (int color = 2; color >= 0; color--) //Starts from LSB
    {
        int color_offset = (2 - argb_order[color]) * 8;
        for (int i = color_offset; i < color_offset + 8; i++)
        {
            if (color_cp & 1) argb_led[idx - i] = ARGB_HIGH;
            else argb_led[idx - i] = ARGB_LOW;
            color_cp >>= 1;
        }
    }
}
```

The idea is to set the clock frequency so that the duration of a single frame is the same as the duration of one bit in the LED protocol. Then the data inside the frame can be set so that it begins with a series of ones followed by zeros. The number of ones in the frame will control the length of the high pulse and thus determine whether the LED receives one or zero bit. The SPI was configured to use 8-bit frames, because they are convenient to work with, and the clock frequency was set so that one frame has a period of 1.6 ns, which is within the specifications of WS2812B. As a consequence of these settings, a single bit inside the SPI frame controls the value of SOUT for 0.2 ns. The optimal frame values were chosen to be as close as possible to the T0H and T1H values in the datasheet: `0xf8` frame for 1 and `0xc0` for 0.

Unfortunately, the SPI approach had one problem – the SOUT line is pulled high by default. Naturally, it was expected that it can be controlled with zero frames, but this did not work because even if two zero frames were sent in succession, a short voltage spike always appeared on the SOUT. Its value was roughly half of the VDD and this was enough to be detected by the LED as a positive edge. This meant that it was impossible to create a long enough reset signal. This behavior seems to be a software or hardware bug, but it was fortunately possible to circumvent it thanks to the ability of each GPIO pin to invert the output value. The frames must also be inverted to correct for inversion at output, but now, when the SPI is idle, the SOUT will be low and it just must be ensured with a blocking wait that the next transaction will not start until the reset period ends. The measurements from the oscilloscope are shown in Figure 4.5 and highlight the problematic behavior of the SPI peripheral. It was also verified that the timings are correct and in accordance with the SPI settings.

In the end, the LED control was fully operational. A code was divided into an API and platform functions that control the SPI. Some colors were named and are part of the **`enum`** `Color_t` type. Only the first 24 bits are used, and the order of colors is red-green-blue, which is the typical order in which the colors are specified in HTML and other common desktop software. The correct order for the WS2812B is achieved later in a platform function that fills the new color of the LED into an SPI buffer.

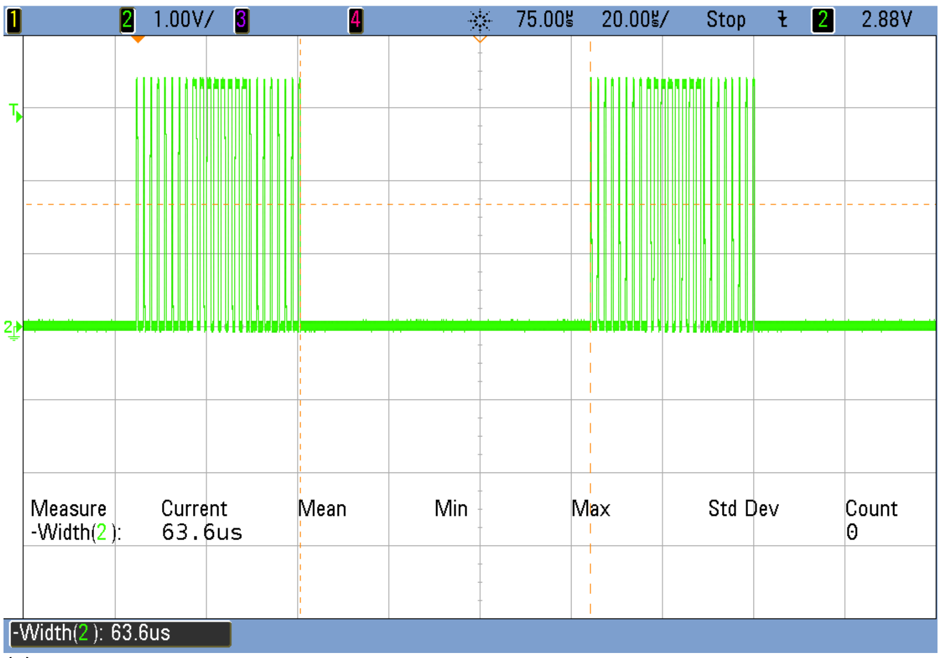
Figure 4.5 Oscilloscope captures of the WS2812B data line



```
Measure      Current    Mean    Min    Max    Std Dev    Count
Period(2):   1.56us                                      0
Max(2):      3.50V                                       0
```

```
Period(2): 1.56us    Max(2): 3.50V
```

(a) Reset signal generated by a series of zero frames. Brief voltage spikes that interrupt the reset sequence may be observed.



```
Measure      Current    Mean    Min    Max    Std Dev    Count
-Width(2):   63.6us                                      0
```

```
-Width(2): 63.6us
```

(b) Valid reset sequence that was obtained after the output pin values were inverted. The reset duration was measured to be 63.6 µs between two subsequent color changes.

Chapter 5

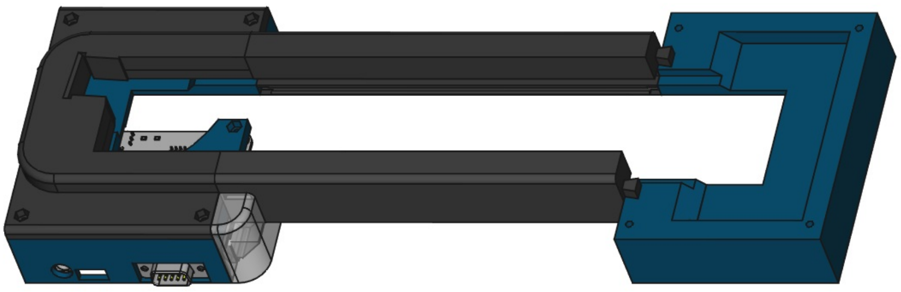
Enclosure design and assembly

The design of an enclosure was a balancing act between esthetics and customizability. It needed to look seamless for demonstrations, with each component tightly integrated, but it was also important to ensure that a small modification of the sensor placement does not mean that the entire enclosure needs to be manufactured again. The original plan was to use FDM 3D printing, which is relatively cheap and fast, but making large prints is still very time consuming and most of all inefficient, as the prototype versions of the model usually end up as waste that is hard to recycle. To solve this, the key parts of the sensor housing are modular, small, and quick to print. In addition, a great deal of effort was put into saving material and reducing print time by eliminating any redundancy in the design.

Most of the parts were designed in FreeCAD, which is a free and open-source graphical CAD software. The use of a graphical CAD was helpful because the mounting spot for the PCB had to be precise, with all connectors easily accessible through well-placed cutouts. Therefore, the workflow required the import of a PCB 3D model with all components, which could then be used as a visual reference during modeling, and all clearances could be verified before manufacturing. The customizability aspect of the enclosure stems mainly from the fact that FreeCAD allows for parameterization of parts, not from the ability to adjust parts after printing. Important dimensions or radiuses exist as a variable that can be referenced and used inside formulas.

However, the use of a parameterization is to a large extent hindered by a bug known as *topological naming problem* present in FreeCAD version 0.20, which was the latest at the time. When the placement of some geometry depends on a feature of another geometry and this parent geometry gets modified to such an extent that the number of faces and vertices changes, the design will most likely break because all the features of the parent

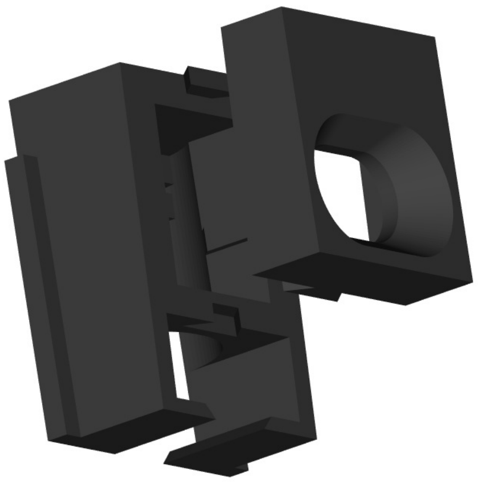
Figure 5.1 3D render of the enclosure



will get internally renamed and links with child geometries will break. For example, this can manifest if the variable that controls the offset for the M3 nut slots on the lid is modified so that the slots will be completely inside the lid. This would be useful because the 3D printing of the lid could be stopped before the layer that covers the slot is printed, and the nuts could be placed into slots and then covered by the remaining layers. From an aesthetic point of view, the enclosure would look nicer and the assembly would be easier, since there would be no need to keep track of every nut. Basically, this could be a differentiating factor between the final and experimental versions of the lid. Changes like these will hopefully be possible in later versions, but for now, it is important to manually check that every feature is placed correctly with every parameter update.

Five different components are included in the FreeCAD project file: base, base with PCB mount, base lid, side cover, and light diffuser. All of them encase the glass pane, which serves as the detection surface. The 3D renderings and the final assembly are shown in Figures 5.1 and 5.3. While the parameterization of the abovementioned components would cause merely unpleasant inconveniences due to the bugs in FreeCAD, designing a parametric sensor holder in this software would be nearly impossible. The final design of a sensor holder is shown in Figure 5.2 and is made of two interlocking parts that have a PCB with a sensor securely attached between them. The idea is to place this holder in a slot found on the lid, which will define the position and direction of the sensor. The bottom of the front part then rests on the glass pane, whereas the back part extends below the glass into a cavity in the base. This cavity serves as a path for cables that provide power and data to the sensors. It is larger than the holder to provide wiggle room for different lid configurations, so only a lid needs to be reprinted to test different sensor positions. Because the only way to remove or insert the sensor from the holder is to slide the front part down, the sensor is secure as long as the whole housing is in place, pressed against the glass.

Figure 5.2 Sensor holder



The difficult part was implementing an easy way to change the tilt of the sensor and its distance from the glass surface. This would require a lot of sketching on angled planes, which is problematic in FreeCAD, especially when the position of the said plane would shift depending on the user parameters. Therefore, it was easier to move the sensor holder design to OpenSCAD, an alternative free and open-source CAD software that translates text files written in a custom scripting language into 3D models. Parameterization is a key feature of OpenSCAD and generally causes no problems, or at least the problems it causes can be easily solved, and it is the responsibility of the designer to make sure everything works as it should. The user is only presented with a list of parameters, which can be easily changed without any understanding of the underlying script, and the correct 3D model is generated based on the input values. Each sensor holder is also marked with two most important parameters - sensor tilt and sensor offset from the glass. They are printed on top of the holder so that each version can easily be identified.

5.1 Printing recommendations

Base (with or without PCB slot) should be printed vertically, as it was designed in a way that allows printing without supports in this specific orientation. The cutout for a sensor holder has been chamfered and round holes for screws cause no issues. Even the CAN bus connector is printable because the right edge can be bridged easily, at least as long as there is a bottom edge that can serve as a bridging point. Hence, the current value of the base height should not be decreased because it leaves a minimal bottom edge that can be printed.

Lid is printed in its natural orientation. For variant with buried nut slots, the print may be paused so that the nuts can be inserted and permanently embedded into the lid.

Side covers must be printed with supports below the dovetail lock.

Light diffuser should be printed in its natural orientation and with the smallest possible infill. The orientation and amount of filling affect how the light will look, and this arrangement worked best. Supports are not necessary.

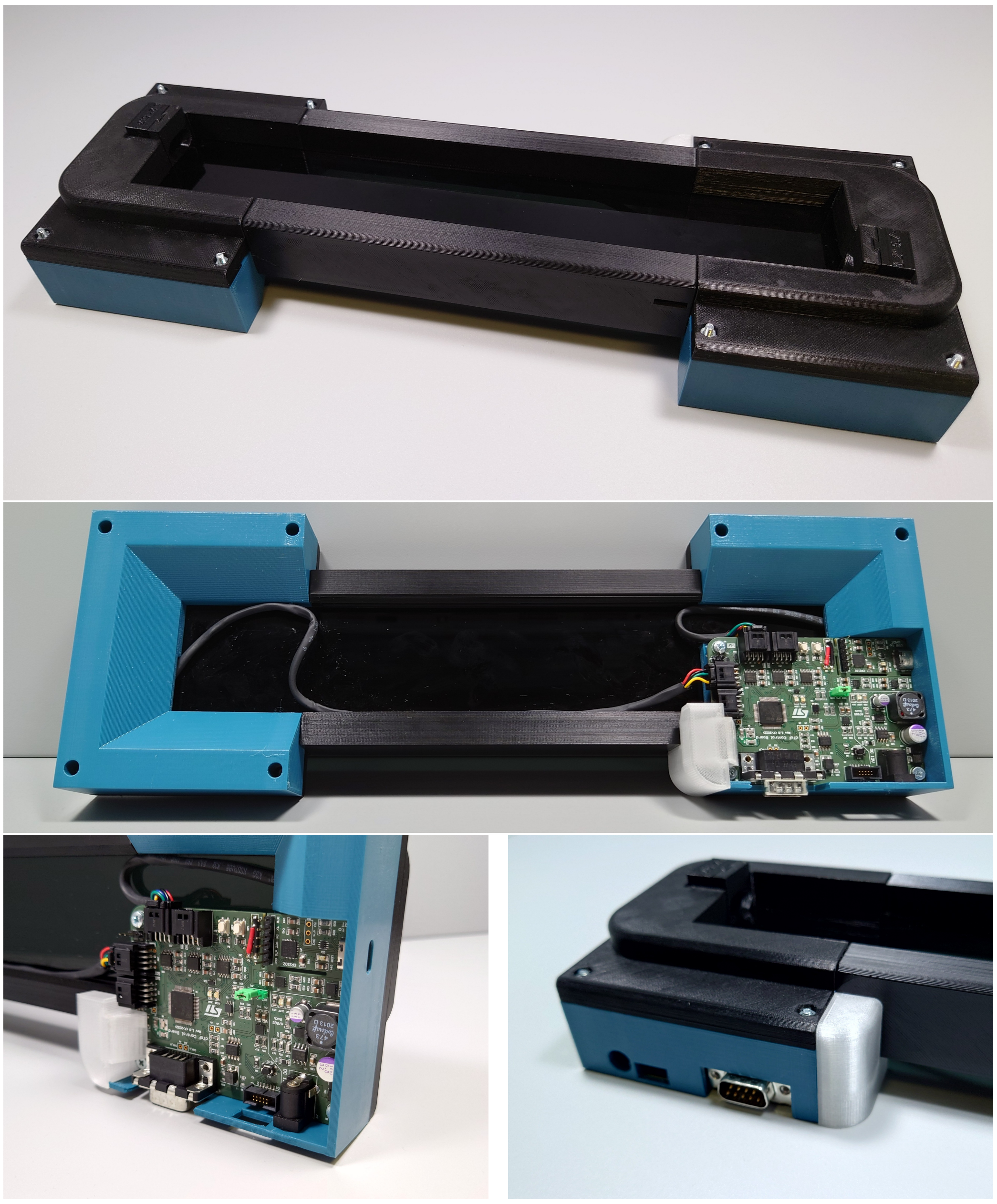
Sensor holder is printed in two parts. The dovetails will probably need a sanding because the front and back parts are printed in different orientations.

5.2 Assembly instructions

The assembly is carried out in the following order:

1. Insert the PCB into its corresponding slot at the bottom of the correct base. On the inner side of the wall with the USB-C connector cutout, there is a shallow ridge into which the board slides. This ensures that even though there is no screw in the corner near the USB-C connector, the bord will be firmly fixed in its place.
2. Secure the board with a single M3 screw placed in the hole marked M2 on the PCB. It is in the corner opposite a USB-C connector and the nut is placed from above into a hexagonal slot. The screw should be of the correct length to avoid any contact with the bottom of the glass surface.
3. Press-fit side covers into dovetail slots on the upper part of the base. Ensure that the hole for a light diffuser is placed on the side where the PCB is located.
4. Slide the glass pane into a railing formed by the two side covers.
5. Press-fit the side covers into an opposing base (without the PCB slot). This will restrict the glass pane from moving.
6. Guide sensor PCBs from below the glass into a cavity intended for a sensor holder.
7. Press-fit lids on both sides without losing the sensors. They should be accessible through the sensor holder slot on the lid.
8. Guide each sensor PCB through the back part of the sensor holder and then press the sensor into the front part. Carefully slide the front upward against the back so that the dovetail locks fall into a place.
9. Push sensor holders into their respective slots and attach both lids with 8 M3 screws and nuts.
10. Attach the light diffuser and press the sensor cables into the cable guides on the side covers if necessary.

Figure 5.3 Photos of the fully assembled enclosure



Chapter 6

Desktop application

The specialized GUI takes a notable amount of time to develop, which might not always be justified, but if it is done well and provides features that simplify the controls of the device or frequent operations, it is always appreciated. Here, the main purpose is to visualize histograms from the sensors, as arrays of numbers are harder to imagine and relate to each other than a graph that represents them. This visualization could be done in some data analysis software like Matlab or Excel, but a dedicated application typically provides several advantages. It works independently and does not have to rely on any other third-party software, which is often not free and requires licensing. Moreover, functionality and appearance can be extensively customized to suit more needs than just plotting of graphs, so the application can become a complete toolkit that also handles settings for the hardware, logging, diagnostics, and so on.

The Avalonia UI was the framework of choice, because it is licensed under the MIT license, thus completely free to use, and also because it is basically a modern version of WPF, Microsoft's library for GUI development, which has a long history and good support. However, Avalonia has the edge over plain WPF because it supports multiple desktop and even mobile platforms. The workflow is otherwise very similar, and typically the MVVM (model-view-viewmodel) architectural pattern is encouraged. This architecture isolates the graphical elements and styling (view) from the data and logic (model), so that the backend can be tested independently. The viewmodel is a class that bridges this gap and exposes data from models to views through data bindings. Views like windows and user controls are described in the XAML markup language and C# is used for the models, viewmodels and any other functional code.

An example of data binding is shown in Code Listing 6.1, where the TextBlock element is created in XAML and two of its properties are set: FontSize is specified with constant, but the actual Text is *binded* to a property SumSignal in ViewModel with special syntax. When elementary types are concerned, as in this case, the binding converts a numerical variable to a string automatically[1], but a user-defined converter may also be specified to resolve conversion in cases where an implicit conversion does not exist, for example, when the target type is part of the Avalonia framework, such as colors or font styles. With property binded, it is guaranteed that whenever its value changes, the Text in TextBlock also gets updated, even though the ViewModel has no knowledge about its existence. This is due to a special setter that notifies binded elements about the change. In the GUI, the MVVM pattern was followed as closely as possible, and the data in the UI elements were always set with data bindings. It is beyond the scope of this thesis to go further into the

[1]This is true for every class that implements ToString() method, because the Text property of TextBox is of type string.

Code listing 6.1 Example of data binding in Avalonia UI

```
<!-- View.axaml -->
<TextBlock FontSize="14" Text="{Binding SumSignal}"/>
```

```
// ViewModel.cs
private float _sumSignal = 0;
public float SumSignal
{
    get => _sumSignal;
    set => this.RaiseAndSetIfChanged(ref _sumSignal, value);
}
```

intricacies of the source code, but knowledge of these two principles should be sufficient to understand how and from where the graphical elements get the data.

To build the application, .NET 6.0 must be installed on the system, and then a command `dotnet build` must be called from the *gui* folder in the project repository. Calling the `dotnet run` will build and then start the application, but the application binary is also present in the *bin* directory so it can be used without any special commands. The `dotnet publish` command is however recommended to create a binary that is more suitable for distribution.

6.1 Board View

Upon launch, the application will go into board view. Here, the current status of all 4 sensor ports may be observed, and active ports will have their histogram data and other information updated in real time. The currect view is always highlighted in the menu on the left that is used as a main navigation element. Three menus in the upper row of the screen are visible across all views and are used to connect to a device, view the connection status, and view the object presence evaluation, respectively.

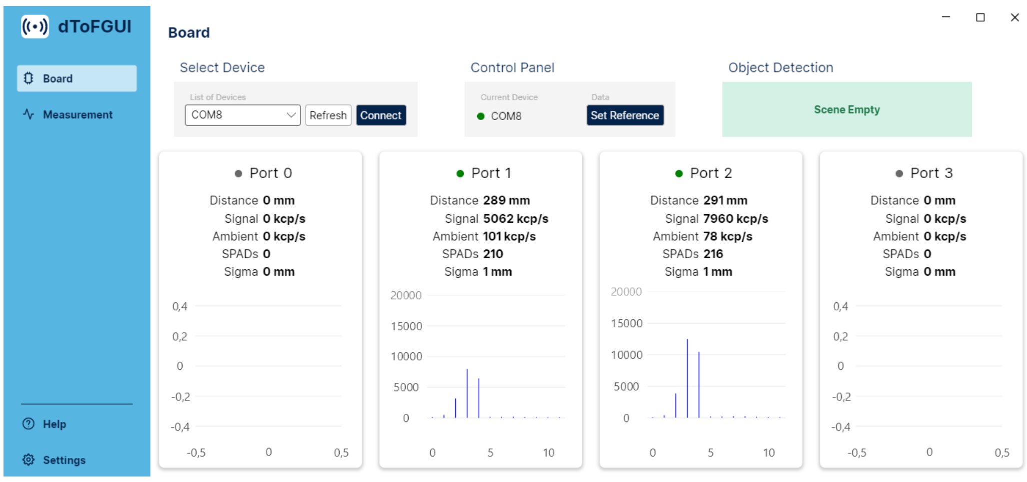
First, the correct UART device must be selected from the list of available devices. This list can be refreshed if the device was not connected before the application was launched. When the connection is successful, the dot next to the name of the interface will turn green and the sensor data will start updating immediately. Individual bin values may also be observed interactively with a mouse hover.

6.2 Measurement view

To start object detection, the reference scene must first be set. This can be done from any view, and the scene should be empty and without any significant background elements during this process. The algorithm will then start processing the histogram data and the current detection status will be displayed in the top right corner. The processing is done inside the application, and currently the algorithm is very basic and observes only two properties of the histogram. The first is the ratio of neighboring bins between the reference and the current scene, and the second is the bin difference compared to the reference. If the ratio is higher than 2 in any of the first four bins of the histogram, the object will be detected. Four bins are used as a rough estimate of the number of bins that are the most affected by objects within the protected area.

If the ratio is ambiguous and does not satisfy this criterion, the signal difference must be used. There, the calculation is no longer so straightforward, and more experimentation is needed. The issue basically is that the addition of the object does not always result in a positive change in the energy, but sometimes the energy decreases compared to the reference. For now, the algorithm does not deal with these conditions and simply finds the largest peak in the area bins, and then subtracts the average value of the bins that are not peaks. In other words, it is a measure of how much the peak stands out from the noise. If this number is greater than 400, the object is detected. The measurement view then shows both of these metrics and can be used to observe and tweak the thresholds. There is also a settings view that has a single checkbox that can pass the result of the algorithm to the device so that the LED also lights up according to evaluation result.

Figure 6.1 Board view of the dToF GUI



Board

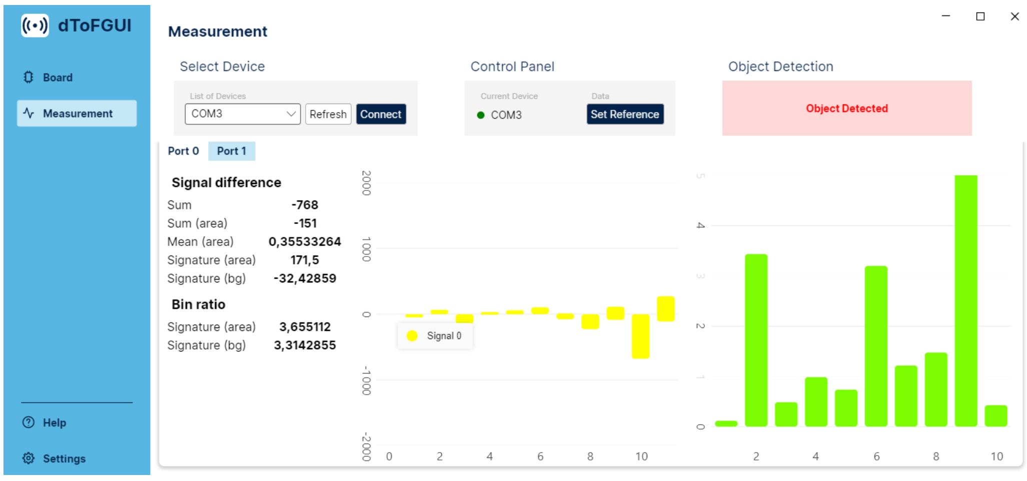
List of Devices: COM8

Current Device: ● COM8

Scene Empty

● Port 0	● Port 1	● Port 2	● Port 3
Distance **0 mm**	Distance **289 mm**	Distance **291 mm**	Distance **0 mm**
Signal **0 kcp/s**	Signal **5062 kcp/s**	Signal **7960 kcp/s**	Signal **0 kcp/s**
Ambient **0 kcp/s**	Ambient **101 kcp/s**	Ambient **78 kcp/s**	Ambient **0 kcp/s**
SPADs **0**	SPADs **210**	SPADs **216**	SPADs **0**
Sigma **0 mm**	Sigma **1 mm**	Sigma **1 mm**	Sigma **0 mm**

Figure 6.2 Measurement view of the dToF GUI



Measurement

List of Devices: COM3

Current Device: ● COM3

Object Detected

Signal difference

Sum	-768
Sum (area)	-151
Mean (area)	0,35533264
Signature (area)	171,5
Signature (bg)	-32,42859

Bin ratio

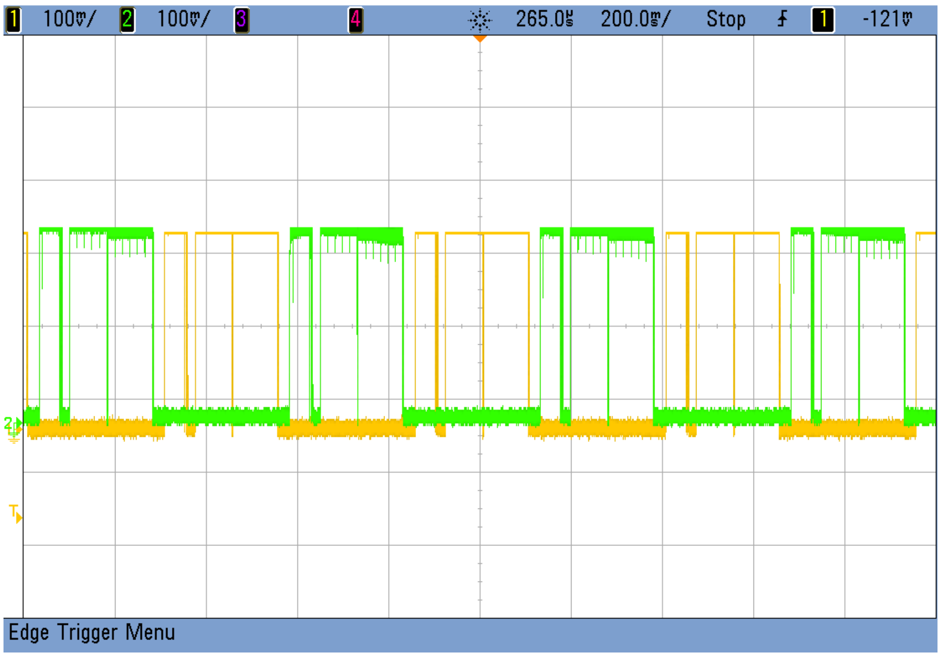
Signature (area)	3,655112
Signature (bg)	3,3142855

Chapter 7

Testing and evaluation

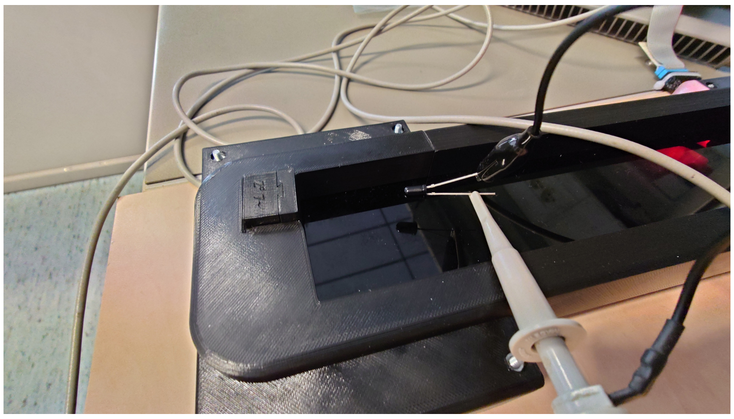
With the full setup working, it first was verified that the sensors do not interfere with each other. This was done with the help of two IR LEDs connected to oscilloscope probes and placed near each sensor. Both halves of the area were isolated and the VCSEL activation times were captured. The results show that there is no crosstalk and that only one of the VCSELs is active at a time, which is an expected behavior. The pictures of the measurement results the setup with IR diode are shown in Figures 7.1 and 7.2, respectively.

Figure 7.1 VCSEL activation sequence



The object detection capabilities of the device were tested with the algorithm described in Section 6.2. Therefore, the calculations were performed externally in the desktop application, because that version of the algorithm was more sofisticated than the simple distance threshold approach implemented in the firmware of the MCU. It also better represented the current direction for future development. The list of objects used during the evaluation was specified in the Chapter 2. It must only be added that the black and white

Figure 7.2 IR diode placed in front of a sensor



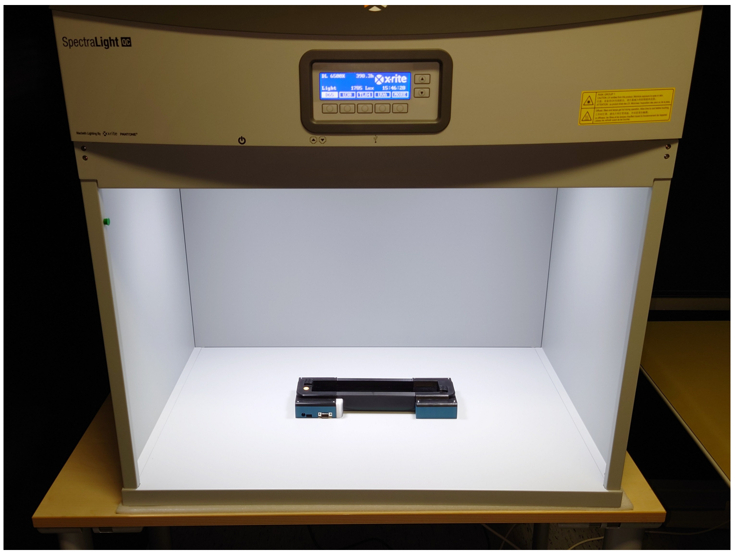
cubes that represent the opposite ends of the reflectivity spectrum were 3D printed and both had dimensions $10 \times 10 \times 15$ mm. In addition, the parking ticket was emulated with a piece of white office paper.

Each item was placed in distinct spots in the area (depending on its size), and it was noted whether the item was reliably detected or not. When detection was not convincing and the state frequently changed to an empty scene, the position was counted as a failure. The experiment was carried out in a professional SpectraLight QC light box with illumination set to TL84 standardized light with a color temperature of 4100 K. This light was set because it has a low amount of IR component, and therefore the effect of ambient light should be minimized. The enclosure was configured so that the center of the sensors would be 5.7 mm from the glass and 11 mm horizontally from the closest sidewall. This was the lowest possible placement achievable with the current sensor holder design. It was chosen because the lower positions are naturally more resistant to false detections of objects behind the scene. Slight 1.2° tilt towards the glass surface was applied for the same reason and the horizontal angle was set to 4° inwards.

The results are shown in Figure 7.5 where each image represents the map of a physical area protected by sensors. The area was divided into 10×10 mm squares according to the size of the black and white test cubes. The cubes were tested in each square, but other larger and less uniform objects usually took more than one square at a time, and their placement was less precise. The credit card could not even fully fit inside the area, so it was tested partially inserted with one side extending upward and also lying on the side walls. Fortunately, it was not necessary to figure out how to convert these positions to squares in the diagram because the credit card and other large objects were always detected regardless of the orientation. Of the base set of objects, only the coin was incorrectly detected in corners that are in a blind spot of the nearest sensor and furthest from the sensor on the opposite side. The reason is that the selected 1 cent is very small and simply does not stand out enough in the signal returned by the nearby wall.

The coverage of the black cube was even worse than that of a coin, even though it was physically larger. Its reflectivity was so low that, in some positions, the sensor did not even capture any difference compared to an empty scene. This was a troubling discovery because if some dark objects do not affect the histogram in any way, then it is simply not possible to detect them using this setup. On the other hand, the coverage of the white cube is almost 100% which emphasizes that the volume of the object is less important than the material and color. Another interesting behavior was that sometimes the histogram bins actually went down with the black cube present. The change was significant and consistent, but it

Figure 7.3 SpectraLight QC Lightbox



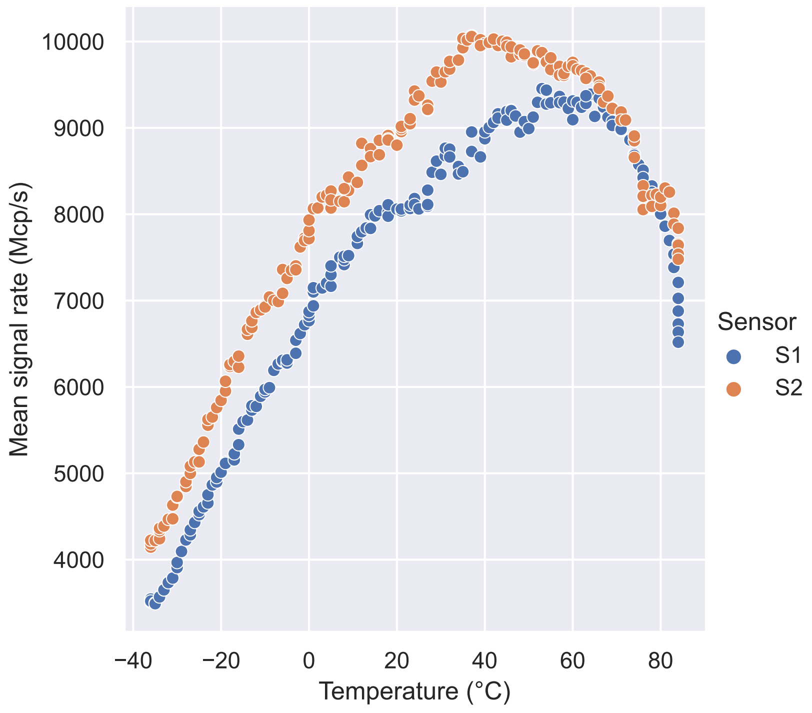
is debatable whether the machine could correctly recognize it. Especially when other, less apparent issues also need to be considered by the algorithm.

One of them is the dust build-up that happens naturally in almost any environment. Unfortunately, even a small amount of dust results in a histogram that could easily be interpreted as belonging to one of the smaller objects. The only assumption that can be made about the dust is that it is uniform and that it should theoretically affect the histogram in some predictable way. However, implementation of any form of dust correction would probably result in an overall worse object detection because the signature of smaller objects shrinks as the dust increases, and any additional filtering is unlikely to make it stand out. There is also a glaring issue with the original premise, because any clean-up of the dust will disturb its uniformity and the system will be very frustrating for the user, since any imperfectly cleaned spot would probably be detected as an object.

The next issue that is not apparent from the results is the detection of fake objects. In some cases, this could be considered a feature because it can provide an early warning to a driver that something is likely to fall somewhere that is not supposed to be. But for this to work, the system must be able to distinguish between this and the situation where the object already violates the rules. However, the problem is that the histogram produced by the sensor is not fine enough and, as a result, the entire protected area is covered only by four bins, with the last bin inconveniently affected by objects behind the frame as well. Maybe, if the dimensions of the protected area were more in line with the spacing of the bins, the boundary could be more easily isolated. The biggest obstacle to this solution will likely be the everpresent multipath reflections.

Another problematic variable is temperature, because in an automotive environment, the system has to be reliable across a wide temperature range. During the experiment in the light box, the temperature was not controlled, but later the system was placed in a temperature chamber and the signal rate was observed. The results in Figure 7.4 show that the sensor output changes drastically with temperature. This is a known fact, and the dToF sensor even has a temperature calibration feature that should solve this problem. According to the manual, temperature only adds a predictable offset to the signal that can be isolated, but this means that temperature must be monitored so that the calibration runs at the right time [30]. Furthermore, it must be tested how the calibration affects the object detection algorithm. Currently, the assumption is that changes caused by a small

Figure 7.4 Signal rate dependency on temperature



temperature drift may be difficult to detect and that the detection sensitivity will have to be reduced to avoid false positives. Even during the object detection test, the histogram of one of the sensors changed over time and a new reference for an empty scene had to be set frequently. This cannot be attributed solely to temperature, but it is definitely one of the likely causes.

Finally, it is apparent from both object detection and temperature testing that each sensor behaves differently under the same conditions. This production spread is significant and was the main reason why the approach was to measure a reference and detect changes to it. It would be possible to calibrate the system during production in a similar matter, but after that it needs to function without any intervention under most conditions. The differences between sensors also make it difficult to take advantage of symmetric arrangments where two or more sensors share a part of the FoV, and the comparison between their outputs could be another useful indicator of object presence.

Figure 7.5 Coverage testing for the histogram algorithm implemented in a GUI application

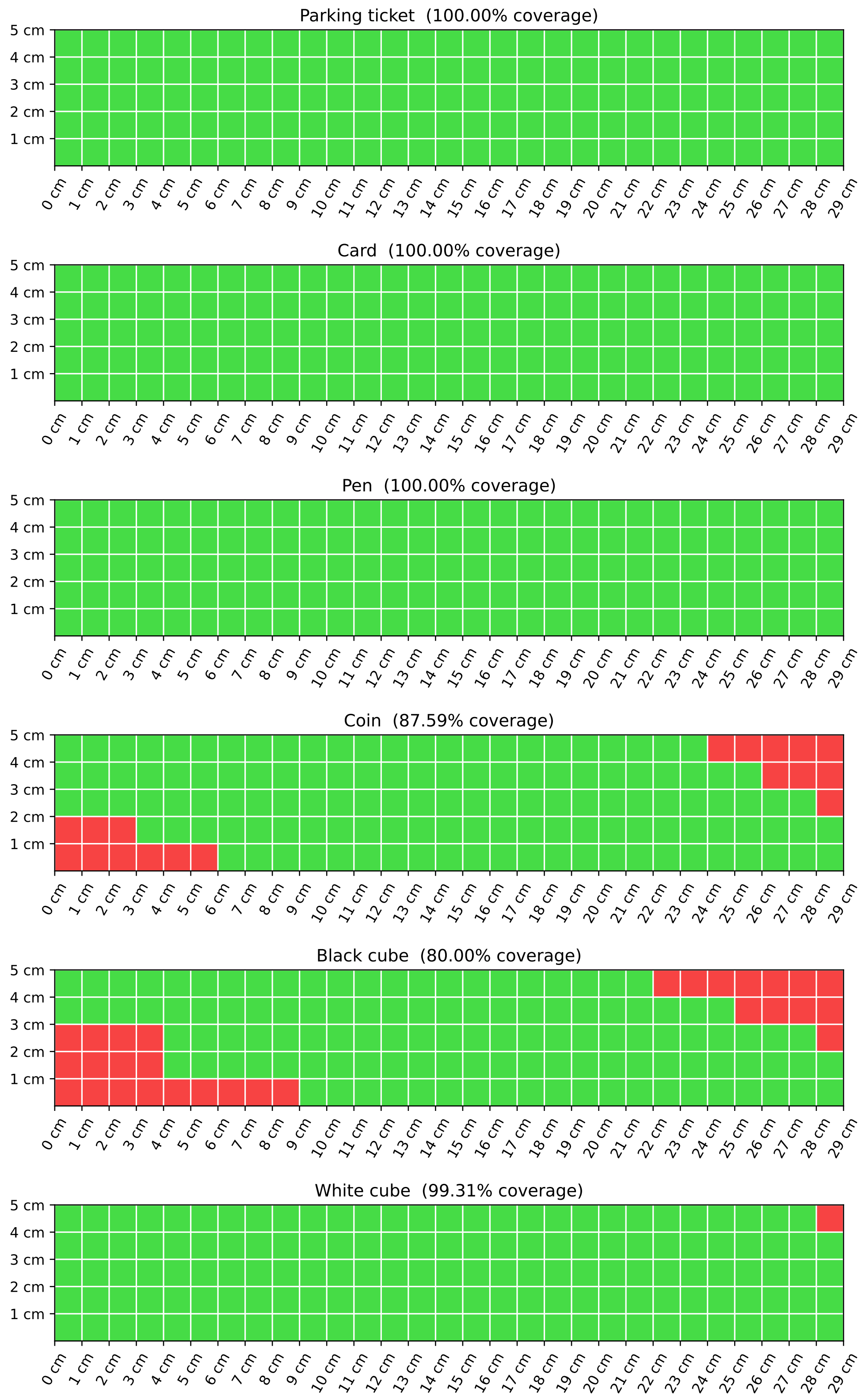
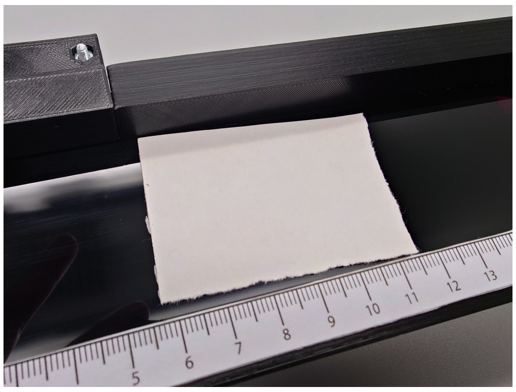
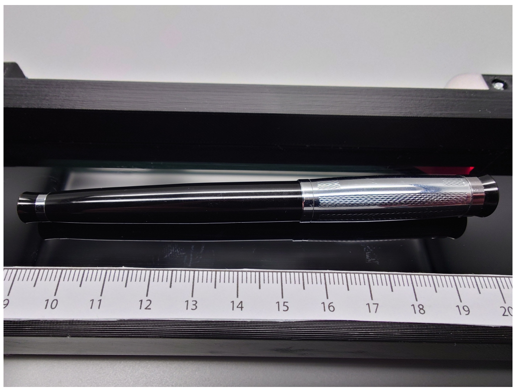


Figure 7.6 Photos of test objects

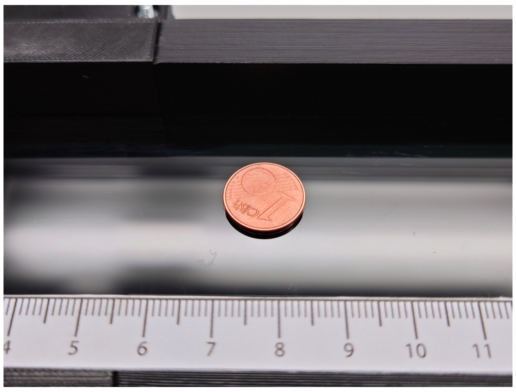


(a) Parking ticket (emulated with a white paper)

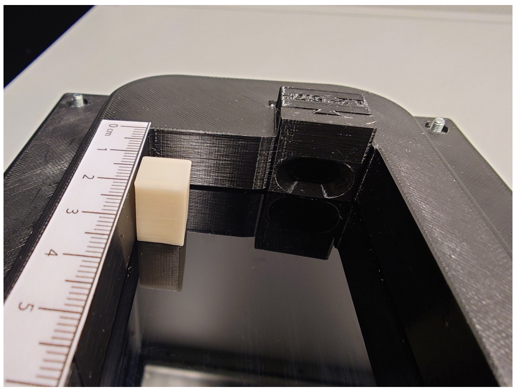
(b) Credit card



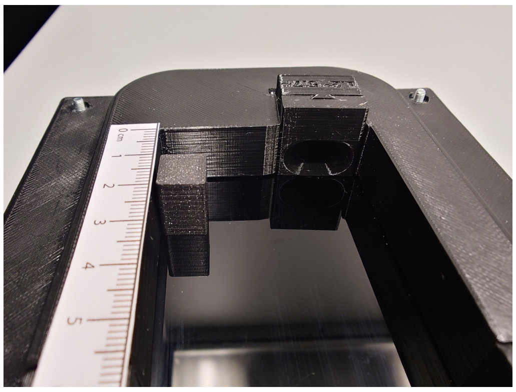
(c) Pen



(d) Coin



(e) White cube



(f) Black cube

Conclusion

The evaluation showed that under laboratory conditions, the setup with two automotive dToF sensors detects the presence of large objects with confidence on the entire surface of the protected area. Unfortunately, the detection algorithm cannot be considered anything more than a development tool because it does not take into account external variables such as temperature and ambient light and also requires frequent manual calibration. These inadequacies were apparent especially when testing small objects with low reflectivity. Large blind spots are formed at the corners of the area that are only covered by the far-away sensor on the opposite side. There, the signature of small objects was very faint, and detecting it was a challenge between the influences of the environment. It is likely that to obtain a system that works reliably and without user intervention, this imperfection cannot be fully eliminated in software with the current design of the area enclosure and this exact sensor.

The shape and dimensions of the protected area were given in the assignment, but it is possible that loosening these requirements in the future could lead to an enclosure that would naturally perform better in the object coverage tests. Moreover, there are still countless unexplored modifications to the current rectangular configuration that might be promising. For example, further testing can be done to assess the influence of different horizontal angles and tilts of the sensor. Sensors could also be moved to form more asymetric arrangements. Fortunately, existing work on the enclosure can be utilized to make these individual changes without much effort, but tackling the sheer number of all possible enclosure configurations and testing scenarios is still going to require a lot of time.

The setup used for the evaluation was built completely from the ground up and will most definitely be useful for future development. It is a standalone device that neatly integrates all sensors, the control board, the cables and the protected area with the glass surface into a 3D printed enclosure. Its construction is robust and it can be moved around without a problem, which makes it great for field testing inside a moving car. The device is fully independent and only needs a compatible power source to function. This can be either a 12 V supply that is present in most cars or a single USB-C cable that also connects to the UART communication interface. Once the system is powered on, the automotive PowerPC MCU automatically starts performing the object detection routine, and the result is clearly indicated in real time by a bright LED. Therefore, communication with some master device is entirely optional.

Both the main control board and the sensor daughter boards were purposely designed for this application. They are almost fully fleshed out, and even the initial prototypes worked reliably and satisfied all expectations. Only the control board contained some minor imperfections, but those would be easily solved in the next version. The most important ICs and components used during assembly are automotive grade. This should be an attractive property for car OEMs because it means that on their side, much less guess work is needed to assess whether this type of system is even something they can integrate into a product.

The current version of the control board firmware implements only the most essential features, and the object detection algorithm simply compares the current distance reported by the sensor to a fixed threshold. However, the functionality can be easily extended thanks to a well-defined command API, and everything is in place for the implementation of an algorithm that also considers the histogram data. The version of such an algorithm was already used during the evaluation, but the computation was delegated to a graphical desktop application that was developed alongside the firmware. At first, the application was only intended to obtain sensor data via USB-C cable and present them in an understandable form to the user. However, once this goal was achieved, it was also practical to utilize it for evaluation as well, because many different histogram processing methods had to be rapidly tested and development on the desktop was faster and more convenient. Of course, the ultimate goal is still to find the best approach and then implement it back into the firmware.

Although the desktop application was originally not required at all, it is probably the component that will see the most further improvements. For example, a special mode for coverage testing is planned to be added. The user will be guided to place objects throughout the protected area and, at each location, the histogram values will be stored. When this procedure is completed, all the data will be exported to a JSON file or any other simple format. Then, if a new idea for the detection algorithm needs testing, this file could be used instead of live data, and it would be immediately clear how the performance changed without doing the whole evaluation routine for every object all over again. This methodology would save a lot of time and would also eliminate inconsistencies caused by changes in the environment between measurements.

In summary, all the objectives outlined in the assignment were fulfilled. The model of the protected area and all the electronics were according to specifications and functioned correctly. The performance of a detection algorithm was tested with a required set of objects, but also other items that more clearly illustrate the typical behavior of dToF sensors. Overall, the evaluation succesfully identified what needs to be solved to bridge the gap between the prototype and a final product.

Appendix A
PCB drawings

Figure A.1 dToF Control Board 1.1 - dimensions

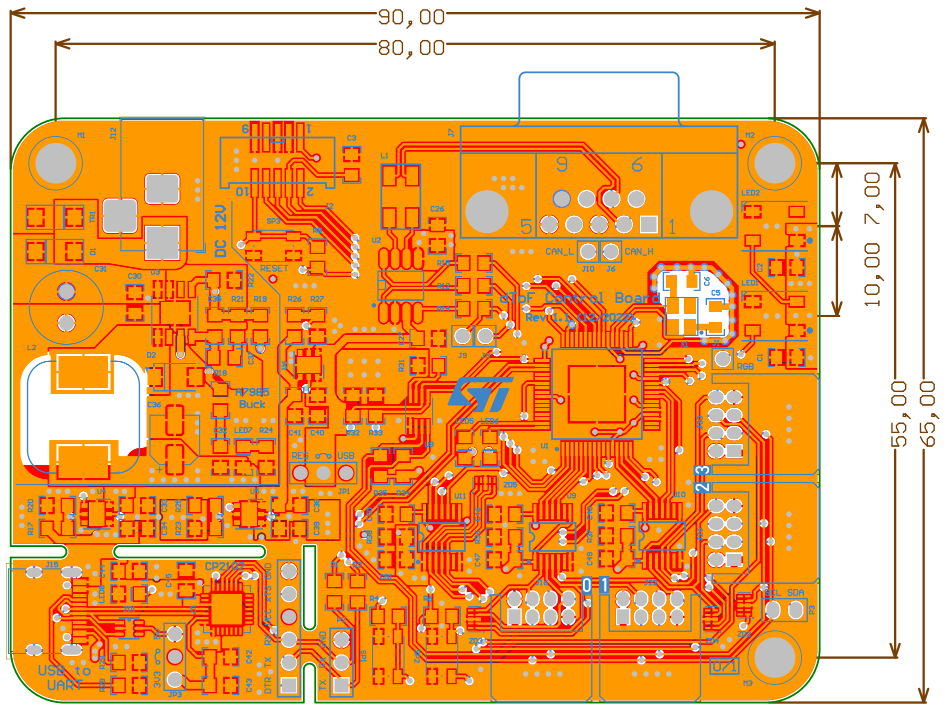


Figure A.2 Sensor Board - dimensions

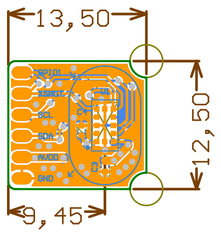


Figure A.3 dToF Control Board 1.1 - Top layer

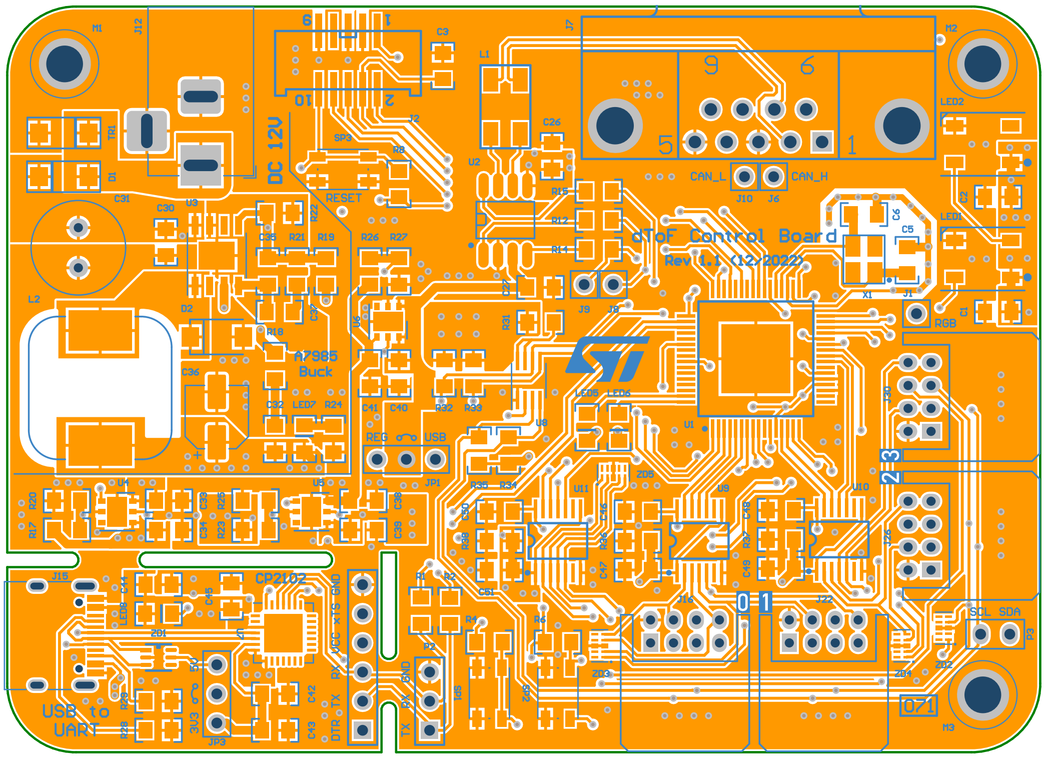


Figure A.4 dToF Control Board 1.1 - Bottom layer

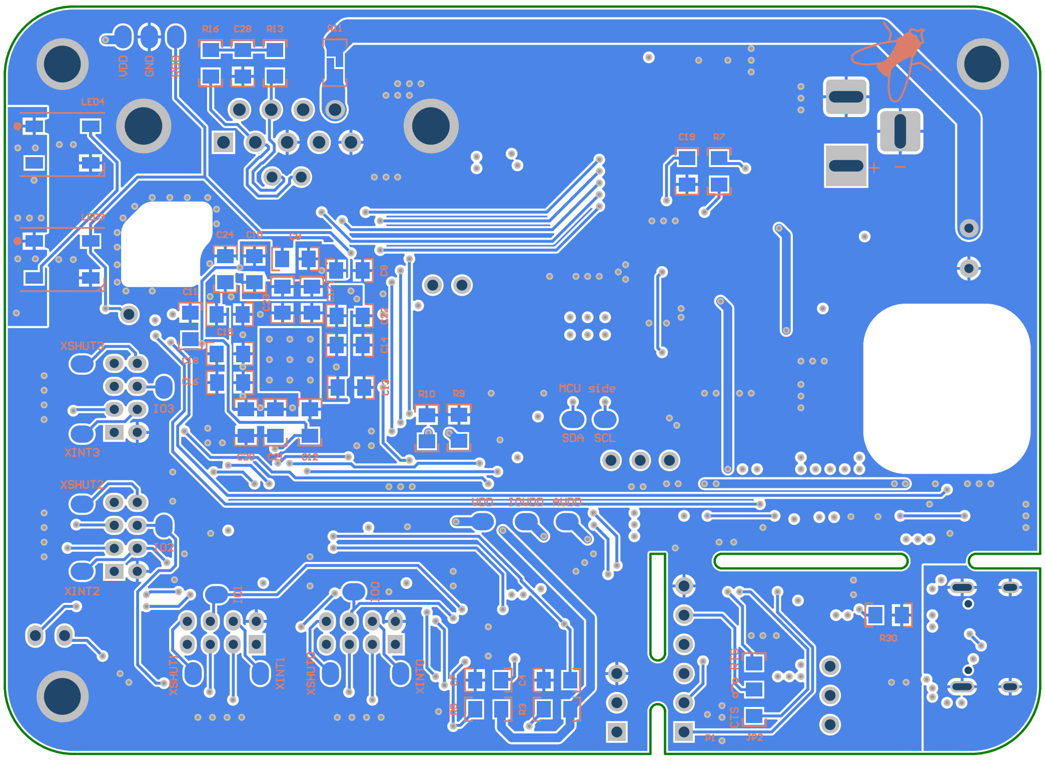


Figure A.5 dToF Control Board 1.1 - Inner power layer

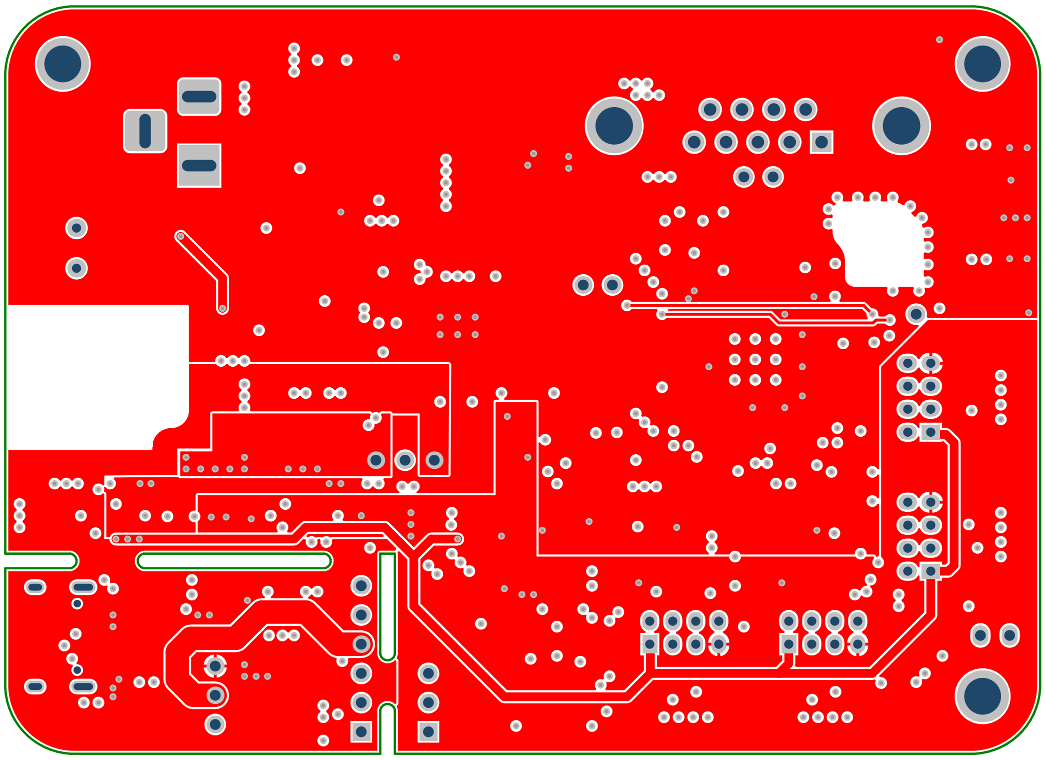
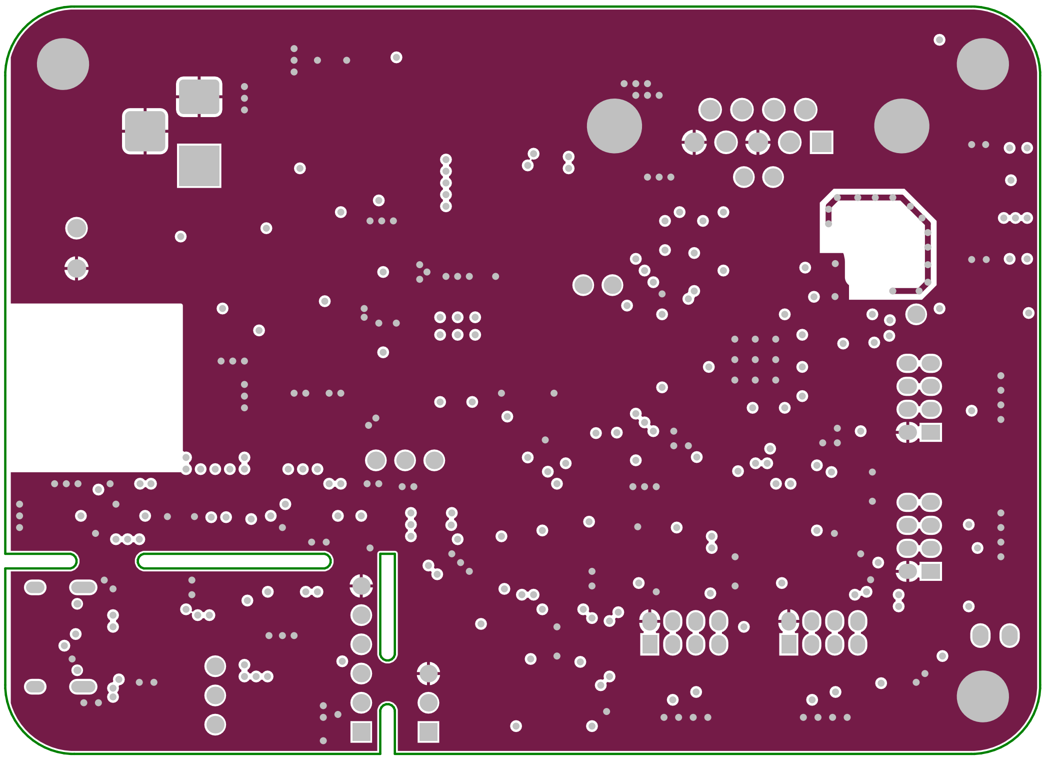


Figure A.6 dToF Control Board 1.1 - Inner ground layer



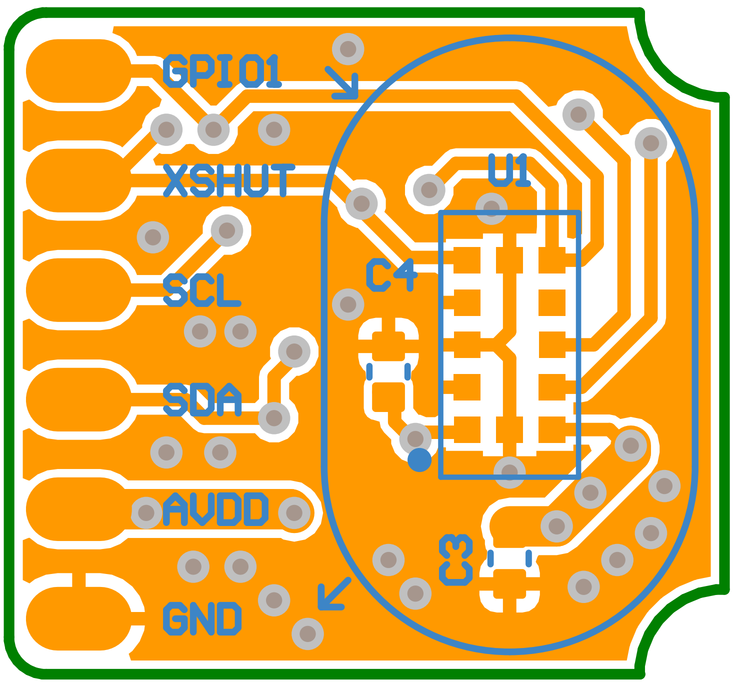


Figure A.7 Sensor Board - Top layer

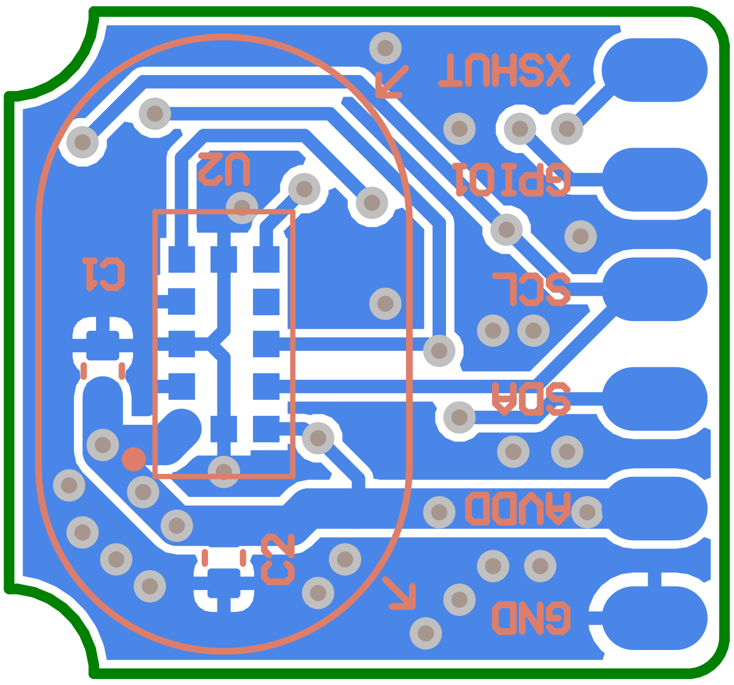
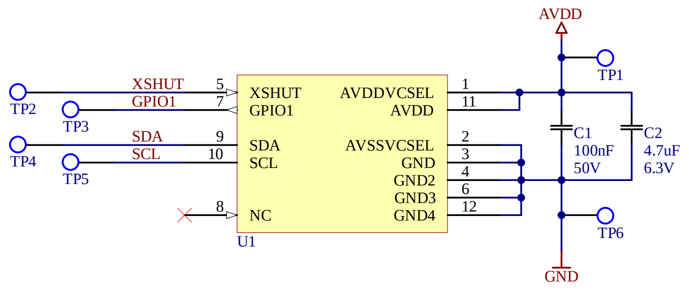
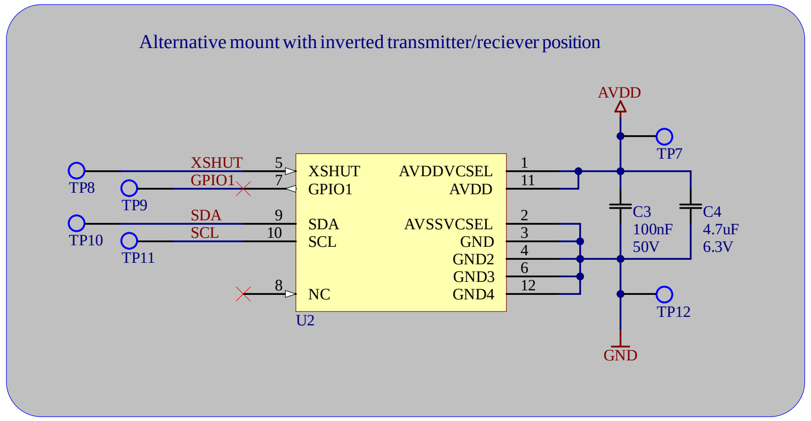


Figure A.8 Sensor Board - Bottom layer

Appendix B

Schematics





STMicroelectronics assumes no responsibility for the consequences of the use of this information

Title: **VL53LA Reversible Board**

Size: A4 Author: EMEA - ADG - M&A Revision: 1.0

Date: 01.01.2023 Time: 22:03:34 Sheet 1 of 1

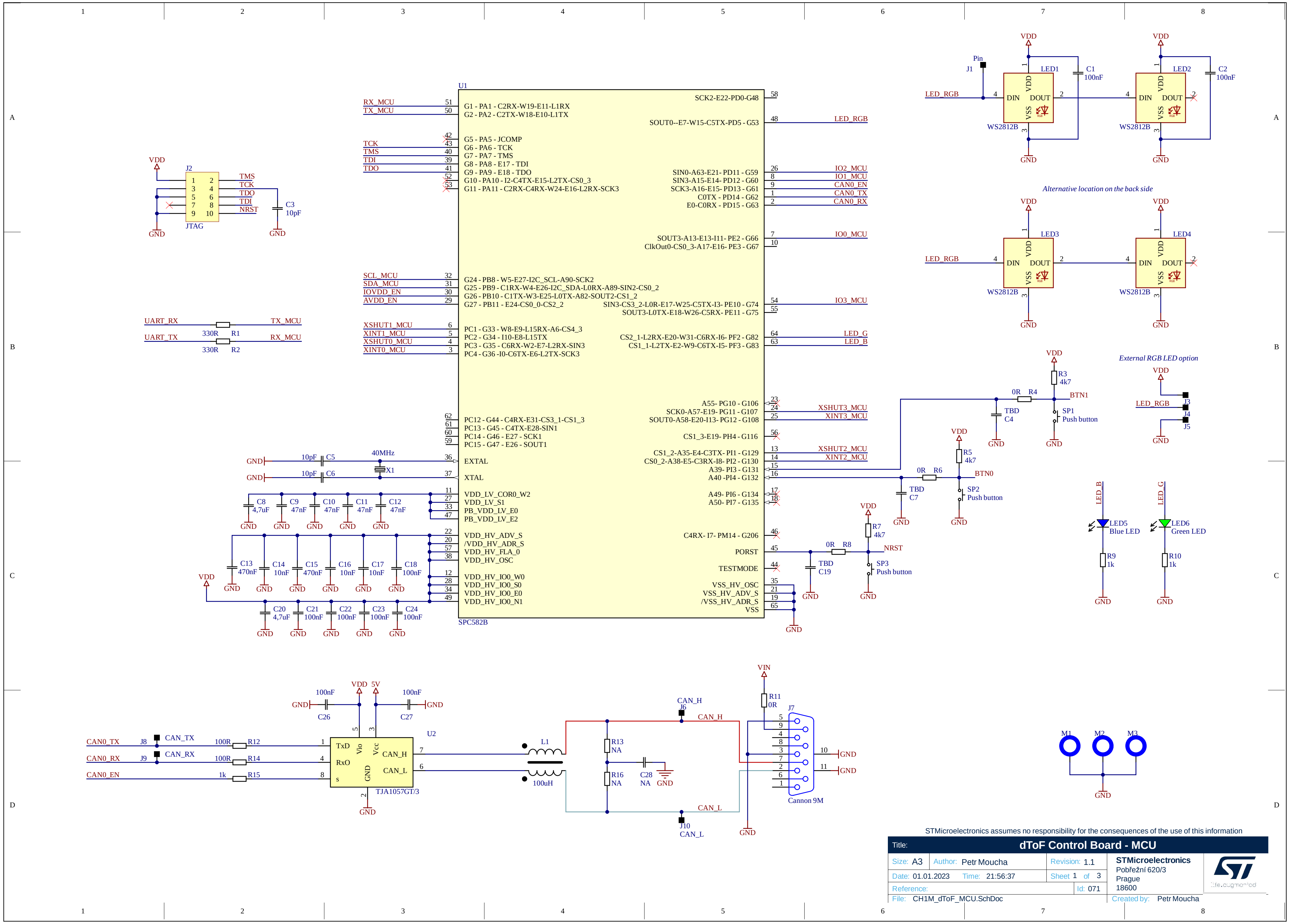
Reference: Id:

File: VL53LA_ReversibleBoard.SchDoc

STMicroelectronics
Pobrezni 3
186 00 Prague 8
Czech Republic

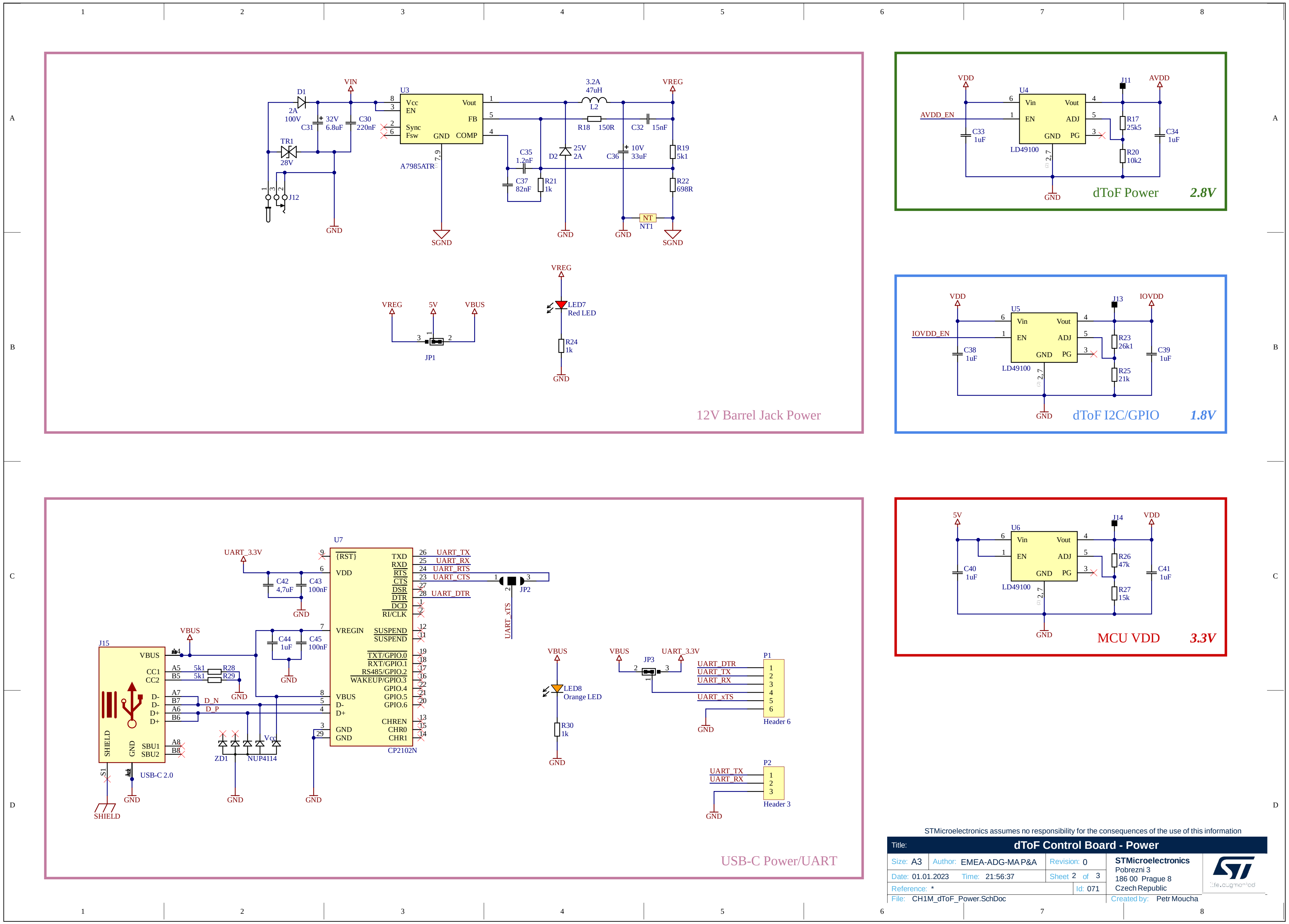
Created by: Petr Moucha

life.augmented



STMicroelectronics assumes no responsibility for the consequences of the use of this information

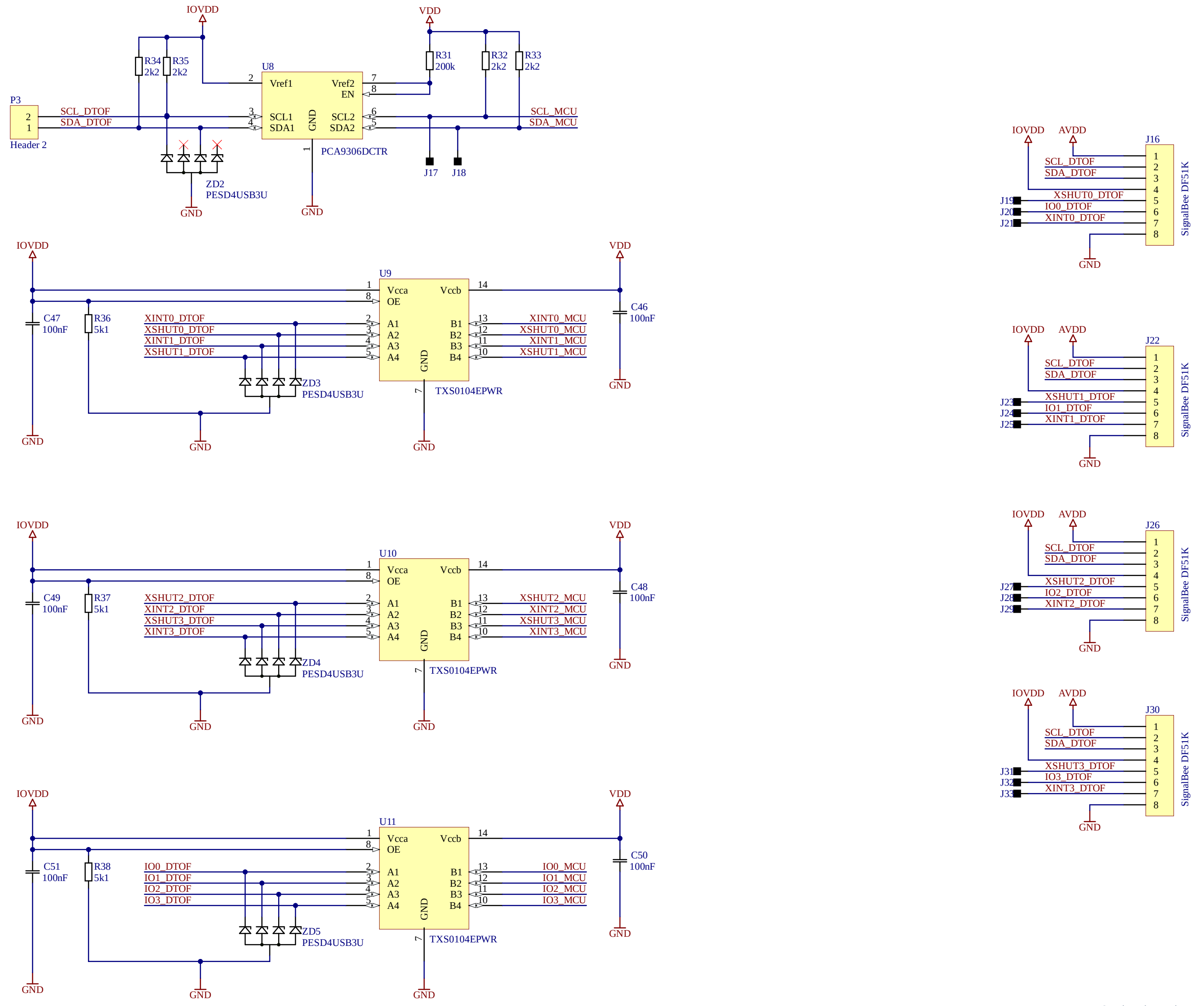
Title:	dToF Control Board - MCU			
Size: A3	Author: Petr Moucha	Revision: 1.1	STMicroelectronics Pobřežní 620/3 Prague 18600	
Date: 01.01.2023	Time: 21:56:37	Sheet 1 of 3		
Reference:		Id: 071		
File: CH1M_dToF_MCU.SchDoc			Created by: Petr Moucha	



STMicroelectronics assumes no responsibility for the consequences of the use of this information

Title:	dToF Control Board - Power
Size: A3	Author: EMEA-ADG-MA P&A
Revision: 0	
Date: 01.01.2023	Time: 21:56:37
Sheet 2 of 3	
Reference: *	Id: 071
File: CH1M_dToF_Power.SchDoc	Created by: Petr Moucha

STMicroelectronics
Pobrezni 3
186 00 Prague 8
Czech Republic



STMicroelectronics assumes no responsibility for the consequences of the use of this information

Title: dToF Control Board - dToF sensors			
Size: A3	Author: EMEA - ADG - M&A	Revision: 0	STMicroelectronics Pobrezni 3 186 00 Prague 8 Czech Republic
Date: 01.01.2023	Time: 21:56:37	Sheet 3 of 3	
Reference: *		Id: 071	
File: CH1M_dToF_Sensors.SchDoc			Created by: Petr Moucha

Bibliography

1. OHTA, Jun. *Smart CMOS Image Sensors and Applications*. Milton: CRC Press, 2020. ISBN 9781498797344.
2. EIGEN, David; PUHRSCH, Christian; FERGUS, Rob. Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Montreal, Canada: MIT Press, 2014, pp. 2366–2374. NIPS'14.
3. GIANCOLA, Silvio; VALENTI, Matteo; SALA, Remo. *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies*. Cham: Springer International Publishing, 2018. ISBN 9783319917610.
4. ZANUTTIGH, Pietro; MARIN, Giulio; DAL MUTTO, Carlo; DOMINIO, Fabio; MINTO, Ludovico; CORTELAZZO, Guido Maria. Time-of-flight and structured light depth cameras. *Technology and Applications*. 2016, pp. 978–3.
5. MICHALZIK, Rainer. VCSEL fundamentals. In: *VCSELs*. Springer, 2013, pp. 19–75.
6. TELEDYNE PRINCETON INSTRUMENTS. *Quantum Efficiency Educational Notes* [online]. [visited on 2023-01-02]. Available from: `https://www.princetoninstruments.com/learn/camera-fundamentals/quantum-efficiency`.
7. WICHMANN, Matthias; KAMIL, Mustafa; FREDERIKSEN, Annette; KOTZUR, Sebastian; SCHERL, Michael. Long-Term Investigations of Weather Influence on Direct Time-of-Flight Lidar at 905nm. *IEEE Sensors Journal*. 2022, vol. 22, no. 3, pp. 2024–2036. Available from DOI: `10.1109/JSEN.2021.3133658`.
8. WIKIMEDIA COMMONS, the free media repository (ed.). *Solar spectrum* [Image] [online]. 2022-02-09. [visited on 2022-12-06]. Available from: `https://commons.wikimedia.org/w/index.php?title=File:Solar_spectrum_en.svg&oldid=628645275`.
9. BAMJI, Cyrus; GODBAZ, John; OH, Minseok; MEHTA, Swati; PAYNE, Andrew; ORTIZ, Sergio; NAGARAJA, Satyadev; PERRY, Travis; THOMPSON, Barry. A Review of Indirect Time-of-Flight Technologies. *IEEE Transactions on Electron Devices*. 2022, vol. 69, no. 6, pp. 2779–2793. Available from DOI: `10.1109/TED.2022.3145762`.
10. MANDA, S.; MATSUMOTO, R.; SAITO, S.; MARUYAMA, S.; MINARI, H.; HIRANO, T.; TAKACHI, T.; FUJII, N.; YAMAMOTO, Y.; ZAIZEN, Y.; HIRANO, T.; IWAMOTO, H. High-definition Visible-SWIR InGaAs Image Sensor using Cu-Cu Bonding of III-V to Silicon Wafer. In: *2019 IEEE International Electron Devices Meeting (IEDM)*. 2019, pp. 16.7.1–16.7.4. Available from DOI: `10.1109/IEDM19573.2019.8993432`.

11. KOSTAMOVAARA, Juha; JAHROMI, Sahba; HALLMAN, Lauri; DUAN, Guoyong; JANSSON, Jussi-Pekka; KERÄNEN, Pekka. Solid-State Pulsed Time-of-Flight 3-D Range Imaging Using CMOS SPAD Focal Plane Array Receiver and Block-Based Illumination Techniques. *IEEE Photonics Journal*. 2022, vol. 14, no. 2, pp. 1–11. Available from DOI: `10.1109/JPHOT.2022.3153487`.
12. PADMANABHAN, Preethi; ZHANG, Chao; CHARBON, Edoardo. Modeling and Analysis of a Direct Time-of-Flight Sensor Architecture for LiDAR Applications. *Sensors*. 2019, vol. 19, no. 24. ISSN 1424-8220. Available from DOI: `10.3390/s19245464`.
13. VILLA, Federica; SEVERINI, Fabio; MADONINI, Francesca; ZAPPA, Franco. SPADs and SiPMs Arrays for Long-Range High-Speed Light Detection and Ranging (LiDAR). *Sensors*. 2021, vol. 21, no. 11. ISSN 1424-8220. Available from DOI: `10.3390/s21113839`.
14. RAJ, Thinal; HASHIM, Fazida Hanim; HUDDIN, Aqilah Baseri; IBRAHIM, Mohd Faisal; HUSSAIN, Aini. A Survey on LiDAR Scanning Mechanisms. *Electronics*. 2020, vol. 9, no. 5. ISSN 2079-9292. Available from DOI: `10.3390/electronics9050741`.
15. ACERBI, Fabio; MORENO-GARCIA, Manuel; KOKLÜ, Gözen; GANCARZ, Radoslaw Marcin; BÜTTGEN, Bernhard; BIBER, Alice; FURRER, Daniel; STOPPA, David. Optimization of Pinned Photodiode Pixels for High-Speed Time of Flight Applications. *IEEE Journal of the Electron Devices Society*. 2018, vol. 6, pp. 365–375. Available from DOI: `10.1109/JEDS.2018.2807918`.
16. SHAHANDASHTI, Peyman F.; LÓPEZ, P.; BREA, V.M.; GARCÍA-LESTA, D.; HEREDIA-CONDE, Miguel. A 2-Tap Macro-Pixel-Based Indirect ToF CMOS Image Sensor for Multi-Frequency Demodulation. In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2022, pp. 1430–1434. Available from DOI: `10.1109/ISCAS48785.2022.9937277`.
17. GUPTA, Mohit; NAYAR, Shree K.; HULLIN, Matthias B.; MARTIN, Jaime. Phasor Imaging: A Generalization of Correlation-Based Time-of-Flight Imaging. *ACM Trans. Graph.* 2015, vol. 34, no. 5. ISSN 0730-0301. Available from DOI: `10.1145/2735702`.
18. PIRON, François; MORRISON, Daniel; YUCE, Mehmet Rasit; REDOUTÉ, Jean-Michel. A Review of Single-Photon Avalanche Diode Time-of-Flight Imaging Sensor Arrays. *IEEE Sensors Journal*. 2021, vol. 21, no. 11, pp. 12654–12666. Available from DOI: `10.1109/JSEN.2020.3039362`.
19. SEO, Hyeongseok; YOON, Heesun; KIM, Dongkyu; KIM, Jungwoo; KIM, Seong-Jin; CHUN, Jung-Hoon; CHOI, Jaehyuk. Direct TOF Scanning LiDAR Sensor With Two-Step Multievent Histogramming TDC and Embedded Interference Filter. *IEEE Journal of Solid-State Circuits*. 2021, vol. 56, no. 4, pp. 1022–1035. Available from DOI: `10.1109/JSSC.2020.3048074`.
20. ZHANG, Long; CHITNIS, Danial; CHUN, Hyunchae; RAJBHANDARI, Sujan; FAULKNER, Grahame; O'BRIEN, Dominic; COLLINS, Steve. A Comparison of APD- and SPAD-Based Receivers for Visible Light Communications. *Journal of Lightwave Technology*. 2018, vol. 36, no. 12, pp. 2435–2442. Available from DOI: `10.1109/JLT.2018.2811180`.
21. CHARBON, Edoardo; BRUSCHINI, Claudio; LEE, Myung-Jae. 3D-Stacked CMOS SPAD Image Sensors: Technology and Applications. In: *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2018, pp. 1–4. Available from DOI: `10.1109/ICECS.2018.8617983`.
22. GYONGY, Istvan; DUTTON, Neale A. W.; HENDERSON, Robert K. Direct Time-of-Flight Single-Photon Imaging. *IEEE Transactions on Electron Devices*. 2022, vol. 69, no. 6, pp. 2794–2805. Available from DOI: `10.1109/TED.2021.3131430`.

23. BUCKBEE, Kasper; DUTTON, Neale A. W.; HENDERSON, Robert K. An Indirect Time-of-Flight SPAD Pixel With Dynamic Comparator Reuse for a Single-Slope ADC. *IEEE Solid-State Circuits Letters*. 2022, vol. 5, pp. 158–161. Available from DOI: `10.1109/LSSC.2022.3179579`.

24. DUTTON, Neale AW; GYONGY, Istvan; PARMESAN, Luca; GNECCHI, Salvatore; CALDER, Neil; RAE, Bruce R; PELLEGRINI, Sara; GRANT, Lindsay A; HENDERSON, Robert K. A SPAD-based QVGA image sensor for single-photon counting and quanta imaging. *IEEE Transactions on Electron Devices*. 2015, vol. 63, no. 1, pp. 189–196.

25. XUE, Ruikai; KANG, Yan; ZHANG, Tongyi; LI, Lifei; ZHAO, Wei. Sub-Pixel Scanning High-Resolution Panoramic 3D Imaging Based on a SPAD Array. *IEEE Photonics Journal*. 2021, vol. 13, no. 4, pp. 1–6. Available from DOI: `10.1109/JPHOT.2021.3103817`.

26. INTERMITE, Giuseppe; MCCARTHY, Aongus; WARBURTON, Ryan E.; REN, Ximing; VILLA, Federica; LUSSANA, Rudi; WADDIE, Andrew J.; TAGHIZADEH, Mohammad R.; TOSI, Alberto; ZAPPA, Franco; BULLER, Gerald S. Fill-factor improvement of Si CMOS single-photon avalanche diode detector arrays by integration of diffractive microlens arrays. *Opt. Express*. 2015, vol. 23, no. 26, pp. 33777–33791. Available from DOI: `10.1364/OE.23.033777`.

27. VILLA, Federica; LUSSANA, Rudi; TAMBORINI, Davide; TOSI, Alberto; ZAPPA, Franco. High-Fill-Factor 60 × 1 SPAD Array With 60 Subnanosecond Integrated TDCs. *IEEE Photonics Technology Letters*. 2015, vol. 27, no. 12, pp. 1261–1264. Available from DOI: `10.1109/LPT.2015.2416192`.

28. STMICROELECTRONICS. *Time-of-Flight sensor with extended range measurement* [DS13805]. STMicroelectronics, 2022-11-24. Version 4. Available also from: `https://www.st.com/resource/en/datasheet/vl53l4cx.pdf`.

29. STMICROELECTRONICS. *Time-of-Flight high accuracy proximity sensor* [DS13812]. STMicroelectronics, 2022-06-29. Version 5. Available also from: `https://www.st.com/resource/en/datasheet/vl53l4cd.pdf`.

30. STMICROELECTRONICS. *A guide to using the VL53L4CD ultra lite driver (ULD)* [UM2931]. STMicroelectronics, 2022-08-16. Version 2. Available also from: `https://www.st.com/resource/en/user_manual/um2931-a-guide-to-using-the-vl53l4cd-ultra-lite-driver-uld-stmicroelectronics.pdf`.

31. STMICROELECTRONICS. *SPC58 2B Line - 32 bit Power Architecture automotive MCU* [DS11597]. STMicroelectronics, 2020-12-01. Version 4. Available also from: `https://www.st.com/resource/en/datasheet/spc582b60e1.pdf`.

32. SILICON LABORATORIES INC. *CP2102/9 – Single-chip USB-to-UART bridge*. Silicon Laboratories Inc., 2017-01-20. Version 1.8. Available also from: `https://www.silabs.com/documents/public/data-sheets/CP2102-9.pdf`.

33. STMICROELECTRONICS. *LD49100 – Automotive-grade, 1 A, low quiescent current, low-noise regulator with soft-start* [DS13115]. STMicroelectronics, 2021-02-18. Version 4. Available also from: `https://www.st.com/resource/en/datasheet/ld49100.pdf`.

34. STMICROELECTRONICS. *Getting started with the X-NUCLEO-53L4A1 expansion board for STM32 Nucleo based on the VL53L4CD* [UM2972]. STMicroelectronics, 2022-06-16. Version 4. Available also from: `https://www.st.com/resource/en/user_manual/um2972-getting-started-with-the-xnucleo53l4a1-expansion-board-for-stm32-nucleo-based-on-the-vl53l4cd-stmicroelectronics.pdf`.

35. TEXAS INSTRUMENTS. *TXS0104E 4-Bit Bidirectional Voltage-Level Translator for Open-Drain Push-Pull A datasheet*. Texas Instruments, 2020-10-09. Available also from: `https://www.ti.com/lit/gpn/txs0104e`.

36. STMICROELECTRONICS. *SPC58xx hardware design guideline* [AN4880]. STMicroelectronics, 2022-07-07. Version 9. Available also from: `https://www.st.com/resource/en/application_note/an4880-spc58xx-hardware-design-guideline-stmicroelectronics.pdf`.

37. PALAND, Marco. *A printf / sprintf Implementation for Embedded Systems* [GitHub repository]. GitHub, 2019. Version 4.0. Available also from: `https://github.com/mpaland/printf`.

Attached media contents

```
doc ............................................ generated documentation
│ ├── fw ....................................... firmware documentation
│ └── pcb .............. PDF documentation and fabrication outputs
├── src. ................................................. source codes
│ ├── cad ..................................... enclosure CAD files
│ ├── fw ................................ dToF Control Board firmware
│ ├── gui ......................................... GUI application
│ └── pcb ...................................... Altium design files
├── thesis ............................. thesis source code in LaTeX
└── thesis.pdf ................................... thesis text in PDF
```