Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CENTER FOR PROJECT-BASED LEARNING

Department of Information Technology and Electrical Engineering

Spring Semester 2026

Linear-Exponential Algorithm-based Open Source Dive Computer

Bachelor Thesis

Sebastian Thomas
tsebastian@ethz.ch

June 2026

Supervisor: Dr. Tommaso Polonelli, tommaso.polonelli@pbl.ee.ethz.ch
Professor: Prof. Dr. Luca Benini, lbenini@iis.ee.ethz.ch

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Tommaso Polonelli, for his guidance and support throughout the different phases of the thesis, as well as for his continued feedback.

I also thank Prof. Dr. Luca Benini for serving as examiner and for coordinating this thesis and the corresponding assessment together with my supervisor.

I would like to acknowledge the Center for Project-Based Learning for providing a supportive environment and laboratory with the required professional tools, as well as the staff for their assistance, and my fellow students for all the insightful comments and discussions.

Furthermore, I am grateful to Prof. Simon Mitchell for encouraging me to start exploring the topic of variable-rate algorithms for dive computers.

I would also like to thank my recent diving instructors and friends with whom I have trained with for their support, discussions on dive computer functionality, algorithms and technology, and the ideas they contributed.

Finally, I thank my friends and flatmates for their support and for creating a positive environment during the duration of my studies and especially during the completion of this thesis.

Abstract

Dive Computers are now standard equipment for scuba diving, continuously tracking inert gas loading, the corresponding decompression obligation, and oxygen exposure. They also allow for more detailed dive logging by allowing users to view and export the recorded depth over time. As safety-critical, battery-powered life-support devices, commercial and military dive computers are frequently based on proprietary hardware and algorithms, limiting customization options, transparency, and reproducibility. Previous work has shown that adaptive or variable-rate scheduling can substantially reduce computational effort and energy consumption in low-power embedded systems. Among existing decompression models, the Thalmann-inspired linear-exponential approach has been associated with reduced decompression sickness risk on deeper dives but is rarely implemented, especially in publicly available dive computer software.

This work presents an extendable open-source hardware and software basis for the development of dive computers that is feasible to build and extend for non-professional audiences, while remaining compact enough to be worn during a scuba dive. The diving-related algorithms for tracking inert gas loading, decompression, and oxygen toxicity are implemented in a separate open-source module. Multiple algorithms are implemented and used for the variable rate scheduling of different tasks in dive mode. These algorithms show a reduction in executions from 1.17× to 6× for decompression calculations without significant difference in the resulting simulated dive profiles compared to the fixed-rate baseline, while the oxygen toxicity calculation does not benefit from the current implementation of variable rate scheduling. Flash logging to accurately reconstruct a dive profile has been reduced by a factor of 3.6× to 4.1× compared to the fixed-rate baseline while incurring no measurable accuracy loss on the simulated profiles. The usage of the Thalmann-inspired linear-exponential algorithm performs as expected, recommending slightly deeper stops on deep profiles without incurring significant computational overhead.

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor. For a detailed version of the declaration of originality, please refer to Appendix B

Sebastian Thomas,
Zurich, June 2026

Contents

List of Acronyms **ix**

1. Introduction **1**
1.1. Motivation . . . 1
1.2. Objective . . . 2
1.2.1. Research Questions . . . 2
1.2.2. Outline . . . 2

2. Background **3**
2.1. Embedded Systems . . . 3
2.1.1. Low Power Consumption in Battery-powered devices . . . 3
2.1.2. Variable-rate scheduling . . . 4
2.2. Diving and Humans in Hyperbaric Environments . . . 5
2.2.1. Inert gas loading and Decompression . . . 5
2.2.2. Bühlmann ZHL-16 Decompression Algorithms . . . 6
2.2.3. Linear-Exponential Decompression Algorithm . . . 7
2.2.4. Oxygen at hyperbaric pressure . . . 9
2.2.5. Further considerations . . . 10

3. Related Work **11**
3.1. Diving Algorithms and Dive Computer Hardware . . . 11
3.2. Low-Power Algorithms and Variable-Rate Research . . . 11

4. Implementation in Hardware, Firmware and Algorithms **14**
4.1. System Requirements . . . 14
4.2. Hardware . . . 15
4.2.1. Component Definition . . . 15
4.3. PCB Design . . . 16
4.4. Firmware . . . 17
4.4.1. RTOS . . . 17

4.4.2. Hardware Abstraction Layer (HAL) . . . 18
4.5. Diving Algorithms . . . 19
4.5.1. Exponential Decompression Algorithm . . . 19
4.5.2. Linear-Exponential Decompression Algorithm . . . 19
4.5.3. Oxygen Toxicity (O2 Tox) calculation . . . 19
4.6. Scheduling of Diving Algorithms . . . 19
4.6.1. FixedRateAlgorithm . . . 20
4.6.2. AimdRateAlgorithm . . . 20
4.6.3. DiffAimdRateAlgorithm . . . 20
4.6.4. DynamicAimdRateAlgorithm . . . 21
4.7. Data Logging to Flash . . . 22

5. Results **23**
5.1. Hardware Verification and Testing . . . 23
5.1.1. Power Consumption . . . 23
5.2. Experimental Setup . . . 24
5.3. O2 Tox Algorithm results . . . 25
5.4. Exponential & Linear-Exponential Decompression Algorithm results . . . 28
5.5. Flash Log Rate results . . . 31

6. Discussion **33**
6.1. Open-source Dive Computer . . . 33
6.2. Impact of variable rate algorithms for Decompression and O2 Tox scheduling 35
6.3. Impact of variable rate algorithm for flash logging . . . 35
6.4. Linear-exponential algorithm . . . 36
6.5. Limitations . . . 36
6.5.1. Data Collection . . . 36
6.5.2. Software and Algorithms . . . 36

7. Conclusion and Future Work **38**

A. Task Description **40**

B. Declaration of Originality **46**

C. File Structure **47**

D. Additional Figures: Printed Circuit Board (PCB) Design **48**

E. Additional Results **50**

F. Additional Tables: Flash Log structure **54**

Glossary **56**

List of Figures

D.1. STDC PCB Layer 1 (Primary Routing & Component Placement Layer) . 49
D.2. STDC PCB Layer 3 (Secondary Routing Layer) 49

E.1. Simulated Depth of 20m profile 50
E.2. Simulated Depth of 50m profile using the lin-exp decompression algorithm 51
E.3. Simulated Depth of 90m profile using the lin-exp decompression algorithm 51
E.4. Simulated 20 m dive profile: Difference between Flash log and actual depth using the variable rate algorithms 52
E.5. Simulated 50 m dive profile: Difference between Flash log and actual depth using the variable rate algorithms 52
E.6. Simulated 90 m dive profile: Difference between Flash log and actual depth using the variable rate algorithms 53

List of Tables

5.1. O_2 toxicity computation time across profiles and configurations 26
5.2. O_2 toxicity rate-algorithm execution time across profiles and configurations 27
5.3. O2Tox decision summary across profiles and configurations 27
5.4. Decompression rate-algorithm execution time across profiles and configurations 28
5.5. Decompression schedule computation time across profiles and configurations 29
5.6. Decompression decision summary across profiles and configurations 30
5.7. Flash log rate-algorithm execution time across profiles and configurations 31
5.8. Flash log depth accuracy: signed error between logged and interpolated dive profile depth 32

E.1. Flash log rate decision summary across profiles and configurations 53

F.1. Flash Log: Control Data Block 54
F.2. Flash Log: Log Data Point Metadata 55
F.3. Flash Log: Log Data Point Data 55

List of Acronyms

AIMD . . . . . Additive Increase, Multiplicative Decrease

BGA . . . . . Ball Grid Array
BLE . . . . . Bluetooth Low-Energy

CC . . . . . Closed Circuit
CDC . . . . . Communication Device Class
CNS . . . . . Central Nervous System

DCS . . . . . Decompression Sickness
DFN . . . . . Dual Flat No-Lead
DFU . . . . . Device Firmware Upgrade
DFVS . . . . . Dynamic Voltage / Frequency Scaling

FHe . . . . . Fraction of Helium (in a gas mix)
FN2 . . . . . Fraction of Nitrogen (in a gas mix)
FO2 . . . . . Fraction of Oxygen (in a gas mix)

GF High . . . . Gradient Factor High
GF Low . . . . . Gradient Factor Low
GF99 . . . . . Gradient Factor 99
GPIO . . . . . General Purpose Input Output

List of Acronyms

HAL . . . . . .Hardware Abstraction Layer

HPNS . . . . . .High Pressure Neurological Syndrome

I/O . . . . . .Input-Output

I2C . . . . . .Inter-Integrated Circuit

IC . . . . . .Integrated Circuit

ICD . . . . . .Isobaric Counterdiffusion

LQFP . . . . . .Low Profile Quad Flat Package

LSE . . . . . .Low Speed External, MCU connection for an external oscillator

LSI . . . . . .Low Speed Internal, real-time oscillator in STM32 MCUs

MCU . . . . . .Microcontroller Unit

MSC . . . . . .Mass Storage Class

N_2 . . . . . .Nitrogen

NDL . . . . . .No decompression limit

O2 Tox . . . . .Oxygen Toxicity

O_2 . . . . . .Oxygen

OC . . . . . .Open Circuit

OTI . . . . . .Oxygen Toxicity Index

OTU . . . . . .Oxygen Toxicity Unit (percent of the NOAA-introduced CNS Clock)

PBL . . . . . .Center for Project-Based Learning

PCB . . . . . .Printed Circuit Board

PHe . . . . . .PP of Helium

PN2 . . . . . .PP of Nitrogen

PO2 . . . . . .PP of oxygen

PP . . . . . .Partial Pressure

QFN . . . . . .Quad Flat No-Lead

RTC . . . . . Real-Time Clock

RTIC . . . . . Real-Time Interrupt-driven Concurrency

scuba . . . . . Self-contained underwater breathing apparatus

SPI . . . . . Serial Peripheral Interface

SWD . . . . . Serial Wire Debug

TBE . . . . . Time Between Executions

TBM . . . . . Time Between Measurements

TTS . . . . . Time to Surface

UART . . . . . Universal Asynchronous Receiver Transmitter

VPM . . . . . Variable Permeability Model

WOB . . . . . Work of Breathing

X2SON . . . . . Extra Small Outline No-Lead

Chapter 1

Introduction

Self-contained underwater breathing apparatus (scuba) diving is a recreational activity that involves breathing one or more gas mixes at hyperbaric (so greater than surface) pressure [1]. Dive computers in their most basic form consist of a depth sensor, a real-time clock, processing unit, and a display. Their primary function is to display dive time and depth, as well as to calculate and display decompression obligation and Oxygen Toxicity Unit (OTU)s during the dive [2]. Furthermore, most computers show statistics about recent dives or allow downloading the dive logs for further analysis and accurate logging when on the surface [3][4].

1.1. Motivation

Dive Computers commonly sample depth and update the corresponding display multiple times per second, while the fixed log rate (the stored logs for later retrieval) is usually user-defined between 1 and 30 seconds. Recomputation of decompression schedules cannot be as simply observed; but by observing the information on the respective displays change, it can be assumed that to happen on a scale of at least every few seconds.

As Prof Dr Simon Mitchell explains, hyperbaric medicine and especially decompression science is "bucket chemistry" [5], meaning that the extent to which accurate Oxygen (O_2) exposure and decompression tracking is useful to divers is at most on the order of multiple seconds. However, this does not mean that more frequent depth sampling and display is useless for all applications, most divers rely on their dive computers to refresh the depth display in increments of 10 cm with a sub-second delay.

However, there is no indication that knowledge about typical dive profiles influences these rates, i.e., there is no research in the public domain that investigates the feasibility of a variable refresh rate for the common components and algorithms of dive computers.

1.2. Objective

The primary objective of this thesis is to create a fully open-source dive computer hardware and software which can use both an exponential and linear-exponential algorithms, without incurring a significant performance overhead. Assembly should be possible without the use of pick-and-place machines, to allow other divers to extend the project and tailor it to their own needs.

1.2.1. Research Questions

1. Can variable-rate task scheduling algorithms be deployed without compromising accuracy?
2. Can sufficiently much energy and log space be saved by employing variable-rate task scheduling to offset the incurred overhead by the variable-rate algorithms?

1.2.2. Outline

The following chapter is an introduction to measurement-based embedded devices, energy-saving measures in battery-power devices, and basic hyperbaric medicine theory, to the extent that it is relevant to understand fundamental principles behind the diving algorithms used in this work. After this, related work to dive computer research and about saving energy in embedded devices, primarily by running at tasks variable rate, will be presented. Following this, the specific implementation of this project in hardware and software is described. Chapter 5 will collect the methods and results received using different software. Finally, the results will be critically discussed and compared with those in related work.

Chapter 2

Background

2.1. Embedded Systems

Definition 1 (Embedded System) *An embedded system (or embedded device) is an information processing or specialized computing system dedicated to a specific task embedded in a larger product, typically under constraints such as real-time behavior, limited resources, and low power consumption.*

Most embedded systems are built around a Microcontroller Unit (MCU), which exist in different package sizes, differ in clock frequency, memory, and flash size, and are produced by multiple manufacturers, the most common families include the STM32 and ESP32. Most MCUs include high speed and low speed (Real-Time Clock (RTC)) clocks, and to increase accuracy, the option of attaching an external crystal or oscillator for either is often provided.

While some embedded systems execute one or more tasks in response to external stimuli, others schedule tasks at regular or quasi-periodic intervals. Embedded systems also differ based on the larger system in which they are integrated: for example, airplanes and automobiles contain dozens or hundreds of individual devices that interoperate with one another, each controlling and accessing different peripherals, whereas small embedded devices often perform a single task and operate in isolation, controlling all connected peripherals.

2.1.1. Low Power Consumption in Battery-powered devices

Firstly, embedded devices are not comparable to general-purpose computer systems in that their hardware and software are fixed before production, increasing usage predictability and allowing for accurate simulations. Usually, only a single or small set

of fixed tasks have to be run and scheduled, typical specs for an MCU include RAM and flash on the order of dozens of kilobytes to a few megabytes, a CPU clock frequency (usually fixed but software-configurable) on the order of single-digit to hundreds of megahertz, and only the minimum required peripherals and components. Special compute power like floating point units and crypto hardware acceleration may be included in some units, while others do not incorporate them.

Power consumption is often a limiting factor for embedded systems, especially when running off a battery, which should provide sufficient runtime for the device's application.

From a hardware standpoint, component selection can significantly impact battery life, and choosing high-quality low-loss components designed for low voltage and current is usually desirable.

The software also influences battery life, mainly through the efficiency of the algorithms implemented, the definition and usage of provided of sleep modes and interrupt/wakeup handling, and the scheduling overhead of tasks. The interrupt controller is a central part of each MCU, and the number of available interrupts differs between packages and units. Interrupts can be internal to the processing unit or exposed via General Purpose Input Output (GPIO), available for external hardware to trigger them.

Low-power Modes

ST Microelectronics provides multiple low-power modes in their MCUs, which differ in their power consumption and wake-up behavior.

In sleep, low-power run and sleep and STOP modes, SRAM and peripherals (excluding clocks in STOP modes) remain initialized, while these are lost in the Standby and Shutdown modes, the main power regulator is turned off in all STOP1, STOP2, Standby and Shutdown, and the modes have different available wakeup sources and wakeup timing. When not running any tasks for some amount of time, lower power consumption and higher wakeup latency can be achieved by choosing a deeper sleep mode, depending on the project requirements.

2.1.2. Variable-rate scheduling

As noted above, some embedded systems only react to external events and interrupts, while others need to schedule tasks periodically. To reduce power consumption when considering a system with periodic tasks, it has been shown that it may be beneficial to schedule tasks at varying rates rather than relying on fixed-rate scheduling [6] [7] [8]. The variable rate may be computed using information about the system, such as battery charge, recharge, or external conditions, such as sensor readings.

The objective for the variable-rate scheduling of tasks is to reduce the CPU active time while adhering to the given requirements; the exact requirements in this project include the following, which have been derived from the experience gathered with multiple dive computers.

- The decompression calculation Time to Surface (TTS) must be accurate to at least 60 s compared to a high-rate reference implementation, as this number is representative for the decompression schedule and is usually shown in a resolution of 1 min.
- Oxygen Toxicity calculation must be accurate to ±1 OTUs, which accumulate at a rate of no more than $1.93\,\mathrm{min}^{-1}$, which equates to a maximum accepted delay of 31 s at high PP of oxygen (PO2). This is assuming the recommended PO2 limit of 162 kPa [9][10].
- The resulting dive log must be accurate to less than 1 m in depth at all times and 0.1 m on in the mean over all samples compared to the true simulated depth.
- The depth display with a resolution of 10 cm should have a sub-second perceived latency.

While there is no publicly available data on recalculation rates for decompression obligation and oxygen toxicity, they are assumed to be on the order of seconds. Thus, the fixed-rate scheduling baseline in this project is one calculation in 5 s for decompression and profile logging and 10 s for oxygen toxicity, which closely matches typical settings and observed behavior in dive computers. For the oxygen toxicity, it is intentionally conservative to approximate frequent recalculation.

To compare the baseline to the variable-rate algorithms, the cycle count will provide an accurate measurement, and this is converted to energy spent on these tasks, under a fixed clock frequency.

2.2. Diving and Humans in Hyperbaric Environments

2.2.1. Inert gas loading and Decompression

Due to the increase in pressure of 1 bar for every 10 meters of seawater, the total pressure of the inhaled gas also increases with depth. The gas mixtures used in scuba diving consist of oxygen, nitrogen, optionally helium, and traces of other gasses, which are present only in negligible quantities. Oxygen is the only of these gasses actively used in chemical reactions in the body; the other gasses are referred to as inert. When breathed at a Partial Pressure (PP) exceeding current saturation, body tissues begin to saturate ("on-gas") with that inert gas until they reach equilibrium with inhaled PP and desaturate ("off-gas") when the inhaled PP is less than the tissue's pressure of that gas. This is

not an instant process; it takes some time for a tissue to saturate and desaturate to the inhaled PP, which is dependent on the specific body tissue.

While working in mines and on bridges starting in the 1840s, workers started reporting symptoms, some becoming paralyzed, that were later classified as Decompression Sickness (DCS), and air embolisms were determined to be the cause. Later, symptoms were significantly reduced by gradually reducing the surrounding pressure over some time instead of immediately ascending to surface pressure [11]. During the second world war, the German, British, and American Armies simultaneously, but independently in secret, reproduced the observed symptoms using hyperbaric chambers and developed tables for saturation pressure and time and the corresponding required decompression time and pressure/depth schedule [12]. Whilst these tables are still taught by agencies and used by divers to perform safe dives, requiring only a timer and a depth gauge to track exposure time and maximum depth, dive computers have become mainstream diving equipment, allowing for multi-level profiles, by knowing not only the maximum depth and total dive time, but exact time spent at each depth, which is used to model tissue pressure (or bubble size in other algorithms, which are not discussed in this work), increasing allowed dive time within the No decompression limit (NDL), the maximum time that can be spent at hyperbaric pressure without incurring mandatory Decompression stops, dynamically based on the actual dive profile.

Tissues in the body on-gas and off-gas different inert gasses at different rates, tissue-based algorithms model multiple tissues with halftimes of 5 min up to multiple hours (e.g. 635 min or 720 min in Bühlmann ZHL-16A and -16C, respectively) to keep the modeled supersaturation within experimentally developed tissue-dependent limits, while bubble models aim to limit bubble growth [13]. These algorithms cannot prevent DCS from occurring, as exact physiological factors and processes are not fully determined nor can all required parameters be measured while diving (sleep, hydration, fitness, and other bodily factors play a role, which are not easily measurable), but adhering to the limits given by tables and computers, significantly reduces the incidence. Most decompression algorithms provide a maximum supersaturation values per tissue, which may be an absolute supersaturation or be depth-dependent (e.g., the Thalmann Algorithm's MPTT (Maximum Permissible Tissue Tension) tables [14][15]).

2.2.2. Bühlmann ZHL-16 Decompression Algorithms

The most common tissue-based decompression algorithms used for recreational and technical diving within the limits of most scuba diving agencies are the Bühlmann ZHL-16 family (B, C) models. The Bühlmann-16C algorithm is an exponential tissue-based algorithm that considers the so-called M-Value (the maximum allowed supersaturation) per tissue, and whenever this value is reached, a decompression stop is added, until the next stop depth can be reached without going beyond the M-Value.

The on- and offgassing is exponential as it is modeled using halftimes, so in a time $t_{1/2}$, a tissue equalizes as given in 2.1, where $p_t\text{i.g.}(x)$ is the partial pressure of the considered inert gas in the tissue t at time x and $p_I\text{i.g.}$ is the partial pressure of the considered inert gas in the inspired gas.

$$p_t\text{i.g.}(t_{1/2}) = p_t\text{i.g.}(t_0) + \frac{1}{2} \cdot (p_I\text{i.g.} - p_t\text{i.g.}(t_0)) \tag{2.1}$$

The halftimes of the ZHL-16 algorithm's tissues are a set of 16 (17 for ZHL-16C) times from 4 min (ZHL-16A) or 5 min (ZHL-16B/C) to 635 min [16].

The M-Value for an inert gas at an ambient pressure is calculated using the Bühlmann formulas and the a and b values [16] (pp. 129f), see 2.2, it is in a linear relation to the ambient pressure.

$$P_{I\text{tol}}\text{i.g.} = \frac{P_{\text{amb}}}{b} + a \tag{2.2}$$

This experimentally determined and then extrapolated M-Value line is often considered too aggressive, so two parameters, Gradient Factor Low (GF Low) and Gradient Factor High (GF High), influence the percentage of M-value the tissues are allowed to reach at the start of decompression (GF Low) and at the end of decompression, when ascending to the surface (GF High), between which a linear interpolation of these two points is followed for any stop depth d_i, where $i \in 1..\text{stop\_count}/3$ (2.3), as stops are built in 3 m increments. Common values range from 0.3 to 0.65 for GF Low and 0.75 to 0.99 for GF High.

$$gf_{d_i} = \text{lerp}(gf_{\text{low}}, gf_{\text{high}}, \frac{i}{\text{stop\_count}}) \tag{2.3}$$

The parameter Gradient Factor 99 (GF99) tracks the minimum distance to the M-Value over all tissues, it is controlled not to exceed the interpolated line in the Bühlmann Algorithm, as in the exponential implemented algorithm in this work.

2.2.3. Linear-Exponential Decompression Algorithm

Violations of Henry's Law

Usually, gas exchange between tissues is limited by Henry's Law, and follows an exponential relationship. In [17], the authors claim that due to physiological limitations related

to the fixed size of the diffusion region in which gas exchange takes place, there may be a limit to the maximum decompression that can occur at a specific depth that does not depend on the PP-gradient between neighboring tissues. Deeper diving is statistically related to a higher incidence of DCS [18], therefore, taking precautions such as limiting the Decompression stress by limiting the amount of molecules that pass through the lungs to be off-gassed, appears to be beneficial. This is the theoretical basis for transitioning to another type of function to model decompression in these cases, and only comes into effect on deep technical dives, where the physical exchange limit is smaller than the calculated diffusion allowed by the supersaturation gradient. When this point is reached, the modeled change in decompression achieved through an in- or decrease in pressure in the diffusion region only is exactly this change, yielding a linear relationship.

Iso-DCS risk algorithms

Bühlmann-16 and RGBM are deterministic [19], they consider the safety of a dive profile binary; safe or unsafe, although this is not a realistic model, as these dives still have some remaining risk of DCS based on the physiology of divers that differ between individuals or even the same individual on multiple days. These algorithms also fail to take into account the already accumulated and total decompression stress, which can increase the risk of DCS, and is often generated by deep dives requiring a long decompression. The US Navy and similar large organized entities often require a more predictable risk management strategy, and the DCS risk should not increase with the parameters of a specific dive. Iso-probabilistic DCS risk algorithms have been developed to model and execute dives with similar risk at varying depths and bottom times by integrating cumulative decompression stress into a continuous probability function [20].

Thalmann Algorithm

The Thalmann Algorithm VVAL 18 was developed by Capt. Edward D. Thalmann for the US Navy in the 1980s [15]. It uses the Haldanian exponential model for ongassing, while offgassing is only assumed to be exponential below a "crossover" supersaturation per tissue. Once the gradient is larger than this crossover, a linear function is used instead to model offgassing until the exponential has the same slope as the linear function, from which point, the exponential function is used to model the remaining decompression.

The Thalmann Algorithm has been statistically tested by the US Navy and yields iso-risk results for DCS. This risk (e.g., 3%, 4%) can be calibrated by changing the parameters of the algorithm [21].

2.2.4. Oxygen at hyperbaric pressure

Oxygen is not an inert gas and is thus not ongassed in body tissues. However, there are two risks when increasing and decreasing Fraction of Oxygen (in a gas mix) (FO2) of a gas: hypoxia and hyperoxia. The first term represents the lack of oxygen, which for humans occurs at a PO2 of around .10 to .16, and divers are recommended to breathe gas with a PO2 greater than .16 or .18 to avoid passing out. Oxygen also becomes toxic at high PO2 due to its chemical reactivity, and this effect has been experimentally shown to be more exposed when submerged in water than in dry hyperbaric chambers [12], which is why the upper PO2 limits for diving are 1.6 during decompression and 1.4 or less while working, while hyperbaric oxygen treatments according to the US Navy Tables in dry chambers often surpass a PO2 of 2.0.

The higher PO2 and the longer the exposures, the higher both risks of Central Nervous System (CNS) and pulmonary oxygen toxicity, and these conditions can cause serious short- and long-term medical complications, seizures, and death. Limits for multiple cases have been experimentally developed and published, among others, by the US Navy and NOAA, in tables and the so-called "CNS clock".

Keeping track of oxygen exposure is also a primary function of dive computers and will also be presented in this project. Although CNS clock limits are more restrictive than the way most technical divers manage risk and new recommendations have recently been published [10], these well-established methods have been implemented in most modern dive computers and will serve as a starting point in this work's implementation.

Dive computers do not only track single-dive exposures; divers are decompressing for hours after a dive is completed, and to complete repetitive dives (i.e. multiple dives within 48 hours) safely require the tracking of offgassing time even after the diver reached the surface. This is why dive computers are almost never completely shut down, as the real-time clock has to run to track inert gas offgassing and O_2 toxicity reduction at surface pressure over time.

O_2 toxicity calculation

NOAA limits, which combine both the CNS and the pulmonary limits, are set out by Table 15-3 published in the NOAA diving manual [22]. More recent statistical research taking into account a large number of dives by recreational Rebreather divers has proposed an extension of the limit for the exact PO2 of 1.3 bar from 180 min to 240 min at work and additional 240 min at rest, i.e. during decompression [10].

Furthermore, OTUs for pulmonary and CNS oxygen toxicity can be calculated separately using the Oxygen Toxicity Index (OTI) method 2.4 according to [9], using the suggested constants $c = 6.8$ with t in min for CNS oxygen toxicity and $c = 4.57$ with t in h for pulmonary oxygen toxicity. P_{O2} is given in absolute atmospheres.

> The pulmonary OTI should not exceed 250. The CNS OTI should not exceed 26,108 for a 1% risk. [9]

$$TI(t, P_{O2}, c) = t^2 * (P_{O2})^c \tag{2.4}$$

As long as none of the limits listed above is reached, a dive is considered safe under these theoretical models.

2.2.5. Further considerations

There are more considerations to planning dives and conditions that may occur particularly on deeper dives that should be taken into consideration by dive computers, although research on these is not yet conclusive. The following list is not complete, but is presented as an introduction to the many more topics that are researched in hyperbaric medicine.

- Isobaric Counterdiffusion (ICD) is a state in which a tissue is ongassing one gas while offgassing another, which is caused by inverted PP gradients between tissues and inhaled gas, between two inert gasses (i.e., Nitrogen and Helium). In most cases caused by gas switching on ascent, it may lead to temporary over- or undersaturation of tissues exceeding the M-Values, even though the decompression stops were followed according to the tissue loading at the start of the stop. [23]
- Gas density affects the ability of the lungs to recycle the gas and exhale CO2 [24] by increasing the so-called Work of Breathing (WOB). It is suggested to limit the gas density to $5.2\,\mathrm{g\,L^{-1}}$, with an absolute recommended maximum of $6.3\,\mathrm{g\,L^{-1}}$. Together with Inert Gas Narcosis [25], these are the primary reasons for using Helium-based gas mixtures for dives below 30 m.
- High Pressure Neurological Syndrome (HPNS) [26] is a condition triggered by the rapid compression phase in the beginning of a deep dive. Except for a slower descent or ascending when the condition arises, no prevention or cure has been found, except for a single trial dive with hydrogen [27].

Chapter 3

Related Work

3.1. Diving Algorithms and Dive Computer Hardware

There are not many publications related to diving outside hyperbaric medicine, which serve only as a limited reference for the implementation of decompression and oxygen toxicity calculations developed in this project, especially for variable-rate research, as it is relevant for dive computers. The algorithms have been researched and compared both from a medical and computational perspective, notably the Variable Permeability Model (VPM) algorithm being more computationally complex than Bühlmann's ZHL and similar algorithms [28].

Though diving has become a more mainstream hobby over the past 40 years, much research into wearable dive computer technology takes place in a commercial settings, which is not published, so the aim in this case is to provide an extendable open-source hardware and software platform.

From a medical perspective, Prof. Simon Mitchell confirmed that the recalculation rate of diving algorithms for scuba diving is on the order of seconds, not milliseconds or tenths of seconds, calling it "bucket chemistry" [5]. No specific research on this topic has been published, which is the primary motivation for this work.

3.2. Low-Power Algorithms and Variable-Rate Research

For low-power hardware and firmware design, multiple approaches have been evaluated on different platforms, and these influence the development of this project.

In [6], multiple approaches to Adaptive Sampling and Filtering in the context of IOT devices are discussed. While not an IOT device, but a wearable, the context of this work

resembles the Adaptive Filtering techniques insofar as a boolean decision (transmitting or processing data) is derived from the sampled data (as opposed to adaptive sampling, which would decide whether produce a new sample in the first place), but the algorithms are not filters in the signal processing sense. The behavior of the system reacts online to sensor samples.

In the following, multiple algorithms are discussed that have been proposed to calculate the dynamic sampling or filtering rate.

Energy-aware approaches take into account battery status and harvesting rate to achieve self-sustaining operation of IOT devices while keeping the sampling rate as high as possible [8], using an algorithm similar to Reno's Additive Increase, Multiplicative Decrease (AIMD). The sampling rate is inverse to the one used in this paper, reducing the rate additively and increasing it multiplicatively to avoid frequent sampling when the battery runs out. The proposed algorithm is lightweight, requiring 527 cycles per iteration, while achieving between 2.06× and 8.68× as many sensor samplings ("localizations") on average.

In [7], adaptive sampling based on changes in the data to sample has been researched, by deriving the sampling rate from the difference between consecutive samples. Introduces Time Between Measurements (TBM) (the name adapted in this work to Time Between Executions (TBE)), which is calculated using formula 3.1 as the basis, with E as the so-called exploratory value, the normal sampling rate, y_i and M_i sensor reading and measurement time at time i. This work produces independent TBM values, i.e., the sample rate does not depend on previous executions of the algorithm and can thus be calculated in a stateless fashion. They note that in some cases, the two current most recent samples do not provide sufficient and relevant data, but rather to use processing before calculating TBM. The technical results in this paper are not of interest in the context of this work as the resulting figures are not qualitatively comparable.

$$TBM = E - \left(S * \left| \frac{y_i - y_{i-1}}{M_i - M_{i-1}} \right| \right) \tag{3.1}$$

In the Adaptive Selection Algorithm for ISP Adaptive Filtering proposed in [29], a window of samples that is passed into the filter is filled until the next run. Absolute limits on the sampling window size and maximum duration are used to avoid missing many samples or only receiving stale data. Quantitative analysis yields a reduction of 6.1× to 567× in additions and multiplications, respectively, compared to other algorithms in test datasets, but these results are combined from the entire enhanced processing pipeline and not only from the ASA algorithm.

In this work, pressure sampling is performed in regular intervals and variable algorithms use the results from this sampling. Therefore, all the papers above provide limited

reference and comparison as to the extent to which the adaptive rate processes in this project can benefit the energy consumption.

Chapter 4

Implementation in Hardware, Firmware and Algorithms

4.1. System Requirements

The Dive Computer should be able to accurately measure ambient pressure, compute and display depth, decompression, and O_2 toxicity while diving and also on surface breaks, log events to flash, and allow for log download. The display must be large, bright and clear enough to be readable in the sun, as well as in low-light and low-visibility conditions. The device should be waterproof and resistant to absolute pressures greater than 4 bar to allow diving to 30 m, though this should be increased in upcoming prototypes to allow deeper diving.

The battery should last for at least two dives with on average 90 min per dive without charging, including the download of the dive log, and a sufficient surface interval to track the entire offgassing process.

To fulfill these requirements, a wireless approach has been chosen, induction is used for wireless battery charging, while Device Firmware Upgrade (DFU) and dive log download are performed by Bluetooth Low-Energy (BLE).

In order to save power, two distinct modes are implemented, dive and surface mode, where the first measures depth and tracks Decompression and O_2 toxicity on a sub-second scale, while the latter only measure ambient pressure every few seconds and enables the Bluetooth task for surface communication when necessary.

The screen has been determined to be at least 1.8″ large and full-color to be well-visible even in low-light and low-visibility conditions underwater, and to fit the low-

power requirements and since most of the display will be black during operation, an OLED screen is appropriate.

4.2. Hardware

4.2.1. Component Definition

The STM32L476 MCU is used as the computational center of this project, providing protocol support for Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI), Serial Wire Debug (SWD) and Universal Asynchronous Receiver Transmitter (UART). It has low power consumption, supports multiple sleep modes, and there are plenty software and help resources available on the internet.

The MS5849-30BA is used as a pressure sensor, it is rated to a depth of 290 m (30 bar) and provides a base accuracy of a few centimeters, provided optimal conditions. To reduce pressure drift, a supply voltage of 3.3 V has been chosen, although the sensor would be able to run with 1.8 V. The total error at extreme temperatures and pressures is, according to the datasheet, less than 0.04 bar, which is approximately 0.4 msw, though these extreme temperatures and pressures are not commonly reached. It includes an ADC, has direct SPI and I2C pinouts to make it easy to interface with, and development boards compatible with the Click boards™ standard are available and have been used for quick integration testing.

The STM32's LSI is not accurate enough for the timekeeping requirements of this project, therefore an external oscillator with a standard frequency of 32 768 Hz is used.

To verify power and active dive mode, Bluetooth active and Battery Charging, four LED have been added, a red LED to confirm power on 3.3 V, a blue LED that is turned from a GPIO pin on the MCU while in dive mode, an orange LED on the charging IC to show charging status, and a green LED on the $P0\_2$ output of the Bluetooth Transmitter IC, which turns on while transmitting.

For debugging purposes, 4 test-points for a UART connection to read live logs and a 6-pin pad-header for SWD are exposed, both including 1.8 V and GND header for reference.

To meet the power requirements, an SMD JST-PH Connector is provided on the PCB, which may be connected to a Li-ion battery with a nominal voltage of 3.7 V and a capacity of a recommended minimum of 2200 mAh. This also requires dedicated battery protection and charging Integrated Circuit (IC)s, which are chosen to provide sufficient protection to the board from external components (battery and charging coil). The effective battery output voltage is referred to as V_{BATT} and ranges from 3 V to 4.2 V. Most components require 1.8 V (MCU, flash) or 3.3 V (Pressure Sensor, LEDs, Bluetooth Module), which are generated using a fixed-voltage Dual Step-Down Converter.

A Newhaven Display 1.8″ full-color 160x128 OLED display (160128B) and a Microchip RN4871 BLE module for Bluetooth communication are connected to the MCU with SPI on separate buses. The display requires multiple reference voltages, defined in this project (within the constraints given by the datasheet) as $V_{DDIO} = 1.8\,\mathrm{V}$, $V_{DD} = 2.5\,\mathrm{V}$, and $V_{CC} = 17\,\mathrm{V}$. V_{DDIO} is already available on the PCB, V_{DD} and V_{CC} are generated using an LDO from 3.3 V and a step-up DC-DC PWM converter from V_{BATT}, respectively.

4.3. PCB Design

To reduce production cost, a layer count of 4 has been chosen for the PCB, including two ground plane layers to provide a stable reference to the signals on the remaining two layers. Instead of the typical layers 2 and 3, the ground planes are on layers 2 and 4 to reduce the impact of the inductive charging coil, which is placed adjacent to layer 4. All Power is routed using traces and filled zones on layers 1 and 3 of the PCB, the zones are used in dedicated places with higher expected current such as the battery connector, the DC-DC converters (which are laid out as suggested by the respective datasheets) and the power supply for components with higher instantaneous current draw such as the display V_{CC} and the pressure sensor.

The PCB has dimensions of $38\,\mathrm{mm} \times 48\,\mathrm{mm}$, the 64-pin MCU Low Profile Quad Flat Package (LQFP) is located in the center, the battery connector, charging, status ICs and converters are in the upper left quadrant, display Input-Output (I/O) and power converter in the lower left quadrant, while the right side holds, starting in the lower right corner, the Bluetooth module, flash IC, debug connector, debug LEDs, pressure sensor, and UART test pads.

All resistors and capacitors are of size 0402 (1005), LEDs 0603 (1608), two inductors 1210 (3225) and one inductor 1812 (4532), and most larger components are LQFP packages, with the following exceptions. There are three Ball Grid Array (BGA)-mount (battery status and power MOSFET for display and Bluetooth), two Quad Flat No-Lead (QFN) (pressure sensor and battery charger), one Dual Flat No-Lead (DFN) (dual step-down converter) and one Extra Small Outline No-Lead (X2SON) (I2C level shifter) components. Lastly, two Schottky diodes of Metric 0603 size are used. To connect external components, a 34-pin FFC connector for the display and a 3-pin JST PH connector for the battery are present on the board.

A figure of the routing layers in the PCB design can be found in D.1 and D.2.

All components have been soldered by hand using pneumatic solder paste dispensers, placed with a pair of tweezers, and reflowed using a heating plate at 250 °C. This process is not difficult, except for the display connector and the non-LQFP ICs, which require more concentration and time.

4.4. Firmware

The Firmware has been written in the Rust programming language, using the no−std environment due to dynamically-allocated objects not being available without a heap and only uses the core language features to reduce overheads introduced by the standard library, and to reduce firmware size. Multiple experimental features such as const generics, const trait implementations, and more const features are used to achieve the highest performance predictability and avoid runtime overhead and the need to multiplexing between implementations.

In Rust, an important concept is the so-called Zero Cost Abstraction, originally a principle defined by Bjarne Stroustrup, creator of the C++ programming language. Features such as zero-size types (including custom abstractions, like the ones used for pressure units in the algorithm implementations), iterators, and more enable writing very readable code while incurring no performance loss, which is especially important in the embedded environment where every cycle counts in order to reduce energy requirements.

4.4.1. RTOS

Real-Time Interrupt-driven Concurrency (RTIC) has been chosen as a task and concurrency management framework. It is not a full RTOS, as it does not include a kernel or HAL, relying on external crates, described in the dedicated subsection 4.4.2.

RTIC uses procedural macros for modules and functions to configure the app and tasks.

- rtic :: app: The main module
- rtic :: init : fn run at startup to set up peripherals
- rtic :: idle : fn run when no other task is running
- rtic :: task: fn to be scheduled as tasks, configured by the parameters on the macro

Tasks are scheduled by calling my_task::spawn for each execution, where my_task is the name of the task function. There are five tasks in the implementation:

- init to initialize peripherals, and calling the two main tasks task_mode_tick and task_battery_update
- idle as an infinite loop waiting for interrupts or other running instructions, which is preempted by RTIC as soon as another task may be run
- task_mode_tick is the main loop that runs a procedure based on the current mode
- task_battery_update running at a fixed rate (30 s) to update the latest measurements with the current battery status

- task_bluetooth_mode enables the Bluetooth module and handles incoming data and commands, and stops running when a timeout occurs
- task_bluetooth_button (unused in the V1 implementation discussed in this work)

Accessing external resources in RTIC is possible via locals and globals, which are passed as context to a task. Locals can only be used by a single task, whereas globals can be used by any task (and represent shared memory). The shared objects are the struct is latest_measurements, this is mainly to be able to write and read the battery status from both the dedicated battery and the main task, as well as three peripheral access objects to allow reading and writing to the flash from multiple tasks.

The main task task_mode_tick multiplexes between two different modes (dive mode and surface), runs one tick of the respective mode, and changes the mode if required. It is an infinite loop inside an async function, which awaits the delay between two executions asynchronously, allowing RTIC to schedule another task.

In surface mode, the main task runs less often than in dive mode, and it is useful to utilize a low-power mode built into the STM32 MCUs. Since the chosen sleep time of 5 s is many magnitudes higher than wakeup time from any mode, STOP2 is chosen, the lowest sleep mode in which all peripherals (except clock configuration, which is quickly reinitialized upon wakeup) and RAM are saved.

After waking up from STOP2, registers, RAM and most peripherals keep their state, but it is required to reinitialize the clocks because STOP2 mode might reset the clock tree, PLL, SYSCLK and associated timers and delays.

4.4.2. HAL

The standard package to access abstracted peripherals on ARM Cortex-M MCUs is the cortex−m (and cortex−m−rt) crate, which provides wrappers around core peripherals, for example SysTick, registers like MSP and BSRR, and functionality around startup and interrupt handling, including the NVIC (Nested Vector Interrupt Controller).

The HAL used in this project, which expands the functionality of the general cortex−m, specific to the MCU family, are maintained and provided by the community project stm32−rs, specifically the HAL crate for the STM32L4xx MCUs is used.

These three packages are purely written in Rust and do not depend on an underlying C implementation. The build chain uses Rust's Cargo build system, which uses probe−rs for flashing, debugging, and log collecting.

Example drivers provided by manufacturers are often written in C, the drivers for all used ICs have been rewritten in Rust and are also included in the project source code. These include Display, BLE, pressure sensor, flash, and battery status ICs.

4.5. Diving Algorithms

The algorithms described in this section are implemented in Rust in a dedicated repository, available under the MIT License. The repository is provided only for research and educational purposes. No warranty or guaranty is provided regarding the correctness, reliability, or suitability of the implementation for diving or other safety-critical applications, and the author assumes no liability for any damages, injuries, or losses resulting from its use.

4.5.1. Exponential Decompression Algorithm

The first algorithm implemented is the Bühlmann ZHL-16C algorithm, including the ZHL-16C halftimes and using an M-Value calculated using the Bühlmann formulas and the a and b values (pp. 129f in [16]), as described in 2.2.2. It can be tuned by altering the parameters GF Low and GF High, which reduce the calculated M-Value.

4.5.2. Linear-Exponential Decompression Algorithm

The linear-exponential decompression algorithm in this project is based primarily on the factors mentioned above and uses the Thalmann Algorithm [15] [14], as a reference for the parameters (specifically, the XVAL_HE9_040_F32 parameter set).

4.5.3. O2 Tox calculation

OTUs are calculated using the OTI equations outlined in 2.2.4, in addition, both the original NOAA tables and the revised recommendations for CNS and pulmonary O2 Tox risk mitigation at a PO2 no higher than 1.3 bar published in [10] are implemented. The NOAA tables serve as a backup for higher PO2 (and lower, if the resulting limit is longer than the newly recommended working limit of 240 min.

4.6. Scheduling of Diving Algorithms

As the primary research topic in this thesis, multiple approaches have been used for variable- and fixed-rate task scheduling.

To be able to compare them easily, a Rust trait RateAlgorithm is implemented for all concrete algorithms, which may hold some state and has a single function next_iter, taking two parameters (information about previous runs or a previous instance and a current instance) and returning an output, or an error. The output in all current implementations is the TBE in milliseconds, i.e., an output of 5000 signifies that the task

in question should be run if the time since the last execution is greater than or equal to 5 s.

4.6.1. FixedRateAlgorithm

The most basic algorithm is the FixedRateAlgorithm, which always returns a fixed number TBE, i.e., the task should run at regular intervals.

4.6.2. AimdRateAlgorithm

All dynamic scheduling rate algorithms in this work are based on the AIMD principle, the basic AimdRateAlgorithm has a number of milliseconds for additive increase, a fraction (numerator, denominator) for multiplicative decrease, and a maximum TBE that should not be exceeded (Listing 4.1).

Listing 4.1: AIMD algorithm

```
fn next_iter(self.current: u32, change: bool) -> u32 {
    if change
        self.current = self.current * MD
    else
        self.current = self.current.saturating_add(AI)
    if self.current > self.max
        self.current = self.max;
    self.current
}
```

Furthermore, information on the current rate is stored in every instance, as this number can change at every execution of the rate algorithm, and it is an output of the previous iteration rather than an input to the algorithm. The only input to the algorithm is a bool that indicates if the AI or MD branch should be executed. This algorithm is not used directly for any scheduling decision, but the following algorithms are based on this one.

4.6.3. DiffAimdRateAlgorithm

The algorithm DiffAimdRateAlgorithm works on a fixed-size window of previous measurements, always taking into account the oldest and newest sample in the window to be able to compare larger-scale changes, as small deviations between two adjacent samples are expected, and the larger the time between the first and last measurement in the window, the more of the overreaching trend can be captured by this algorithm. This means that the window cannot be too large or too small, in this case a fixed window

size of 5 has been chosen, which means that every iteration of the algorithm compares values that have been recorded $5 \cdot 200\,\text{ms} = 1\,\text{s}$. The window size does not influence the working or energy efficiency/computation time of the algorithm, only the sensitivity to change in measurements. Sensitivity can be further optimized by changing the constants (e.g. r_{ref}), which are properties of every algorithm instance.

Let t_0 be the time of the first measurement and t_n the last measurement (in this example, $n = 5 - 1 = 4$) in the window, and v_0 and v_n the corresponding values (4.1). We first calculate the change rate per second r, then transform it into a number f between 0 and 1 using a fixed reference speed r_{ref} (4.2).

$$r = (v_n - v_0)/(t_n - t_0) \tag{4.1}$$

$$f = \min(r/r_{ref}, 1) \tag{4.2}$$

If the change f is significant (assume >= 1), a change is triggered to invoke the MD branch; otherwise AI is used until the maximum is reached, as described in 4.6.2.

The window is implemented as a circular buffer to save memory and prevent otherwise necessary memory shifting after each iteration.

4.6.4. DynamicAimdRateAlgorithm

This algorithm adds one more component compared to the previous implementation, which is an absolute threshold. If this threshold is exceeded for one measurement, a change in the AIMD algorithm is triggered, i.e., once this threshold is crossed for a small to medium period of time (depending on the MD factor), the algorithm will quickly converge to the minimum value while the threshold is exceeded and can only recover once the threshold is passed.

The idea behind this addition is that in "deep diving"[1] limits are reached much faster than during shallow dives, so especially during bottom phases of the dive, seeing the most recent data is critical for decision-making, as for every period on the bottom, the ascent may become longer by a multiple (e.g., around 100 m, a minute of bottom time incurs additional decompression of at least 5 min provided optimal decompression gases are enabled, or more

This does not mean that the minimum rate is used for the whole dive; as soon as the bottom phase is completed, (depending on the dive, this may be more than half or only a small fraction of the total dive time, the deepest dives to date require more than 15 h of decompression time for less than an hour of bottom time [27]), and when decompression

[1] This term is commonly used in different contexts, but there is no absolute definition; in this case, it will be assumed that "deep diving" refers to situations in which O2 Tox and NDL are easily reached and significant amounts of decompression need to be taken into account while planning and executing these dives.

stops shallower than this absolute threshold are reached, the rate algorithm will recover to a slower result again, as less changes can be expected in this phase of a normal dive.

This algorithm, using different constants for the minimum and maximum rate, change detection, and absolute thresholds, is used for O2 Tox and decompression algorithm rate calculation.

4.7. Data Logging to Flash

The primary goal of data logging here is to log enough information to accurately reconstruct all desired parameters based on some allowed error, while keeping energy consumption and space minimal. To save on energy consumption and space requirements, an adaptive logging rate has been implemented, compared to most existing dive computers, which log a data point at a fixed rate. Furthermore, to save space on the chip, not all data points contain all parameters, which we call the log parameter selection.

The selected flash chip has 4 MB storage space, of which 2 MB are reserved for firmware updates and static storage, so 2 MB storage remain to store dive logs. Assuming a log width of 16 B, 125000 records can fit in this space, assuming a fixed log rate of one log in 5 s, these are more than seven days of continuous logging. Since the computer will not log more than once per minute on the surface, this is a practical configuration. Using parameter selection, we halve the space required for some log entries, but we are not dependent on reducing the log size this way, as the flash chip has enough space for the constraints of this project.

A control data block for each dive is saved; it contains information that is unchanged during the dive. To reconstruct the logged dive data, there is one byte that contains metadata associated with each data point F.2, the other 7 or 15 bytes hold measured or calculated data F.3.

Chapter 5

Results

5.1. Hardware Verification and Testing

A fixed-voltage DC Power Supply that has been set to output 3.7 V is used to provide power to the PCB.

The 3.3 V rail has the correct voltage, as do the 2.5 V and 17 V rails for the display. The MCU shuts off the display rails to reduce idle power consumption when the display is off by opening the gate to the 3.3 V line which feeds the 2.5 V and 17 V converters. The 1.8 V rail is close to 2.5 V and the voltage is reduced under load to approximately 2.3 V.

A debug connection with the host computer via the SWD pads can be established and code can be executed on the MCU; the pressure sensor and the BLE module respond, although a Bluetooth connection with an external device could not be fully established. Data can be stored to the external flash IC and is retained after a reboot.

The display connector is flipped, and thus the display cannot be correctly connected without potentially breaking it.

5.1.1. Power Consumption

At 3.7 V, with all components in the lowest possible power state except for the Bluetooth module (which always runs, including the transmit status LED and cannot be shut off on the V1 PCB), current draw is 5 mA, so power is 18.5 mW. When the MCU is also active, compared to the STOP2 mode, the current difference is not accurately measurable; according to the datasheet it should be 1.6 mA at 16 MHz. This figure is used in all subsequent calculations in which the MCU energy is computed.

When the MCU is active and both status LEDs are continuously active, current draw rises to 12.5 mA. With only the dive mode status LED, current draw drops to close to 9 mA.

In surface mode, every tick is very brief and only uses 0.0185 W (approximated over a longer time). The battery lifetime of a 2200 mAh battery in surface mode is therefore equal to 18.33 d.

In dive mode with the display inactive, this shortens to 9.64 d (0.035 W). The display draws significantly more power, with it on, the power consumption increases to 0.51 W + 0.035 W = 0.545 W, the battery life with a full battery in dive mode is thus 14.93 h.

5.2. Experimental Setup

Although cycle counting and timing are possible on external hardware, the most accurate results can be obtained by simulating fixed dive profiles to compare the different algorithms. The software is programmed onto the MCU using the 6-pin JTAG connector *J1* via a STLink-V3SET.

The MCU runs at a fixed clock frequency of 16 MHz on the STM32L476, each main task execution scheduled at a rate of approximately 1/200 ms.

The software can be configured using the Rust Build System cargo's feature flags, which are used for conditional compilation. To simplify interaction with different setups and simulations, a Justfile is provided for the just command runner. The standard dive computer application can be flashed using just embed, to enable the display and bluetooth, run just features=display,bluetooth embed, while live simulation and result extraction can be obtained by running just embed−live−sim−log−DEPTH(−FEATURE), where DEPTH can be one of 20m, 50m and 90m and FEATURE is optional, may be exp or fixed−rate.

Data were extracted using the logs from the MCU, by accumulating the logs in a file. This output has shown to be particularly unreliable, but the log lines have been reduced and made as machine-readable as possible to increase reliability. The numbers do not match exactly (e.g., see decisions due count is smaller than algorithm / flash log execution) due to missing spaces, newlines, carriage returns and other escape sequences that are not properly parsed when the stream is written to the file on disk and result in corrupt data. However, after multiple iterations yielding results with only insignificant differences, the relations between the numbers of multiple executions are considered to be preserved.

The following figures and tables are generated from the `csv` files that are output from the `extract_bench_tables.py` script in the project repository. No preprocessing steps are performed except to clean up invalid characters in the exported log. The figure generation

scripts are available in the thesis repository and are also run automatically once a commit is pushed to GitHub via GitHub Actions, namely `generate_decision_summary.py`, `generate_timing_summary.py` and `generate_dive_profile_graphics.py`.

The fixed rate simulations serve as the baseline for comparison, while the exp ones enable the exponential and the recipes without FEATURE run the linear-exponential algorithm.

In the following section, three types of table are used to present the data; in all tables **N** denotes the number of samples or data points. The fixed-rate scheduling algorithm is referred to as Baseline configuration, while the Exp. and Lin.-Exp. results refer to variable-rate scheduling decisions under the respective decompression algorithm. The Baseline configuration always uses the linear-exponential algorithm.

- Algorithm Computation Time: These show a summary (average, median, minimum, maximum cycles and seconds) of all executions during the simulation of one dive profile (maximum depth 20 m, 50 m and 90 m) under a specific configuration (Baseline fixed-rate, variable-rate with exponential or variable-rate with linear-exponential algorithm).
- Rate Algorithm Computation Time: These show a summary of all rate algorithm executions related to the scheduling of a specific algorithm (fixed over all rows, named in the table caption) during the simulation of one dive profile under a specific configuration.
- Decision Summary: Detail decisions (due, not due, and avg/max skipped seconds) of the rate algorithm related to the scheduling of a specific algorithm during the simulation of one dive profile under a specific configuration.

5.3. O2 Tox Algorithm results

The implemented oxygen toxicity algorithm is only differential, runs on just one new measurement, and thus is comparatively cheap. The cycle count per execution of the fixed-rate algorithm can be extracted from Table 5.2, 235 ± 2, is only 10% smaller than the execution of the O2 Tox algorithm itself, which is 266 ± 2 as can be seen in Table (5.1). This difference will become larger if more samples are used per-iteration of the O2 Tox algorithm to reduce the skew that the linear interpolation may occasionally introduce between previous and current measurement.

Config.	Depth	N	Avg c.	Avg [µs]	Med. c.	Med. [µs]	Min c.	Min [µs]	Max c.	Max [µs]	Tot. c.	Tot. [ms]
Baseline	20m	8	265	16.6	265	16.6	265	16.6	265	16.6	2120	0.1
Baseline	50m	42	266	16.6	266	16.6	266	16.6	266	16.6	11172	0.7
Baseline	90m	87	266	16.6	266	16.6	266	16.6	266	16.6	23142	1.4
Exp.	20m	1	265	16.6	265	16.6	265	16.6	265	16.6	265	0.0
Exp.	50m	27	265	16.6	265	16.6	265	16.6	265	16.6	7155	0.4
Exp.	90m	26	266	16.6	266	16.6	266	16.6	266	16.6	6916	0.4
Lin.-Exp.	20m	1	267	16.7	267	16.7	267	16.7	267	16.7	267	0.0
Lin.-Exp.	50m	34	266	16.6	266	16.6	266	16.6	266	16.6	9044	0.6
Lin.-Exp.	90m	27	265	16.6	265	16.6	265	16.6	265	16.6	7155	0.4

Table 5.1.: O_2 toxicity computation time across profiles and configurations

The variable rate algorithms are on average 7.4× more expensive, 1858 ± 32 (cf. Table 5.2) than the fixed-rate algorithm and 6.98× more expensive than a O2 Tox calculation (cf. Table 5.1).

Config.	Depth	N	Avg c.	Avg [µs]	Min c.	Min [µs]	Max c.	Max [µs]	Tot. c.	Tot. [ms]
Baseline	20m	933	237	14.8	237	14.8	237	14.8	221121	13.8
Baseline	50m	3346	234	14.6	234	14.6	234	14.6	782964	48.9
Baseline	90m	6258	237	14.8	237	14.8	237	14.8	1483146	92.7
Exp.	20m	894	1829	114.3	1829	114.3	1829	114.3	1635126	102.2
Exp.	50m	3206	1877	117.3	1810	113.1	1954	122.1	6020581	376.3
Exp.	90m	6209	1854	115.9	1114	69.6	1946	121.6	11516165	719.8
Lin.-Exp.	20m	864	1816	113.5	1117	69.8	1818	113.6	1569350	98.1
Lin.-Exp.	50m	3418	1880	117.5	1125	70.3	1954	122.1	6426195	401.6
Lin.-Exp.	90m	5929	1852	115.8	115	7.2	1944	121.5	10984583	686.5

Table 5.2.: O_2 toxicity rate-algorithm execution time across profiles and configurations

Following Table 5.3, compared to the fixed rate baseline, the O2 Tox algorithm is executed 0.65× and 0.82× as often on the 50 m profile, 0.15× and 0.25× as often on the 90 m profile, depending on the specific decompression profile. For the shallow dive to 20 m, no execution is required following the variable rate algorithm.

Config.	Depth	Dec.	Due	Not due	Avg skip. [s]	Max skip. [s]
Baseline	20m	487	6	481	47.567	135.836
Baseline	50m	2099	42	2057	6.468	24.191
Baseline	90m	3955	84	3871	3.209	17.005
Exp.	20m	515	0	515	0.000	0.000
Exp.	50m	2205	28	2177	8.689	37.064
Exp.	90m	3872	20	3852	13.682	45.469
Lin.-Exp.	20m	511	0	511	0.000	0.000
Lin.-Exp.	50m	2295	19	2276	14.733	54.736
Lin.-Exp.	90m	3700	19	3681	13.959	57.967

Table 5.3.: O2Tox decision summary across profiles and configurations

5.4. Exponential & Linear-Exponential Decompression Algorithm results

The decompression stop profile calculation algorithms are significantly more expensive than the O2 Tox algorithms, especially on dives with increased decompression obligation.

The difference between the exponential and linear-exponential algorithms is small for shallow dives, while on deeper dives, the average and total cycle counts are up to 16% larger (Table 5.5) for the linear-exponential algorithm, which is caused, among others, by the tendency to create more (thus deeper) stops, and with more depth changes during decompression, more executions are required. Using the STM32L476 datasheet current draw, this difference is between 0.0007 J and 0.017 J per execution.

Config.	Depth	N	Avg c.	Avg [µs]	Min c.	Min [µs]	Max c.	Max [µs]	Tot. c.	Tot. [ms]
Baseline	20m	757	219	13.7	219	13.7	219	13.7	165783	10.4
Baseline	50m	2935	218	13.6	218	13.6	218	13.6	639830	40.0
Baseline	90m	5662	220	13.8	220	13.8	220	13.8	1245640	77.9
Exp.	20m	689	1811	113.2	1796	112.2	1936	121.0	1248088	78.0
Exp.	50m	2783	1878	117.4	1793	112.1	1936	121.0	5227824	326.7
Exp.	90m	4907	1838	114.9	1787	111.7	1930	120.6	9020538	563.8
Lin.-Exp.	20m	689	1801	112.6	1788	111.8	1928	120.5	1241483	77.6
Lin.-Exp.	50m	2999	1878	117.4	1795	112.2	1934	120.9	5634143	352.1
Lin.-Exp.	90m	4814	1833	114.6	1225	76.6	1925	120.3	8826205	551.6

Table 5.4.: Decompression rate-algorithm execution time across profiles and configurations

The decompression rate algorithm takes $1842 \pm^{34}_{32}$ cycles as can be seen in Table 5.4, while execution of the decompression schedule algorithm varies widely with the dive profile and current state, between $101'521$ and $886'860$ cycles, close to the lower range of the spectrum on no-deco dives, where no tissue reaches its M-Value, with average cycles between 338614 to 395242 cycles. The average execution overhead of an additional rate calculation is between $\frac{1837}{395'242} = 0.0046 = 0.5\%$ and $\frac{1810}{101'521} = 0.018 = 1.8\%$.

Config.	Depth	N	Avg c.	Avg [ms]	Med. c.	Med. [ms]	Min c.	Min [ms]	Max c.	Max [ms]	Tot. c.	Tot. [ms]
Baseline	20m	36	101811	6.36	101811	6.36	101811	6.36	101811	6.36	3665196	229.1
Baseline	50m	137	189820	11.86	136057	8.50	101069	6.32	347567	21.72	26005350	1625.3
Baseline	90m	249	371455	23.22	263000	16.44	101167	6.32	882588	55.16	92492527	5780.8
Exp.	20m	8	101457	6.34	101457	6.34	101457	6.34	101457	6.34	811656	50.7
Exp.	50m	88	177434	11.09	134590	8.41	101713	6.36	349706	21.86	15614217	975.9
Exp.	90m	92	343165	21.45	150278	9.39	101549	6.35	885697	55.36	31571200	1973.2
Lin.-Exp.	20m	9	101360	6.33	101360	6.33	101360	6.33	101360	6.33	912240	57.0
Lin.-Exp.	50m	76	177419	11.09	134329	8.40	101523	6.35	349066	21.82	13483846	842.7
Lin.-Exp.	90m	85	328362	20.52	256207	16.01	101521	6.35	885463	55.34	27910819	1744.4

Table 5.5.: Decompression schedule computation time across profiles and configurations

The energy required for execution of deco calculations in the longest profile is 0.085 449 6 J on average and in total 6.41 J for the variable-rate executions and 21.27 J for the execution at a fixed rate. The difference is 3.5× on the deepest profile, 14.86 J.

Config.	Depth	Dec.	Due	Not due	Avg skip. [s]	Max skip. [s]
Baseline	20m	480	22	458	12.540	45.900
Baseline	50m	1777	83	1694	3.182	13.950
Baseline	90m	3222	135	3087	1.934	7.146
Exp.	20m	473	4	469	40.257	57.273
Exp.	50m	1798	57	1741	4.577	16.649
Exp.	90m	3258	54	3204	4.987	30.009
Lin.-Exp.	20m	421	4	417	50.320	90.395
Lin.-Exp.	50m	1924	57	1867	4.760	17.171
Lin.-Exp.	90m	3224	53	3171	4.929	20.403

Table 5.6.: Decompression decision summary across profiles and configurations

Table 5.6 summarizes the decisions as to whether a recomputation of the decompression schedule is due or not based on the respective fixed or variable-rate algorithm. Analyzing it, compared to the baseline execution count and depending on the profile, the variable rate algorithm leads to a 1.17× to 6× decrease in the execution count, which translates in an almost linear relationship to the total cycle count spent on the computation of the decompression schedule, while only incurring a 0.5% to 1.8% overhead compared to the execution of a decompression schedule computation. Although this may slightly increase battery usage during the deep and descent phases of the dive due to running the additional variable-rate algorithm, the MCU will be inactive for more cycles during decompression. For the additional execution of the variable-rate algorithm, the overhead for the whole 90 m sums up to 1.74 J, thus saving compared to the execution at a fixed rate is $14.86\,\mathrm{J} - 1.74\,\mathrm{J} = 13.12\,\mathrm{J}$.

In the 50 m profile, which represents an average easy decompression dive, the benefit through the variable rate algorithm is smallest, as the Decompression time does not significantly exceed the time at depth, while in shallow and deep dives, the decrease in cycles is significant, as no deco needs to be computed, or the execution count during decompression is significantly reduced, respectively.

5.5. Flash Log Rate results

The primary reason for using a variable rate algorithm for flash logging in this work is not to save cycles and energy by logging less often but to save space on the flash chip.

The flash log rate algorithm (DiffAimdRateAlgorithm) takes about 7.1% less cycles than the DynamicDiffAimdRateAlgorithm used for Decompression and O2 Tox calculation. This additional execution time compared to logging in every iteration amounts to approximately 1/1850 of the main loop rate (on average about 108.1 µs) compared to the 200 ms rate), as can be derived from Table 5.7. This additional execution requires 0.000 397 44 J per execution and 0.297 J per dive, 0.0018% of the 2200 mA battery per dive.

Config.	Depth	N	Avg c.	Avg [µs]	Min c.	Min [µs]	Max c.	Max [µs]	Tot. c.	Tot. [ms]
Baseline	20m	632	209	13.1	209	13.1	209	13.1	132088	8.3
Baseline	50m	687	207	12.9	29	1.8	209	13.1	142503	8.9
Baseline	90m	891	210	13.1	210	13.1	210	13.1	187110	11.7
Exp.	20m	745	1731	108.2	1726	107.9	1856	116.0	1290200	80.6
Exp.	50m	692	1731	108.2	1725	107.8	1855	115.9	1198014	74.9
Exp.	90m	802	1731	108.2	1722	107.6	1852	115.8	1388920	86.8
Lin.-Exp.	20m	742	1728	108.0	1723	107.7	1855	115.9	1282547	80.2
Lin.-Exp.	50m	734	1734	108.4	1727	107.9	1857	116.1	1273067	79.6
Lin.-Exp.	90m	622	1727	107.9	1719	107.4	1850	115.6	1074446	67.2

Table 5.7.: Flash log rate-algorithm execution time across profiles and configurations

Figures E.4, E.5, E.6 shows that the variable rate algorithm reduces the flash log points per dive by a factor of approximately 3.6× to 4.1×.

Finally, compare the flash log with the actual sampled or simulated depth at the time; the difference is referred to as the error below. The maximum error can be studied in Table 5.8, it shows that the algorithms all limit the maximum error to a few centimeters, except for the fixed-rate algorithm, which includes individual errors up to a few meters, most of which already recover on the next log point (cf. Figure E.5, E.6 for a figure showing the difference over time).

The maximum error using the variable rate algorithms are smaller than 70 cm, but similar to the errors produced by the fixed-rate logging, these always recover on the next log point and only up to two such errors occur per simulated dive profile. The average error is 0 cm for both scheduled flash logs at fixed and variable-rates.

Config.	Depth	N	Min [m]	Max [m]	Avg [m]	Med. [m]
Baseline	20m	78	-0.009	0.000	-0.000	0.000
Baseline	50m	332	-0.015	3.067	0.025	0.000
Baseline	90m	599	-0.014	5.812	0.022	0.000
Exp.	20m	19	-0.000	0.000	-0.000	0.000
Exp.	50m	85	-0.358	0.001	-0.004	0.000
Exp.	90m	152	-0.700	0.003	-0.007	0.000
Lin.-Exp.	20m	20	-0.000	0.026	0.001	0.000
Lin.-Exp.	50m	90	-0.420	0.003	-0.005	0.000
Lin.-Exp.	90m	149	-0.671	0.015	-0.004	0.000

Table 5.8.: Flash log depth accuracy: signed error between logged and interpolated dive profile depth

Chapter 6

Discussion

6.1. Open-source Dive Computer

The resulting PCB is small enough to be worn on a diver's wrist and includes all necessary components to serve as a dive computer; it is of a similar size as common dive computers, such as the Garmin Descent X50i or the Shearwater Perdix and Petrel series. It is solderable by hand and only contains five components (two metric 0603 diodes and three BGA mount components) which are preferably mounted using a microscope for visual reference. It could be placed in a watertight enclosure with a suitable battery and does not need to be opened again, since all the charging and communication methods are wireless. Only one extrusion must be present in the enclosure to connect the pressure sensor to the water, which is ready to be sealed using a standard O-Ring (⌀1.8 mm, height 0.8 mm, such as the ones used on high pressure swivels).

Not all components worked in V1, in a new version, the errors and shortcomings should be addressed as explained below.

- The low voltage rail is pulled up by a missing resistor on an I2C level shifter, by adding this resistor, the voltage should drop from 2.3 V–2.5 V to 1.8 V.
- The display connector is inverted, pins 34-1 instead of 1-34.
- There is no way to remove power from the Bluetooth module if not in use, which wastes significant current in surface mode when Bluetooth is inactive. A similar Load Switch as already exists for the display should be added to cut power.
- It is impossible to activate the Bluetooth module after the computer has been running for some time. A button should be added to allow starting the Bluetooth task asynchronously.

- A smaller battery connector might be used and placed more appropriately at the PCB edge to save space.
- The display connector and pressure sensor are not placed ideally. The display connector should be rotated to fit in the middle of the long side to better match the final orientation of the display without complicating cable routing. The pressure sensor should be placed so that neither the battery nor the display collide with the vertical axis through it to simplify a connection to the outside pressure.

The software is modular; it is split into diving algorithms and embedded firmware that includes peripheral management. This provides divers with knowledge in software/embedded development the flexibility to use new developed algorithms on the tested platform, or to make small adjustments to the hardware without the need to rewrite the entire software.

The output of the decompression algorithms was compared with the output of the Subsurface software, which is a common dive planning application, and a Shearwater computer, and the exponential algorithm stop depth and length matches approximately (both Subsurface and Shearwater apply custom safety factors, as does this software, thus, a perfect match will never occur). The linear-exponential algorithm provides a stop schedule that tends to start deeper on the 90 m-dive, as expected, the implementation is similar to the relevant procedures from the Thalmann algorithm design document [15]. The performance difference between the two implemented decompression algorithms does not exceed 16%, thus the linear-exponential algorithm is still less computationally complex than bubble models (e.g. the VPM algorithm, [17][28]).

Recreational dive computers are commonly less feature-rich than full technical diving computers. The implementation in this work offers the most common features of technical diving computers, including the consideration of ICD and the gas density for predicted gas switching in decompression calculations, and even has the option to use the most recent recommendations to prevent oxygen toxicity presented in 2025 for OTU computations [10].

Although the battery life in the dive mode has been calculated to reach 15 h, which lasts a typical diver for at least 5 to 15 dives, the battery life in the surface mode is only up to 18 d, which does not completely meet the requirements set forth by the goals section. Since tissues are fully saturated to surface pressure after a surface interval of at most 72 h, this does not limit the usefulness of the dive computer, although it means that charging is required at least twice per month for continuous operation even when not diving.

6.2. Impact of variable rate algorithms for Decompression and O2 Tox scheduling

The implemented variable rate algorithms use the AIMD mechanism prominently used by TCP Reno [30], which has already been tested in this context in existing works, e.g. [8]. From the data collected, the following conclusions can be drawn.

- After reviewing the differences in actual decompression schedules, no significant differences, more than a few seconds, can be observed during up to 80 min-long decompression dives, compared to running the same decompression algorithm at a fixed rate, with a decrease of up to 6× in the execution count of the decompression algorithm. This decrease is comparable to the 2.06×-8.68× increase in localizations achieved by [8] and 6.1× decrease of additions in [7], although the underlying reasons are very different and no more similar research has been found. The energy decrease in the deepest simulated dive profile is 13.12 J, which equals a reduction to 1/3.1× taking into consideration the additional rate algorithm execution time. In the simulated deep dive, the decompression calculation only requires 0.25% of a 2200 mAh battery, instead of 0.73% when running the algorithm at a fixed rate.
- The O2 Tox computation is always within the given requirement. However, for less expensive algorithms such as the O2 Tox algorithm, the cost of running a more complex scheduling algorithm outweighs the fewer cycles spent running the algorithm less often, thus simple decision algorithms should be chosen for these.

6.3. Impact of variable rate algorithm for flash logging

The amount of flash logs was reduced by 3.6× to 4.1× when using a variable rate algorithm, while the error remains consistently smaller than 1 cm. Only up to two log points deviate more from the actual profile when using the variable rate algorithms, while the fixed rate algorithm flash log creates more such errors.

Therefore, flash space required for profile logging can be significantly reduced by employing variable-rate algorithms, without compromising the accuracy of the logged profile.

This is true for these ideal dive profiles, which do not include more quick depth changes and small variations, as real dive profiles and pressure measurements might produce. Such changes of a few decimeters usually do not occur at a rate greater than the fixed-rate log frequency of 5 s selected for this work, but using a variable-rate algorithm may allow to either capture or ignore exactly these small quick variations, depending on the settings. However, the algorithm has not been verified to produce equally good results under these conditions.

6.4. Linear-exponential algorithm

The difference in cycle count between the implementations of the linear and linear-exponential algorithm does not exceed 16% during the simulated dive profiles, so the user can choose the algorithm that best suits their experience and planned dive without a major impact on the dive computer's performance.

6.5. Limitations

6.5.1. Data Collection

The data collected come only from simulated dives, using a fixed clock frequency and power supply, so it may not match perfectly the conditions of a real dive. Changing the clock frequency should not affect the cycle count, which was the metric used in the previous comparisons.

The difference between the MCU running and idle/sleeping is dominated by the display, and even if it is off, the status LEDs and BLE module energy usage, which are magnitudes higher. Since the display has such high energy usage and is required to be on in dive mode, the energy saving through running the decompression algorithm less often is marginal.

The variable-rate algorithm has also been used for flash logging, as the dive can still be reconstructed using non-uniformly samples, and this reduces the storage required. More research is required to evaluate this under real diving conditions.

6.5.2. Software and Algorithms

The current implementation of the algorithms is not yet perfectly optimized for performance, so there may be opportunities to decrease the time/cycles required per iteration of the algorithms. To invalidate the findings, substantial changes in the basic functionality of the algorithms would be required, though, not only the remaining optimizations.

Furthermore, most implemented algorithms differ significantly from standard or commonly used ones, being implemented from scratch, in a more extendable and reusable way, and the exact rate algorithms have not been researched before. To fully understand and extend the software, one must not only be comfortable with Rust and RTIC, but also read this write-up or the code. This is acceptable to the author as making substantial changes to potential life-support equipment, a deep understanding of the technology, and a full understanding of the given implementation are prerequisites.

The bluetooth module is not yet completely functional thus the JTAG pads are used for DFU in this V1 prototype.

Although some divers started to experiment with hydrogen for dives below 200 m, as is done in saturation diving, to further reduce the gas density and prevent HPNS [27], the number of scuba divers who deploy hydrogen-based mixtures and research documents on this topic pertaining to scuba diving remains negligible, and the currently available scientific literature is too limited to justify including hydrogen in the scope of this work.

Chapter 7

Conclusion and Future Work

This work has presented a functional open-source hardware and software implementation of a personal dive computer for scuba diving. Two distinct tissue-loading-based decompression algorithms and oxygen toxicity calculation methods can be chosen, including a linear-exponential decompression algorithm. The hardware contains all required parts and costs no more than $60 per fully-soldered PCB, excluding a suitable Li-ion 3.7 V battery (e.g., 2200 mAh, 3-pin JST-PH Connector with thermistor connected on the middle pin). Such a battery is estimated to support up to 18 d of surface mode operation or 14.9 h of diving. The diving algorithms are implemented as a Rust library crate to allow users to use these algorithm implementations in custom projects. The embedded software is split into multiple modules to facilitate updating a component while keeping the remaining part running as given.

This thesis is primarily concerned with the energy-saving aspects of dive computers, not hyperbaric physiology. The implemented diving algorithms should now be evaluated in diving for medical and physiological correctness, but this would require a more extensive investigation in a larger multidisciplinary team to verify and gather the algorithms and results.

The main task is scheduled at a fixed sub-second rate to ensure that the most recent depth measurement is up to date. Decompression, oxygen toxicity algorithms and flash logging are performed at variable rates to increase resolution in times where sufficient information gain leads to significant changes in the expected output, and to decrease battery usage when no significant changes are expected. This reduces the required average cycle count per task execution without compromising the useful accuracy of the resulting information, and the incurred overhead is a magnitude smaller than the cycle skipped executions.

Provided the user designs a waterproof enclosure with only one extrusion to allow the pressure sensor contact with the water, the computer can be used for at least 6 h to 20 h, depending primarily on the current draw of the display, in the dive mode and for more

than a month in surface mode, which is sufficient to complete a deep technical dive and complete the following offgassing tracking in surface mode.

Future work could improve the runtime of current algorithms by optimizing code paths or running these decision algorithms less often. Another direction would involve taking into consideration more than just one older sample that the most recent one is compared to to be able to build a virtual trend line instead of relying on the accuracy of single measurements. The impact of the variable flash rate algorithm on reconstruction of real-world dives should be investigated; this may allow to save flash space while capturing more detailed information about the actual dive profile.

Although not all components have worked together in this prototype, valuable data have been extracted and conclusions could be drawn that can impact dive computer development in the future, including but not limited to the STDC project.

Appendix A

Task Description

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CENTER FOR PROJECT-BASED LEARNING

Task Description for a Bachelor Thesis on

Open Source Linear-Exponential-Algorithm Based on a Custom Dive Computer

at the Department of Information Technology and
Electrical Engineering

for

Sebastian Thomas
tsebastian@student.ethz.ch

Advisors:	Tommaso Polonelli, tommaso.polonelli@pbl.ee.ethz.ch
Professor:	Prof. Dr. Luca Benini, lbenini@iis.ee.ethz.ch
Handout Date:	23.02.2026
Due Date:	05.06.2026

1 Project Goals

Dive Computers are used in SCUBA diving primarily to measure depth and time of the current immersion and compute tissue loading or bubble growth to allow divers to safely execute multi-level dives beyond no-decompression limits (NDL) by reducing the risk of decompression sickness (DCS) as well as pulmonary and CNS oxygen toxicity. Most modern Dive Computers are based on the tissue models (Bühlmann ZHL-16 (B/C)) or Bubble Models (e.g. the Reduced Gradient Bubble Model, RGBM), while use of the latter is less frequent. They primarily consist of one or more depth sensors, accurate real-time clocks, a computing unit and a display.

The Thalmann decompression algorithm developed by decompression scientists in the US Navy has been designed for deep diving beyond 100m based on an iso-risk model but is not available in any commercial dive computers. Next to the specific tables of parameters published, the main idea of the algorithm is to limit the modeled amount of decompression in all tissues while supersaturation is present in the tissues by employing a linear function with high and an exponential for lower supersaturation, while the Bühlmann algorithm only uses an exponential function even for decompressing in high supersaturation. The iso-risk Thalmann algorithm makes deep diving DCS incidence more predictable, keeping the risk constant independent of maximum dive depth and thus maximum supersaturation.

Virtually all commercial dive computers for SCUBA divers are battery-powered embedded systems, which makes energy management crucial to allow divers to make multiple and longer dives on a single charge, while keeping the form factor practicable. In broader embedded systems research, adaptive sample rates for components and calculation rates for algorithms have been researched and shown to significantly reduce energy consumption by increasing idle phases. In SCUBA diving, most dives consist of ascent- and descent stages as well as constant-depth stages (only small variations in depth). These stages have different demands on the sample as well as refresh rates of components to accurately report depth and recalculate decompression obligation and oxygen toxicity limits, and no research has been published as to how this may be exploited in dive computers.

Most commercially available dive computers are developed closed source and only provide information on dive computer functions and limitations. This project attempts to design and prototype a dive computer which can be adapted to specific divers need either by configuring available software or redesigning parts of the system without having to build a complete end-to-end system of hardware and

software. By implementing multiple decompression algorithms, including the Thalmann algorithm, this prototype will be comparable to existing commercial models while allowing divers to choose to increase their safety for deeper dives. Lastly, it demonstrates adaptive sampling and refresh rates for sensors, computations and the user interface, optimizing energy efficiency without compromising safety and maintaining the required accuracy.

2 Tasks

The project will be split into four phases, as described below:

Phase 1 (Week 1-4)

1. Wearable sensor component definition. This step also include the definition of system requirements
2. Final schematic which is used as a base for firmware and algorithmic development and improvement.
3. Investigate previous works and systems which use adaptive sampling to improve energy efficiency of electrical designs.
4. Literature review and analysis of DIVE decompression algorithms.

Phase 2 (weeks 5-8)

1. Schematic and project review with the supervisor.
2. Implement firmware for the dive computer, including the selected decompression algorithm on an evaluation kit that mimic the final MCU.
3. PCB layout design
4. Implement variable sample and refresh rate algorithm, tuneable by parameters, and use its output to schedule tasks.

Phase 3 (week 9-11)

1. PCB assembly and electrical verification.
2. Test the firmware on the developed DIVE computer design.
3. Field verification of the hardware/firmware co-design in comparison with the intended project specifications.

4. Energy efficiency calculation for a set of representative dive profiles.

Phase 4 (week 12-14)

1. Result analysis.
2. Report and presentation.

Milestones

The following milestones need to be reached during the thesis:

- PCB Design and manufacturing
- Final report and presentation

3 Project Organization

During the thesis, students will gain experience in the independent solution of a technical-scientific problem by applying the acquired specialist and social skills. The grade is based on the following: student effort; thoroughness and learning curve; achieving qualitative and quantitative results with a scientific approach; supporting practical findings with theoretical background and literature investigations; final presentation and report; documentation and reproducibility. All theses include an oral presentation, a written report and are graded. The report and presentation need to have publication grade quality to achieve a good grade. Students are graded based on the official ITET grading form[1]. For students of IIS (Prof. Benini) a special grading scheme exists, please contact your supervisor for details there. Before starting, the project must be registered in myStudies and all required documents need to be handed in for archiving by PBL.

3.1 Laboratory Rules

The students agree to follow the lab rules set by PBL staff, for detail please contact us. The most important points are:

- All ETH safety regulations need to be followed[2], in addition to ones given by PBL staff

[1] https://ethz.ch/content/dam/ethz/special-interest/itet/department/Studies/Forms/ITET_Grading_Form_2025.xlsx

[2] https://ethz.ch/staffnet/en/service/safety-security-health-environment/sicherheit-in-laboren-und-werkstaetten/laborsicherheit.html

- No device in the lab is used without introduction by your supervisor or PBL staff
- No device leaves the lab without being officially borrowed, this is done by PBL staff and needs your Legi.
- Any damage to devices or tools needs to be reported immediately to PBL staff.
- The Lab-desk is clean and free for others after you finished your task, or when you take longer breaks. All tools are correctly sorted into their drawers/cupboards when you leave.

3.2 Weekly Report

There will be a weekly report/meeting held between the student and the assistants. The exact time and location of these meetings will be determined within the first week of the project in order to fit the students and the assistants schedule. These meetings will be used to evaluate the status and document the progress of the project (required to be done by the student). Beside these regular meetings, additional meetings can be organized to address urgent issues as well. The weekly report, along with all other relevant documents (source code, datasheets, papers, etc), should be uploaded to a clouding service, such as Polybox and shared with the assistants.

3.3 Project Plan

Within the first month of the project, you will be asked to prepare a project plan. This plan should identify the tasks to be performed during the project and sets deadlines for those tasks. The prepared plan will be a topic of discussion of the first week's meeting between you and your advisers. Note that the project plan should be updated constantly depending on the project's status.

3.4 Final Report and paper

PDF copies of the final report written in English are to be turned in. Basic references will be provided by the supervisors by mail and at the meetings during the whole project, but the students are expected to add a considerable amount of their own literature research to the project ("state of the art").

3.5 Final Presentation

There will be a presentation (15 min presentation and 5 min Q&A for BT/ST and 20 min presentation and 10 min Q&A for MT) at the end of this project in order to

present your results to a wider audience. The exact date will be determined towards the end of the work.

References:

Will be provided by the supervisors by mail and at the meetings during the whole project.

Place and Date Zürich, 17.02.2025 Signature Student 

Appendix B

Declaration of Originality

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. **In consultation with the supervisor**, one of the following two options must be selected:

☐ I hereby declare that I authored the work in question independently, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

☒ I hereby declare that I authored the work in question independently. In doing so I only used the authorised aids, which included suggestions from the supervisor regarding language and content and generative artificial intelligence technologies. The use of the latter and the respective source declarations proceeded in consultation with the supervisor.

Title of paper or thesis:

Linear-Exponential Algorithm-Based Open Source Dive Computer

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Thomas	Sebastian

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guidelines.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zürich, 03.06.2026	S. Thomas

If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] For further information please consult the ETH Zurich websites, e.g. https://ethz.ch/en/the-eth-zurich/education/ai-in-education.html and https://library.ethz.ch/en/researching-and-publishing/scientific-writing-at-eth-zurich.html (subject to change).

Appendix C

File Structure

```
/
├── README ..... README with general information about the project.
├── KiCad ..... KiCad Source Files for PCB Schematic and Layout
├── STDC_C ..... STM32CubeMX-configured project .ioc
├── STDC-Ordering ..... Exported ordering information for the PCB V1
├── STDC-Ordering.zip ... Archive: Exported ordering information for the PCB V1
├── STDC-STM32-rs ..... Embedded Firmware implementation submodule
│   ├── Cargo.toml ..... Cargo project file for firmware
│   └── DEBUGGING.md ..... Debugging information
├── STDC-thalmann ..... Diving Algorithm implementations submodule
│   └── Cargo.toml ..... Cargo project file for diving algorithms
├── 01_report ..... Source files of the report submodule
└── 02_presentation ..... Source files of the presentation.
```

To flash an assembled PCB, use the `STDC-STM32-rs` submodule. There is a `DEBUGGING.md` that explains how to flash and test the software. The command runner just is recommended to easily run the predefined tasks, e.g. just features=bluetooth,display embed, just embed—live—sim—log—DEPTH.

To view and update the diving algorithms, update the `STDC-thalmann` submodule and use the updated dependency version in the firmware crate's `Cargo.toml`.

Appendix D

Additional Figures: PCB Design

The following figures are screenshots from the KiCad software, which has been used for the PCB design. The first and third layer are displayed here, the second and fourth layers only provide a ground reference, which is cut where VIAs are used to travel other signals than ground between layers 1 and 3.

The MCU is the LQFP-component with 64 leads in the center of the PCB, in the top left corner, the battery connector and power management section can be seen. In the lower left hand corner is the display connector and required DC-DC converters, capacitors and MCU GPIO connections.

On the right side, from the top to the bottom, there are the UART pads, the pressure sensor, debug LEDs next to SWD pads, flash chip and the Bluetooth module.

D. Additional Figures: PCB Design

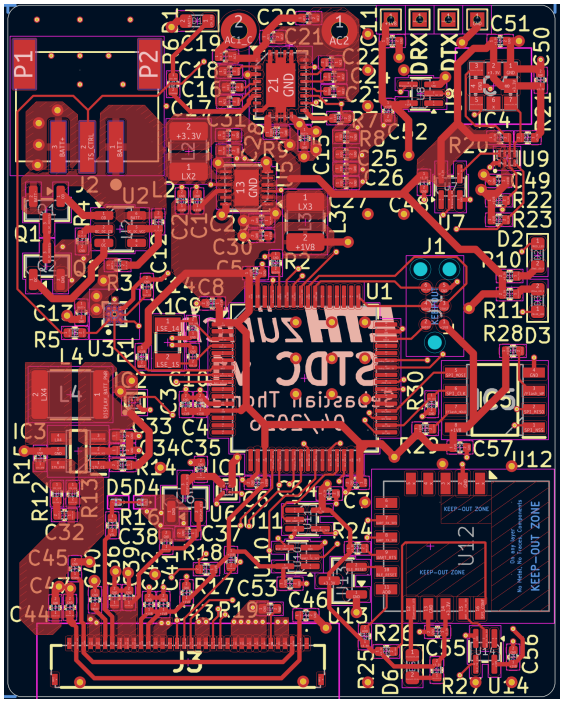

Figure D.1.: STDC PCB Layer 1 (Primary Routing & Component Placement Layer)

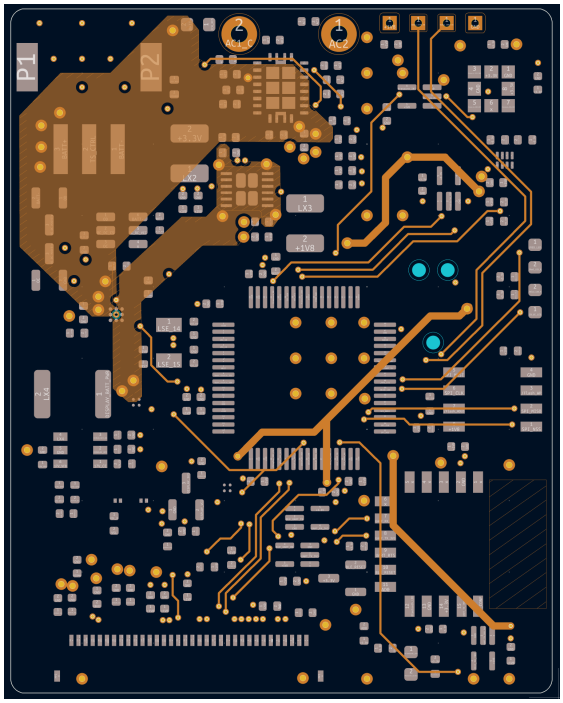

Figure D.2.: STDC PCB Layer 3 (Secondary Routing Layer)

Appendix E

Additional Results

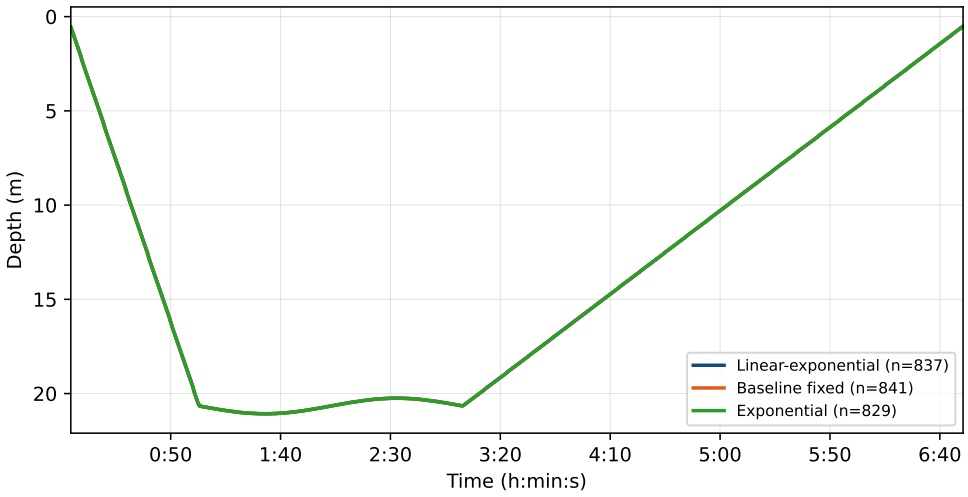

Figure E.1.: The dive profile of the simulated shallow dive with a maximum depth of 20 m. The descent and ascent rate are the only limiting factors.

The table Table E.1 is expected to show similar results, due to the inaccuracy of the occurrence of terminal escape sequences while parsing and writing to the file, the results only show a difference of 2.1× to 2.63×, which is still significant, but off the actual results derived from the flash log lines, which showed a reduction of 3.6× to 4.1×.

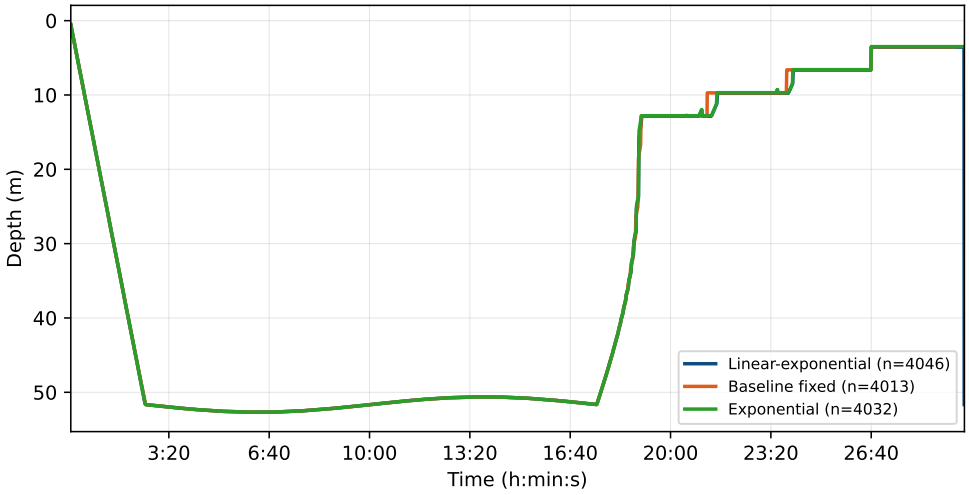

Figure E.2.: The dive profile of the simulated light decompression dive with a maximum depth of 50 m. The descent rate limits the descent, after 15 min bottom time, the linear-exponential or exponential decompression schedule is followed.

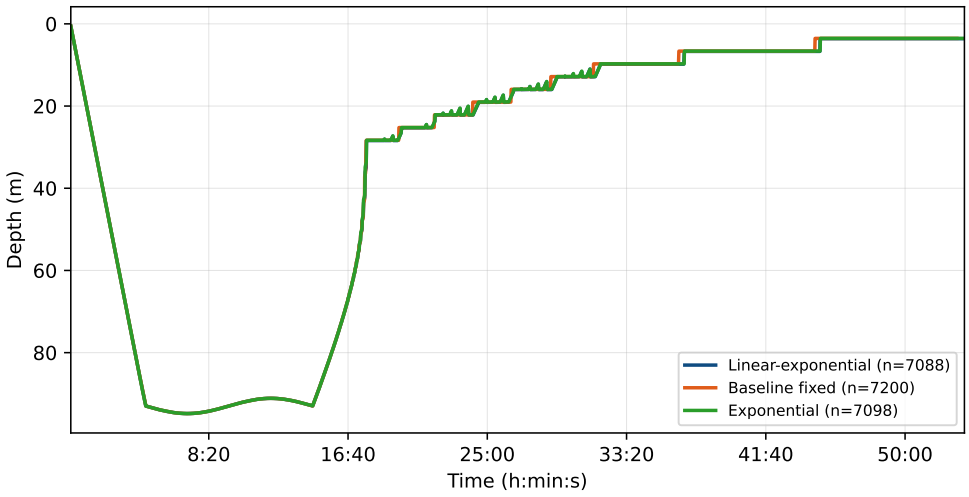

Figure E.3.: The dive profile of the simulated light decompression dive with a maximum depth of 90 m. The descent rate limits the descent, after 10 min bottom time, the linear-exponential or exponential decompression schedule is followed. Small spikes in depth using the fixed-rate algorithms occur because the simulation software changes depth once the previously approximated time ran out, even if the stop has not been cleared yet, after recalculation (which happens quickly after the ascent starting due to the variable rate algorithm's sensitivity to depth changes) the deeper stop is reached again.

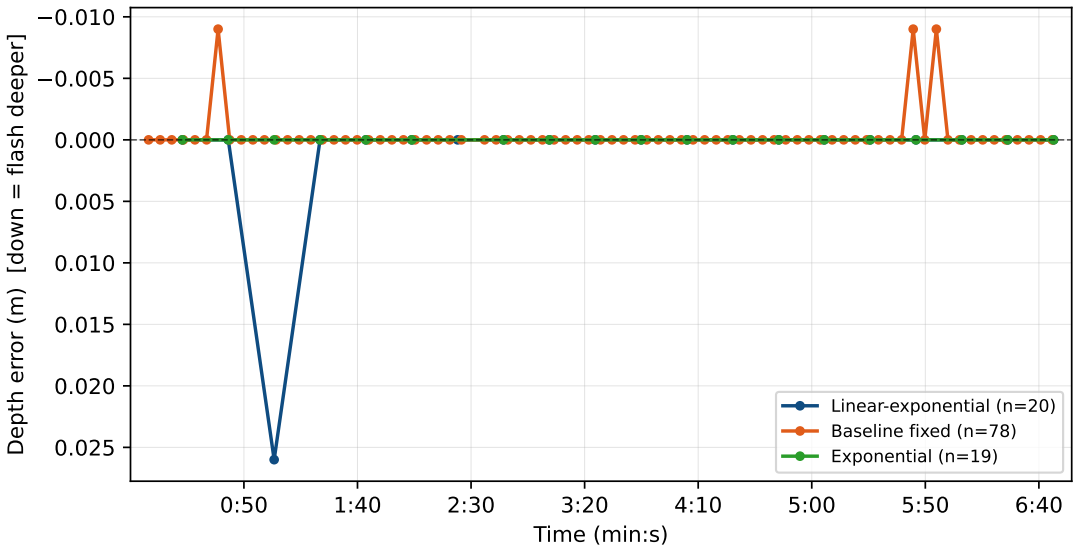

Figure E.4.: Simulated 20 m dive profile: Difference between Flash log and actual depth using the variable rate algorithms

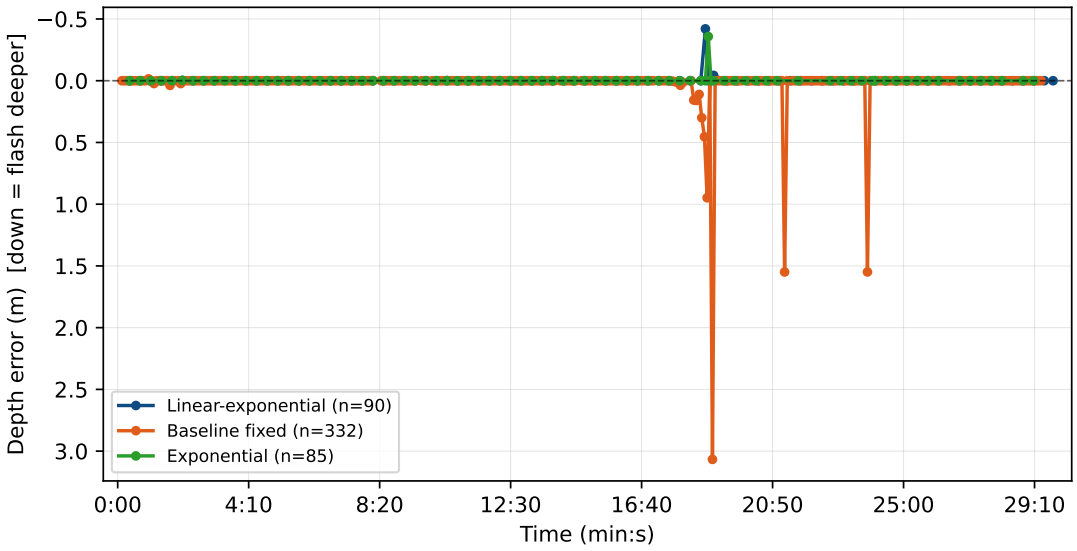

Figure E.5.: Simulated 50 m dive profile: Difference between Flash log and actual depth using the variable rate algorithms

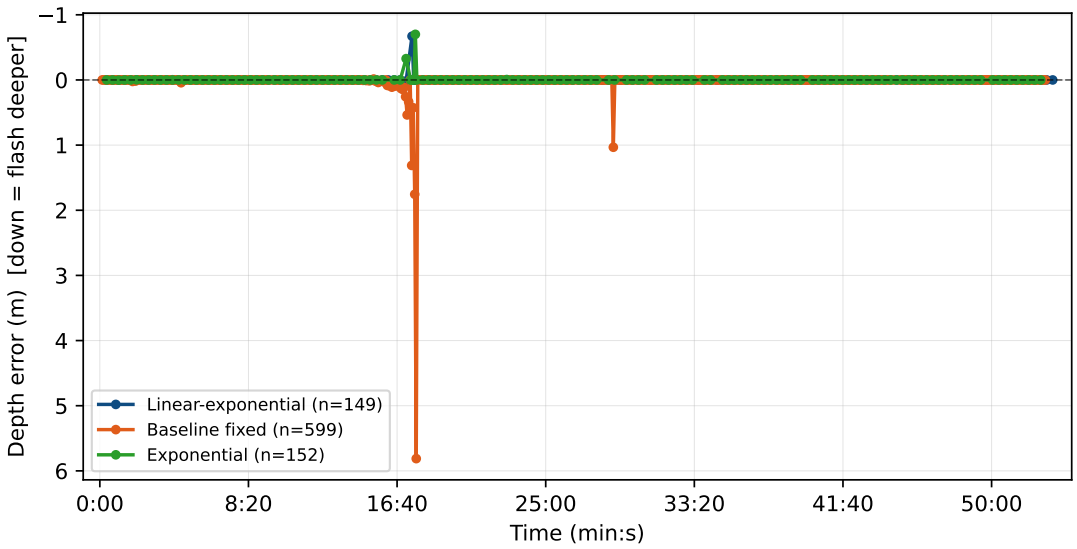

Figure E.6.: Simulated 90 m dive profile: Difference between Flash log and actual depth using the variable rate algorithms

Config.	Depth	Dec.	Due	Not due	Avg skip. [s]	Max skip. [s]
Baseline	20m	617	14	603	18.295	91.800
Baseline	50m	2424	73	2351	3.594	14.754
Baseline	90m	5048	108	4940	2.506	11.374
Exp.	20m	647	6	641	25.413	40.392
Exp.	50m	2555	29	2526	8.958	31.922
Exp.	90m	4821	55	4766	4.899	20.196
Lin.-Exp.	20m	644	6	638	41.606	60.588
Lin.-Exp.	50m	2954	38	2916	7.027	20.196
Lin.-Exp.	90m	4416	41	4375	6.504	20.196

Table E.1.: Flash log rate decision summary across profiles and configurations

Appendix F

Additional Tables: Flash Log structure

Name	Bytes	Description
firmware_version	2	identifier, from firmware, incrementing
computer_serial_nr	1	identifier, from firmware/hardware flashing
log_model_version	1	identifier, from firmware, incrementing
time_offset	4	s since 1970
surface_interval	4	in s
dive_number	2	Dive Number (identifier, strictly incrementing)
surface_pressure	2	in hPa
surface_temperature	1	in $2 \cdot °C$
ascent_rate_agg	1	Ascent Rate Aggregation in s, and 255 indicates firmware-defined non-constant aggregation
salinity	1	Salt (1035kg/m^2), fresh (1000kg/m^2), EN13319 (1020kg/m^2)
dive_mode	1	OC, CC
deco_algorithm	1	Table in firmware for mapping algorithm to identifier
gf_low	1	GF Low
gf_low	1	GF High
gas_nr	1	Number of distinct gases or gas cylinders
gas_content	$3 \cdot$ gas_nr	Number of distinct gases or gas cylinders

Table F.1.: Flash Log: Control Data Block

Name	Bits	Description
dive_mode_active	1	Indicator, if dive mode is active.
deco_obligation	1	Indicator, if there is a decompression obligation.
level_state	2	0 = level, 1 = ascending, 2 = descending, 3 = error in determining pressure or level
optional_param_pages	4	Indicator for optional 8-byte pages of log information, at the moment, only MSB used, as there is only 8 bytes of optional parameters

Table F.2.: Flash Log: Log Data Point Metadata

Name	Required	Bytes	Unit	Description
time_delta	Yes	2	s since t_0	real time of the measurement.
depth	Yes	2	m (= 0.1bar) (DM) hPa (SM)	10.6-fixed-point
metadata	Yes	1	n/a	as described in F.2
battery	Yes	1	%	Battery percent
temperature	Yes	1	$2 \cdot °C$	as signed 8-bit integer
ascent_rate	Yes	1	m/min	5.3-fixed-point, aggregation interval may vary with the firmware version
ndl	No	1	min	$[0, 99]$, 99 = overflow
ceiling	No	1	m	absolute value
gf99	No	1	n/a	current GF99
leading_tissue_id	No	1	n/a	if applicable, depending on algorithm, tissue index referenced by GF99
risk	No	1	%	if applicable, depending on algorithm
gas_mix_id	No	1	n/a	Index of the current Gas Mix, MSB 1 = CC, MSB 0 = OC
po2	No	1	10^3 Pa	Inspired PO2
cns	No	1	%	Percentage of the CNS clock, with 255 indicating invalid

Table F.3.: Flash Log: Log Data Point Data

Glossary

Decompression Decompression in diving nomenclature is the process of releasing bound gas from the tissues through breathing a gas that has a lower PP of that gas than currently present in some tissue. In a wider sense, it also includes the specific stop schedule that divers adhere to while ascending.

Technical Diving While the exact terminology differs between technical divers and agencies, most classify technical diving as every form of scuba diving that includes a virtual or physical ceiling, like decompression ceiling, cave and wreck diving. It also includes planning and executing a dive with multiple distinct gas mixes.

Bibliography

[1] C. Shilling, M. Faiman, C. Carlston, and R. Mathias, *Physics of Diving and Physical Effects on Divers. In: Shilling, C.W., Carlston, C.B., Mathias, R.A. (eds) The Physician's Guide to Diving Medicine.* Springer, Boston, MA, 1984.

[2] M. Kutter, "The history of the dive computer," *DIVE Magazine*, 2014. [Online]. Available: https://divemagazine.com/scuba-diving-equipment/history-of-the-dive-computer

[3] E. Azzopardi and M. D. J. Sayer, "A review of the technical specifications of 47 models of diving decompression computer," *Underwater Technology*, 2010.

[4] ——, "The silent witness: using dive computer records in diving fatality investigations," *Diving and Hyperbaric Medicine*, 2014. [Online]. Available: https://www.researchgate.net/publication/266948075_The_silent_witness_using_dive_computer_records_in_diving_fatality_investigations

[5] S. J. Mitchell, private communication, Dec. 2025, email communication about research-worthiness variable-rate measurement or recalculation in dive computers.

[6] D. Giouroukis, A. Dadiani, J. Traub, S. Zeuch, and V. Markl, "A survey of adaptive sampling and filtering algorithms for the internet of things," *DEBS '20: Proceedings of the 14th ACM International Conference on Distributed and Event-based Systems*, 2020.

[7] H. Algabroun, "Dynamic sampling rate algorithm (dsra) implemented in self-adaptive software architecture: a way to reduce the energy consumption of wireless sensors through event-based sampling," *Microsyst Technol*, 2020.

[8] M. Giordano, S. Cortesi, P.-V. Mekikis, M. Crabolu, G. Bellusci, and M. Magno, "Energy-aware adaptive sampling for self-sustainability in resource-constrained iot devices," *Proceedings of the 11th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems (ENSSys '23)*, 2023.

[9] R. Arieli, "Calculated risk of pulmonary and central nervous system oxygen toxicity: a toxicity index derived from the power equation," *Diving and Hyperbaric Medicine*, Sep. 2019. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC6881196/

[10] J. Hoyt, F. G. Murphy, N. W. Pollock, D. Kernagis, N. Bird, M. Menduno, J. Bright, and S. J. Mitchell. (2025, Sep.) Revised guideline for central nervous system oxygen toxicity exposure limits when using an inspired PO2 of 1.3 atmospheres. [Online; accessed 16-February-2026]. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC12500339/

[11] A. E. Boycott, G. C. C. Damant, and J. S. Haldane, "The prevention of compressed-air illness," *Journal of hygiene*, 1908. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC2167126/pdf/jhyg00378-0035.pdf

[12] R. Lance, *Chamber Divers: The Untold Story of the D-Day Scientists Who Changed Special Operations Forever*. Penguin Group, 2024.

[13] S. J. Mitchell, "Decompression illness: a comprehensive overview," *Diving and Hyperbaric Medicine*, 2024. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/38537300/

[14] D. J. Doolette, F. Murphy, and W. A. Gerth, "Thalmann algorithm parameter sets for support of constant 1.3 atm po2 he-o2 diving to 300 fsw," Navy Experimental Diving Unit, Panama City, FL, Tech. Rep., 2020. [Online]. Available: https://apps.dtic.mil/sti/html/trecms/AD1215316/index.html

[15] W. A. Gerth, "Thalmann algorithm decompression table generation software design document." Navy Experimental Diving Unit, Panama City, FL, Tech. Rep., 2010.

[16] A. A. Bühlmann, E. B. Völlm, and P. Nussberger, *Tauchmedizin (in German)*. Springer, 2002.

[17] R. S. Srinivasan, W. A. Gerth, and M. R. Powell, "Mathematical model of diffusion-limited evolution of multiple gas bubbles in tissue," *Annals of Biomedical Engineering*, Apr. 2003. [Online]. Available: https://doi.org/10.1114/1.1561288

[18] N. Bird, "Back to basics : Understanding decompression illness," *Divers Alert Network: South Africa Blog*, 2022. [Online]. Available: https://www.dansa.org/blog/2022/01/18/back-to-basics-understanding-decompression-illness

[19] R. D. V. Wayne A. Gerth, *Development of iso-DCS risk air and nitrox decompression tables using statistical bubble dynamics models*. U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, Office of Undersea Research, 1996. [Online]. Available: https://catalog.hathitrust.org/Record/008331480

[20] L. D. H. Paul K. Weathersby and E. T. Flynn, "On the likelihood of decompression sickness," *Journal of Applied Physiology*, 1984. [Online]. Available: https://doi.org/10.1152/jappl.1984.57.3.815open_in_new

[21] T. Blömeke, "Dial in your dcs risk with the thalmann algorithm," *InDepthMagazine*, 2024. [Online]. Available: https://indepthmag.com/thalmann-algorithm/

[22] *NOAA diving manual : diving for science and technology*, 1991. [Online]. Available: https://archive.org/details/noaadivingmanual00unit

[23] O. Hyldegaard, D. Kerem, and Y. Melamed, "Effect of isobaric breathing gas shifts from air to heliox mixtures on resolution of air bubbles in lipid and aqueous tissues of recompressed rats." *Eur J Appl Physiol*, Sep. 2011.

[24] *Respiratory physiology of rebreather diving. In Proceedings of the Rebreathers and Scientific Diving*, Feb. 2015.

[25] M. Rocco, P. Pelaia, P. Di Benedetto, G. Conte, L. Maggi, S. Fiorelli, M. Mercieri, C. Balestra, R. A. De Blasi, and R. P. Investigators, "Inert gas narcosis in scuba diving, different gases different reactions," *Eur J Appl Physiol.*, Jan. 2019.

[26] J. C. Rostain, C. Lemaire, M. C. Gardette-Chauffour, J. Doucet, J, and R. Naquet, "Estimation of human susceptibility to the high-pressure nervous syndrome," *J Appl Physiol Respir Environ Exerc Physiol*, Apr. 1983.

[27] R. J. Harris, C. J. Challen, and S. J. Mitchell, "The first deep rebreather dive using hydrogen: case report," *Diving and Hyperbaric Medicine*, 2024.

[28] B. Wienke and T. O'Leary, *Understanding Modern Dive Computers and Operation*. Springer, Cham, Sep. 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-94054-0_1

[29] S. M. Qaisar, "Efficient mobile systems based on adaptive rate signal processing," *Computers & Electrical Engineering*, vol. 79, p. 106462, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790618333433

[30] D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN, Vol. 17, No. 1*, pp. 1–14, Jun. 1989.

[31] STMicroelectronics, *STM32L476xx Ultra-low-power Arm® Cortex®-M4 32-bit MCU + FPU, 100DMIPS, up to 1MB flash, 128 KB SRAM, USB OTG FS, LCD, ext. SMPS Datasheet*, STMicroelectronics, July 2024, dS10198 Rev 11, 232 pages. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32l476rg.pdf