TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kilian Hubertus Ochs 184624IASM

SOLID MODELLING AND TESTABLE ELECTRONIC DESIGN OF A SMALL UNDERWATER ROBOT

Master's thesis

Supervisors: Roza Gkliva
MSc
Jaan Rebane
MSc

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kilian Hubertus Ochs 184624IASM

VÄIKESE ALLVEEROBOTI MAHTMODELLEERIMINE JA TESTITAV ELEKTROONILINE DISAIN

Magistritöö

Juhendajad:	Roza Gkliva
	MSc
	Jaan Rebane
	MSc

Tallinn 2020

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kilian Hubertus Ochs

25.05.2020

Abstract

A small-scale underwater robot developed by the Centre for Biorobotics, Tallinn, is the focus of this thesis work. This robot has primarily been developed for demonstration of bioinspired locomotion, but it has slowly transitioned into a research object and a teaching tool for university students in the field of robotics. A robot with such high manoeuvrability could very well be used in applied underwater missions for archaeological or maintenance purposes, if the robot's computational and sensing capabilities were improved.

This thesis work aims to raise the significance of this miniature robot by providing the necessary upgrades: a new motherboard which implements a more powerful processor, integrates a camera and an advanced communication and remote control system, along with an updated physical design – resulting in improved user interfacing, dependability, producibility and ease of assembly. Apart from that, a hierarchical structure for dividing tasks into high-level and low-level control is proposed and implemented in the example of a test suite which serves as a practical aid in debugging the circuit board components. As a result of the tasks carried out, the new prototype is not only more competitive compared with other robots in its class, but also enhances the study quality when being used as a teaching tool.

Establishing the requirements and then presenting specifications and methods to fulfil them constitutes the larger part of this thesis.

This thesis is written in English and is 99 pages long, including 7 chapters, 32 figures and 17 tables.

List of abbreviations and terms

ADC	Analogue-to-digital converter
ARROWS	Archaeological Robot systems for the World's Seas
ASCII	American standard code for information interchange
AUV	Autonomous underwater vehicle
CAD	Computer-aided design
CNC	Computer numerical control
CPG	Central pattern generator
DOF	Degrees of freedom (unit)
EEPROM	Electrically erasable programmable read-only memory
EMC	Electromagnetic compatibility
FDM	Fused deposition modelling
FIFO	First in, first out
FM	Frequency modulation
FPGA	Field-programmable gate array
FSK	Frequency-shift keying
FTDI	Future Technology Devices International
GCC	GNU compiler collection
GPIO	General-purpose input/output
GPS	Global positioning system
I^2C	Inter-integrated circuit
IC	Integrated circuit
ICSP	In-circuit serial programming
IDE	Integrated development environment
IMU	Inertial measurement unit
INS	Inertial navigation system
LDO	Low-dropout regulator
LED	Light-emitting diode
MEMS	Microelectromechanical system

MOSFET	Metal-oxide silicon field-effect transistor
MUX	Multiplexer
PC	Personal computer
PCB	Printed circuit board
PWM	Pulse width modulation
RF	Radio frequency
RGB	Red/green/blue
RISC	Reduced-instruction-set computing
ROS	Robot Operating System
SLA	Stereolithography
SMD	Surface mount device
SPI	Serial peripheral interface
SPP	Serial port profile
SSH	Secure shell
TalTech	Tallinn University of Technology
TCP	Transmission control protocol
U-CAT	Underwater Curious Archaeology Turtle
UART	Universal asynchronous receiver-transmitter
USB	Universal Serial Bus
UWSim	Underwater simulator
WLAN	Wireless local area network

Table of contents

List of figures

List of tables

1 Introduction

In this chapter, the reader will be introduced to the general background of the matter and study field with which this thesis is involved. Since this project is concerned with the improvement of an existing robot named µ-CAT (pronounced as “micro-cat”), the developers, the history and the main purpose of this robot are laid out first. After that, a discussion on comparable underwater robots will help clarify µ-CAT’s standing among them. This leads to a statement on the need to upgrade this underwater vehicle and finally to a personal note on the motivation for this thesis work.

The remainder of this thesis is structured as follows: Chapter 2 gives an overview on the scope of the work on which this thesis is based by describing the initial state of the robot and then detecting the required modifications and upgrades. Chapter 3 to Chapter 4.23.6 present the methodology for execution of the tasks in the fields of physical design, electronics and software development, and each of them contains a section on the achieved results in the end. The thesis closes with a cost estimation of the new robot and some summarizing statements on the accomplished work.

1.1 About µ-CAT

The Centre for Biorobotics is a science institute operating under TalTech (Tallinn Technical University) as a subdivision of the department of Computer Systems. It merges engineering and research activities of several fields, such as mechatronics, robotics and bioinspired sensing and locomotion. One of its main research areas has been the exploration and development of underwater robots which are inspired by biological lifeforms.

It has recently called out for students willing to improve one of their underwater robots, namely µ-CAT which is a smaller version of a similar robot named U-CAT (Underwater Curious Archaeology Turtle) [1]. While U-CAT was meant right from the beginning to serve in field projects, µ-CAT was initially developed for demonstration purposes and turned into a teaching tool and an object for in-lab tests. Compared to U-CAT, the production cost of µ-CAT has been kept at least one order of magnitude lower.

The underwater robots U-CAT and µ-CAT have been designed and built in 2013 as part of the European-Union-funded project ARROWS (Archaeological Robot systems for the World’s Seas) [1], [2]. µ-CAT, the scaled-down version of U-CAT, was then first shown at an exhibition called “Robot Safari”, held at the Science Museum London. For this show, four µ-CAT copies were produced within three months to replace the not yet functional U-CAT.

U-CAT itself was intended mainly as a helper for archaeological deep-sea explorations, since it can manoeuvre in very confined spaces due to its size and unconventional mode of locomotion which is inspired by a sea turtle [3]: Four independently driven flippers enable holonomic control on all six DOF (degrees of freedom). µ-CAT implements the same locomotion principle as U-CAT but can manoeuvre in even tighter confinements.

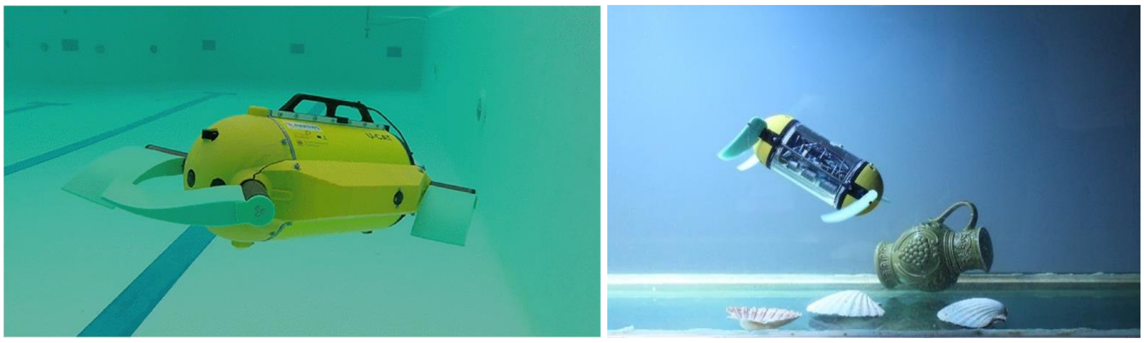


Figure 1. Underwater robots U-CAT (left) and µ-CAT (right).[1]

1.2 Literature review on small-scale underwater robots

For a robot to be able to do field work, it must sense its surroundings and move efficiently on the terrain or in the medium where it has been deployed. Some environments are hostile to human beings, and scientific and industrial sectors have been pushing forward the development of robots which can replace humans and perform dangerous tasks. One such example is the class of robots specialized on underwater missions. Traditionally, they have been large and not very agile, and they show diminished performance when

[1] Image used with permission from TalTech Centre for Biorobotics.

motion in confined spaces and stable station-keeping is required. To address these issues, the scientific community has turned to nature for inspiration, particularly learning from organisms which have been thriving in these environments [4]. In this regard, the sea is one of the most fascinating environments for robotic endeavours, offering a multitude of species to learn from. A large part of developed underwater robots does not accomplish missions, but consists of research and testing machines, required to study techniques of making underwater robotics more capable and efficient. Especially in the field of bioinspired fish locomotion, a lot of research has been done, and a variety of bio-inspired underwater robots have been developed in the recent years [5], [6], [7]. Underwater robots serve very different purposes. Some of them are utilized as study objects, mostly in order to analyse the modes of locomotion or sensing employed in biological lifeforms, others exist for practical use in field projects. In the following analysis, abilities of underwater robots which are comparable in size (largest dimension smaller than 100cm) and application to µ-CAT will be discussed and compared. A large part of the research is based on reviews by Raj and Thakur [6] and by Salazar et al. [7]. The aim of this literature review is to gain clarity on µ-CAT's role in the family of existing small-scale underwater robots and to understand its significance and potential.

As Wang et al. point out, [8, Ch. 7.1], AUVs for shallow waters are of rising interest to the robotic community. Their application scenarios are wide-ranging compared with the cost. µ-CAT falls into that category, and its usefulness depends on its level of autonomy. Even a robot with limited diving depth can perform complex missions – inspecting ship chests (Canterbury AUV; [8, Ch. 7]), serving as an aid in industrial processes or as a maintenance tool for nuclear storage ponds and water treatment facilities [9, Ch. 1.2], for pipeline-tracking in the oil and gas industry, for operations under-ice, gathering data to develop climate change models [10, Sec. I], to mention just a few examples. The applicability to many of these use cases depends not so much on diving depth, but rather relies on the small size and the high manoeuvrability of the vehicle. Considering µ-CAT's outer dimensions, the full six DOF including in particular its ability to turn on the spot about all axes, it performs outstandingly in this regard and is therefore a serious candidate for a variety of applications where other (even more costly) robots would most likely not perform comparably. In this regard, it is worth noticing that most robots presented in this analysis serve the main purpose of mimicking fish locomotion.

As Watson states [9, Ch. 2.6], there exist only few underwater vehicles with a small form factor and mass – the class into which µ-CAT belongs. The author emphasizes the special potential for such vehicles to be used in swarms, which presupposes some form of inter-communication between the robots. The fact that micro-AUVs are especially useful when operated in swarms may lead to the conclusion that the cost-factor of a single device becomes more important than in robots which are meant to perform solo missions. Apart from that, robots which are supposed to operate in confined spaces are most useful if they can be operated untethered (to avoid entanglement) [5, Ch. 1], which means that they require enough onboard computation power to carry out a mission without external control. Their ability to perform a mission unattended depends also on their sensing capabilities.

Robot control is usually divided into at least two levels: high-level (executive) and low-level (reactive or behavioural) control [11, Sec. I], [12]. High-level control considers task executions which usually require longer time spans – several seconds or more. This typically includes definition and scheduling of overall mission goals and long-term navigational planning. Low-level control includes processes that are tightly coupled to the physical world and therefore directly interface with employed hardware – sensors, actuators and communication systems – and they are typically time-critical and perform in the magnitude of milliseconds or less. These two distinct levels of control are often reflected in the hardware architecture of the robot, as Section 1.2.1 shows.

The following comparison is divided into two parts: hardware architectures and sensors. The first analyses the number and types of processors used, to understand if the architecture allows a division into high-level and low-level functions, as Wang et al. describe [8, Ch. 3]. The second lists commonly used sensors, to provide insight into the sensing, navigation and localisation capabilities of the robot, amongst others. For all analysed robots, their primary purpose(s) are also stated – see table column “Applications”. The given lists distinguish between the following purposes:

- Study of bioinspired locomotion and swimming patterns. The robot aims to replicate locomotion principles found in biological lifeforms, mainly fishes.
- Study of bioinspired sensing and navigation. The robot aims to replicate means of intercommunication and environmental sensing found in biological lifeforms, mainly fishes. Although all robots are equipped with sensors, only some of them

are directly comparable with those found in fishes, for example the presence of an array of depth sensors is comparable with the lateral line of a fish, or the implementation of electrocommunication is inspired by certain families of fish.

- Teaching tool. If a robot is developed mainly for students to learn about principles of robotics, the robot falls within this category.
- Field projects. If the robot is developed to perform certain missions useful to industry or science (excluding the study of robotics aspects), it falls within this category.

1.2.1 Processor architectures

Valavanis et al. have identified four different types of control architectures for unmanned underwater vehicles [13]. Three of them involve a layered structure – and the one that does not is basically not in use. The division of high-level and low-level tasks is most likely reflected in the hardware: Smaller, low-cost and energy-efficient processors such as ARM[1], AVR[2], PIC[3] handle low-level control. More powerful processors including FPGA (field-programmable gate array) and those designed for PCs (personal computers) are responsible for updating mission goals, maintain supervision or instruct lower-level processes to execute based on decisions made to resolve exceptional situations.

The following comparison between state-of-the-art underwater robots should help estimate the computational power of a robot and reveal the employed hard- and software architecture approach regarding the distribution into higher and lower-level tasks.

[1] Originally “Acorn RISC (Reduced-instruction-set computing) Machine”; computer architecture.

[2] Microcontroller architecture based on RISC.

[3] Family of microcontrollers made by Microchip Technology.

Table 1. Processors and computing architectures in underwater robots. Applications: T: teaching tool; S_L: study of bioinspired locomotion; S_S: study of bioinspired sensing; F: field projects; U: unknown. Processor utilizations: L: low-level; H: high-level; X: level unknown; ●: combined levels (single processor); *: refers to the robot in the state after the tasks of this thesis work are carried out. More than one entry in a cell indicates that two processors of the same type are used for different purposes. Processors with unknown architecture are listed as "Other".

	Applications	**PC**	**FPGA**	**AVR**	**ARM**	**PIC**	**Other**
µ-CAT*	T S_L F			L	H		
Canterbury AUV [8]	F	H		L			
Robotic fish [14]	S_L				●		
Essex G9 [15]	S_L				H	L	
Robot dolphin [16]	S_L				●		
Cownose ray [17]	S_L				●		
Fish robot [18]	S_L S_S				H L		
CPG boxfish [19]	S_L				●		
Boxybot [20]	S_L					H L	
New boxfish [21]	S_L S_S				H L		
KnifeBot [22]	S_L					H L	
Sepios [23]**,** [24]	T S_L		L		H		
Low-cost AUV [25]	F			H L			
Micro-AUV [26]	U			L	H L		
MONSUN [27]	F				H L		
MONSUN II [28]	F						●
Southampton [10]	F						●
Squidbot mini [29]	S_L			H	L		

As Table 1 reveals, ARM processors (such as the one used on the Raspberry Pi) are most commonly used, and in the majority of the cases, they are involved in high-level task planning. In this regard, µ-CAT's new design follows a commonly used hardware architecture. It is, however, surprising how few robots reportedly employ AVRs. These trivial components are either used but not reported or actually not used. If the latter is true, this might be related to the fact that ARM processors have taken over part of this market segment: They have become very available, cheap, and the community provides a variety of tools and support for efficient use of these very capable microcontrollers (including ROS; Robot Operating System). Also, ARM is employed on a lot of single-board computers, and these are convenient to use because they provide data storage and communication peripherals on the same PCB.

1.2.2 Sensors

The following analysis tries to identify the most commonly used sensors on existing small-scale underwater robots.

Localisation for submarines poses engineering challenges. Long-range wireless RF (radio frequency)-based signals or GPS (global positioning systems) are not applicable under water, and ultrasonic signals of hydrophones require external beacons to be placed which emit ping signals. Some robots employ a GPS module and surface from time to time to update their location accurately, and one example was found where the GPS tracker is released from the submarine in case of emergency and surfaces without the vehicle [25, Sec. IV]. If inertial measurements are used to estimate the position, the error increases with the time delay between samples, since it requires a sequence of two integrations with respect to time. Apart from that, inertial measurement sensors often cannot detect those small accelerations which may be caused by drift.

Underwater communication poses challenges similar to those mentioned for localisation. Bluetooth, WLAN and other wireless communication methods commonly used in air are not applicable in water [21, Ch. 1] or can be used only in a few centimetres of range [27, Sec. 3.1].

Table 2. Sensors used on autonomous underwater robots. Applications: T: teaching tool; S_L: study of bioinspired locomotion; S_S: study of bioinspired sensing; F: field projects; U: unknown. *: This includes all wireless modes of communication except for acoustic and optical. 🟧: Wireless communication, swarm synchronization; 🟦 and ⬜: localisation and navigation; ⬜: possibly more specific for obstacle detection and avoidance, target recognition, sampling or mapping; 🟩: diagnostics.

	Applications	**Radio frequency***	**Acoustic**	**Optical**	**Pressure**	**Compass**	**Flow sensor**	**Accelerometer**	**Gyroscope**	**GPS or other**	**Camera**	**Ultrasonic**	**Optical fibre**	**Infrared or laser**	**Light sensor**	**(Imaging) sonar**	**Thermometer**	**Humidity or leakage**
μ-CAT[1]	T S_L F			●	●			●	●		●						●	
Canterbury AUV [8]	F				●			●			●						●	●
Essex G9 [15]	S_L				●			●	●	●				●				
Robot dolphin [16]	S_L	●			●				●					●				
RoMAN-III [30]	S_L				●									●				
Cownose ray [17]	S_L	●			●			●	●					●				
Fish robot [18]	S_L S_S				●	●		●	●					●				●
Robotic dolphin [31]	S_L F				●			●	●	●	●			●				
CPG boxfish [19]	S_L	●				●		●	●		●							
Ostraciiform fish [32]	S_L				●			●	●		●			●				
Robotic fish [14]	S_L				●						●			●				
Boxybot [20]	S_L							●							●			
New boxfish [21]	S_L S_S	●			●			●	●		●			●				
KnifeBot [22]	S_L	●			●			●	●								●	●

[1] Refers to the robot in the state after the tasks of this thesis work are carried out.

Table 2. Sensors used on autonomous underwater robots [continued].

	Applications	Radio frequency	Acoustic	Optical	Pressure	Compass	Flow sensor	Accelerometer	Gyroscope	GPS or other	Camera	Ultrasonic	Optical fibre	Infrared or laser	Light sensor	(Imaging) sonar	Thermometer	Humidity or leakage
Sepios [23], [24]	T SL				●	●	●	●	●		●			●				●
Low-cost AUV [25]	F	●	●		●	●	●	●	●	●						●		
Micro-AUV [26]	U	●			●	●		●	●	●	●							●
MONSUN [27]	F	●	●		●	●		●	●	●	●							
MONSUN II [28]	F	●			●	●		●	●		●			●				
Southampton [10]	F	●			●			●	●	●							●	●
Squidbot mini [29]	SL	●				●		●	●		●							●

As the collected data in Table 2 reveals, the most commonly used sensors on AUVs relate to navigation, and amongst those, a pressure sensor for estimation of depth and an IMU comprising of accelerometers and gyroscopes for estimation of inertial properties. Laser or infrared sensors are often used for obstacle avoidance, in some cases accompanied by one or more cameras. Only one third of the investigated robots use a compass to determine absolute orientation, which means that others must rely on the IMU's gyroscope for relative orientation. The measurement of speed via water flow sensors seems to be rare. Not many robots employ means of positioning: GPS is not frequently used. If it is, the robot (or at least the GPS module) must be near or above the water surface to make use of the signal. Less than one third of the robots use internal leakage or humidity sensors to improve dependability, and for only very few, internal temperature sensors (for detecting overheating, for example) are mentioned. However, some sensors related to fault-detection or internal status monitoring which are rarely mentioned, such as current or voltage sensors, are not included in the table.

For some of the robots presented in this section, information on some employed sensors might not be stated in the referenced papers. This might especially apply to self-diagnostic sensors. In these cases, the table cells remain blank, although there is a possibility that the robot is actually equipped with such a sensor.

µ-CAT can be categorized as a robot with a small number of sensors, compared to others in its class, but the optical communication system seems to make it quite unique amongst small-scale AUVs.

1.2.3 Summary

The following main conclusions can be drawn from the collected data presented in Table 1 and Table 2: Small-scale robots employed in field projects are rare, and those which both serve as study objects and for real-life missions are even more rarely to find. Finally, it can be stated that robots used as a teaching tool are either rare or this use-case is not well documented. Based on the knowledge derived from the stated sources, µ-CAT is a highly versatile robot in terms of application fields, even though it does not currently focus on the study of bioinspired sensing.

All aspects mentioned in the introductory part of Section 1.2 – underwater intercommunication, low-cost production and ease of producibility, untethered operation, high computational power onboard, sufficient sensing capabilities and high manoeuvrability – are core elements in the design of µ-CAT, and many of them are improved as part of this thesis project, hence rising µ-CAT’s significance in the class of small-scale underwater robots. The revised µ-CAT design features two processors – one for mission supervision and one for interfacing with hardware components – and improved sensing capabilities. Robots with a variety of use-cases are rare – µ-CAT takes a unique position, being also able to serve as a teaching tool, mainly due to its low cost. Although other robots have been developed in the scope of students’ projects and must therefore be characterized as study tools, the interactions of future students with them are mostly limited: Each robot is a black box. µ-CAT, however, is designed to be continuously modified and improved, and it offers the possibility for future students to experiment with it on the lowest design levels, involving electronics, software, mechanics and solid modelling.

1.3 Motivation for an upgraded μ-CAT

From the side of the Centre for Biorobotics, the teaching tool μ-CAT must be more easily reproducible, so that more students can work with it at the same time. Also, the updated version of μ-CAT should be more applicable for studying problems related to robotics. In its present state, it lacks complex control, localization and fault detection, and the team of researchers and engineers at the Centre for Biorobotics have expressed their wish to have at least some of these missing aspects integrated into μ-CAT. Having a small capable robot would greatly simplify setting up research experiments and testing. This includes the possibility to run ROS [33] onboard the robot.

1.4 Personal motivation

The author of this thesis project picked up on this topic because of his deep interest in electronics engineering and his wish to combine his experience and knowledge in this field with 3D CAD (Computer-aided design) processes and with Embedded Systems software development. The task gave the opportunity to integrate aspects of all these areas into a hands-on project which would, after successful completion, provide a useful outcome for the sector of bio-inspired robotics. Apart from that, he has been working on U-CAT as his Bachelor project and was therefore already familiar with some of the features of these underwater robots and acquainted with the staff and researchers at the Centre for Biorobotics. The idea to contribute to the research sector of underwater robotics by offering a computationally powerful small-scale robot is inspiring and has been the main driver for carrying out this thesis work.

1.5 Acknowledgements

The author would like to express his gratitude towards the supervisors Roza Gkliva and Jaan Rebane for their unconditional support in the completion of this thesis work. Special thanks go to Andres Ernits, whose expertise in electronics has helped significantly during the execution of tasks related to electronics and software programming. Finally, the author would like to emphasize that this large amount of work would not have been possible to complete in the given time without the positive attitude of the whole team of Centre for Biorobotics which has generated a motivational work atmosphere.

2 Scope of this thesis project

This chapter presents all considerations and preparational works carried out before the implementation of upgrades and updates on the robot can begin. In order to clarify what exact changes need to be implemented and for which purpose, the original state of µ-CAT must be assessed, and requirements must be collected.

2.1 Initial state of µ-CAT

While U-CAT has been equipped with a camera and sensors for autonomous behaviour from the very beginning in order to be able to serve as an archaeological exploration vehicle, µ-CAT mainly adapts only the mode of locomotion from its more advanced sibling and serves as an in-lab test vehicle or to demonstrate the bioinspired locomotion concept of U-CAT. Henceforth, µ-CAT has been kept in a prototype stage of design and manufacturing.

The robot’s capabilities are limited both in terms of computational power and sensing. Even as a test vehicle for in-lab studies of U-CAT, it implements too few features of its larger sibling.

With the aim to get µ-CAT closer to U-CAT in terms of functionality while keeping its advantage of being inexpensive, small and lightweight, the following comparison between the two robots is carried out, in order to reveal potential aspects for improvement.

2.1.1 Physical properties and mechanics

The physical and mechanical properties of both robots, U-CAT and µ-CAT, are listed in Table 3. These include static properties and elements required for locomotion.

Table 3. Mechanical properties of U-CAT and µ-CAT.

	U-CAT	**µ-CAT**
Dimensions	77 x 57 x 28cm	33 x 23 x 10cm
Weight in air	18kg	1.5kg
Nominal max. depth in water	100m	10m
Flippers	4x Zhermack Elite Double 22 silicone	

While µ-CAT in much smaller and more lightweight than U-CAT, its nominal diving depth is also much smaller. This is mainly due to the materials being used – U-CAT employs an aluminium hull which withstands higher pressures than the 3D-printed parts on µ-CAT.

2.1.2 Hardware components

The following table (Table 4) lists the main hardware components on both vehicles and compares them. It becomes apparent that U-CAT has a higher degree of dependability due to self-diagnostic elements and implements more advanced controllers. While U-CAT has a specialized motor controller, µ-CAT relies on software capabilities of the Arduino microcontroller to close the loop to the Hall sensor feedback from the servo motors. Due to the presence of the camera, U-CAT offers more autonomy than µ-CAT. U-CAT employs an advanced ARM-based computer for handling the camera data, i.e., to compress video feed in real-time. The same computer takes care about the communication between robot and external clients and the translation from high-level user commands to low-level control signals or – vice versa – from low-level sensor signals to high-level monitoring messages. µ-CAT's Arduino interfaces all sensors and actuators. While U-CAT uses eight ultrasonic (2MHz) sonars for underwater object detection and avoidance, µ-CAT has no such functionality implemented. The hydrophones on U-CAT are used for adjusting the robot's orientation towards an external ping signal source. An acoustic modem allows transmission of messages under water, and outside of the water WLAN (Wireless local area network) or Ethernet cable connection can be used to communicate with the onboard computer. µ-CAT has two light sensors installed and can detect optical signals on a specific frequency (48kHz) via tone detector modules. The onboard tone detector modules filter incoming signals on that frequency. This way, the robot can distinguish between two directions of incoming light (left and right) and can hence been instructed to change its orientation.

Table 4. Hardware on U-CAT and old µ-CAT.

	U-CAT	**µ-CAT**
Power source	4x HP Compaq NX8200 laptop batteries	2x 18650 Li-ion batteries
Battery voltmeter	Voltage divider and MCP3422 ADC	Voltage divider and Arduino Mini 05 integrated ADC
Current sensor	LEM HO 25-NP	-
Motor interface	4x controller Maxon EPOS2 36/2	4x H-bridge Texas Instruments DRV8830
Motors	4x Maxon 272763 with integrated Hall sensor	4x Robbe FS 70 MG
Motor feedback sensor		4x Rotary Hall encoder AMS AS5040
Camera	PointGrey CMLN-13S2C-CS	-
Onboard computer	BD-SL-I.MX6, ARM, 1GHz	Arduino Mini 05, AVR, max. 20MHz
Humidity sensor	SHT21	-
IMU sensor	MPU-9150	MPU-6050
Localization	Hydrophones Aquarian Audio H1c	-
Obstacle avoidance	Sonars SMSF20C30F21	-
External control	Acoustic modem Applicon Seamodem	2x SFH203 photodiodes and tone detector modules
Pressure sensor	Gems 3101P-0016G-01-B-000	MS5407-AM
Temperature sensor	Included in pressure sensor, in IMU and in humidity sensor	Included in IMU (not used)

2.1.3 Voltage rails

µ-CAT's electric system operates at two different DC voltages (5V and 6V) which are generated from the battery voltage using two linear voltage regulators. The battery cells provide 2200mAh at 7.4V.

The 6V rail is used to power the servo motors through the motor driver chips and the 5V rail to supply the Arduino microcontroller, the tone detectors and the motor encoders. When connected to a USB (universal serial bus) power source, an additional 5V level becomes present which is used to power the USB-to-serial chip and which can be used as an alternative source to power the 5V circuitry.

2.1.4 Software

µ-CAT uses an Arduino Mini 05 board featuring the 8-bit microcontroller ATmega328P. The code on µ-CAT consists of various driver libraries written in C++ and of the main Arduino code. The software includes calculations for the motor motion profiles, several controllers for closed-loop motor control, depth, heading, as well as code for communication with *UWSim* (underwater simulator; used in “marine robotics research and development”[1]), to perform hardware-in-the-loop simulations.

U-CAT’s onboard computer runs Linux Ubuntu 16.04 with ROS (Robot Operating System) *Kinetic Kame* for robot control, communications and interfacing of peripheral devices. ROS is an open-source framework which offers a peer-to-peer networking structure where each piece of information needed to monitor or control a hardware component or subsystem is contained in a certain “message” [34], [33]. When connected using TCP (transmission control protocol), all messages from the robot can be read and evaluated, and the client can publish its own messages to send data. Another essential task for U-CAT is to capture and process video data and publish messages containing the results.

2.1.5 Design, manufacturing and assembly

The hull consists of three main parts – the front, the back and a transparent cylindrical Plexiglass tube in between. O-rings and gaskets are used to keep water out. Apart from these, there are several other parts which complete the physical and mechanical robot design – a battery holder, a circuit board base plate, several threaded rods to hold ballast and to keep front and back end caps in place, the parts related to transmission of torque

[1] Source: http://wiki.ros.org/uwsim

onto the fins and the fins themselves. The front and back end caps are covered with plastic domes to enhance streamlining of the robot in water.

µ-CAT is composed of parts which are available off the shelf and other parts which are custom-designed and then manufactured using rapid-prototyping (3D-printing), silicone casting, turning or manual methods. Some parts are a result of a combination of the above-mentioned properties and methods. Regarding assembly methods, µ-CAT combines revertible mechanical fastening methods with non-revertible methods involving adhesive bonds (glue or epoxy).

2.2 Contributions of this thesis

This Master thesis project's aim is to systematically improve the underwater robot µ-CAT in terms of two main aspects: the electronic hardware and the body design. The first relates mainly to functional improvements, and the second follows as a necessity, in order to host new components. Since the upgrades should not only achieve an improvement in the robot's capabilities but also in its dependability, the creation of a test software framework and highly automated test cases is also included in this thesis project.

As for the electronics part, not all implemented features are new developments. The circuit schematics for many subsystems are taken from the existing µ-CAT design – if they have proven to function as expected – or from other existing designs. Section 4.1 gives an overview. Similarly, for the body design of the robot, many features are taken as used on the original robot, and Appendix A 3.1 clarifies the source for each part's design.

In addition to the upgrade requirements and specifications listed before, the author of this thesis project has determined additional tasks to be carried out. The complete set is presented in the following subsections.

Some features are specifically tested against the requirements before implementing them, to detect if they are effective in accomplishing their purpose. Other features are implemented, and subsequent tests on the final product reveal their effectiveness.

2.2.1 Required upgrades

The required upgrades can be summarized as follows:

- improved computational capabilities;
- ability to capture and process video images;
- ability for communication using an advanced optical messaging system;
- wireless communication for uploading code and for messaging;
- more easily reproducible design of the robot.

When implementing the changes on the robot, the following constraints must be considered:

- maintain original main dimensions of the robot;
- maintain original weight to guarantee neutral buoyancy in water;
- consider roll and pitch stability of the submarine in the physical design upgrades;
- maintain position of actuators, including motors and flippers;
- maintain type and number of batteries;
- use similar production techniques of the custom-made parts;
- aim for an easy-to-assemble product;
- keep production cost below 500 € net.

2.2.2 Improving computational capabilities and capturing video data

The Raspberry Pi Zero W computer and a camera module must be integrated into the new design (see Section 4.8.1). This way, µ-CAT can now capture video data to collect visual samples, perform visual servoing or use it to run object-detection and avoidance algorithms. Both features are important aspects of an AUV (autonomous underwater vehicle or aquatic unmanned vehicle) [7, Ch. 7], and the latter would significantly improve the degree of autonomy of the robot.

With its small form factor (66 x 30.5mm) and its insignificant weight of only 9.3g (in air), the computer can be easily integrated into the confined space of the robot without posing a risk of exceeding its limits on buoyancy. The single-core Broadcom BCM2835 CPU runs at 1GHz and provides enough computing power to run certain Linux distributions. With such an operating system, the implementation of ROS on µ-CAT becomes an option. Since ROS is used on U-CAT, µ-CAT can then serve as a feasible in-lab experimentation vehicle to explore the possibilities of U-CAT. Apart from that, this framework is the de-facto standard in robot software, and students experimenting and working on µ-CAT would therefore gain a more relevant experience in the future. With the implementation of the Raspberry Pi computer as part of this thesis work, a division of tasks into levels becomes an option and a hierarchical architecture with a top-down approach can be achieved [13]. As of now, the plan is for the Arduino to handle real-time sensitive tasks, such as low-level control including interfacing sensors and motor drivers, and for the Raspberry Pi to handle high-level control, image processing and communication with the simulator.

According to commonly available specifications of the Raspberry Pi Zero W, the power consumption of this computer is 180mA at 5V. With the code to be run on the computer adding to the power consumption, it can be estimated that the power consumption will not exceed 200mA (1W) during regular operation, with WLAN enabled. This is within reasonable limits considering the available battery power of 2200mAh at 7.4V (16.28W for one hour).

Since Raspberry Pi Zero W implements a WLAN and a Bluetooth module onboard, it can establish a connection to external clients in order to receive high-level task definitions. Although WLAN is not usable while the vehicle is in diving mode, it might still be functional when it is close to the water surface and it can definitely be used to communicate with the robot right before being placed in the water.

2.2.3 Implementing internal and external communications

As for inter-circuit communications, having at least one reliable pathway between Raspberry Pi and Arduino allows the Raspberry Pi access to the devices connected to the Arduino. Via the Raspberry Pi’s WLAN module, the user can then access peripheral devices wirelessly. The design and implementation of such methods of communicating between Arduino and Raspberry Pi are also part of this thesis project’s scope (see Section 4.11 and Section 4.12).

Concerning external communications, after the changes carried out in this thesis project, µ-CAT should be able to be used in field projects. To improve ergonomics of usage, the means to instruct the vehicle while it is above the water surface or to receive real-time feedback and mission reports must be simplified. In particular, the possibility to avoid any cable connections would facilitate the handling of the robot. So far, µ-CAT can only be instructed via USB cable. The implementation of the Raspberry Pi already improves the situation, because it offers wireless capabilities, as stated above. However, with the additional implementation of a standalone Bluetooth module (see Section 4.10.2), the Arduino microcontroller can be directly interfaced, hence making potentially more power-demanding wireless connections to the Raspberry Pi unnecessary during regular operation.

An advanced optical communication system is under development [35], and integration of this (albeit not yet fully functional) module and its surrounding components is therefore part of this thesis work (see Section 4.8.2). This might enable µ-CAT in the future to be instructed using messages modulated onto an optical source and allow intercommunication between robots while being under water. It could yield a significant improvement in terms of usability of the robot outside the lab.

2.2.4 Improving dependability

This thesis project tries to improve dependability of the robot. This includes its reliability (being free from erroneous behaviour during operation), its availability (being ready to be used) and its maintainability (making it easy to debug and fix).

Availability can be improved with the implementation of a new power supply for some of the logic components. The addition of the Raspberry Pi and the camera module and the implementation of other advanced capabilities are most likely to pose a challenge on the battery lifetime and therefore reduce the maximum range of the robot’s missions. Since there is a strict constraint regarding type and number of batteries used, the only way to counteract this problem is to design a new power supply unit. The development details are presented in Section 4.6.

In terms of reliability, the new PCB (printed circuit board) design features an improved placement of components to avoid long wire runs. This makes the system less susceptible to electromagnetic noise. In addition to that, the motor controllers, the USB-to-serial chip, the circuits detecting the frequency of incoming light signals and the Arduino microcontroller with its surrounding components are embedded on the new PCB instead of being connected using pin headers. This improves reliability when exposed to vibrations due to handling and transportation and in case of collisions. Also, the power supply components are placed on the same PCB and not on separate circuit boards, hence eliminating more unnecessary wire connections which are prone to fail due to corrosion or vibration.

Regarding maintainability, the avoidance of long freely hanging wires facilitates the study and debugging of system components. The previously existing mainboard and its separate subsystems require many long wires, and these do not only complicate the debugging of malfunctioning devices but make it also challenging for µ-CAT to serve as a research vehicle for people who were not involved in its original development process. The new PCB features silkscreen print layers on both sides, which identify all subsystems and relevant components on the circuit board. Apart from that, a lot of uniquely named test pads are placed on the top surface of the board, to give access to relevant signal lines by use of an oscilloscope or a multimeter. A detailed overview on the use of these test pads and their meanings can be found in Section 4.20.

An RGB (red/green/blue) LED (light-emitting diode) is added to the PCB to allow for instantaneous visual feedback on system status and errors (see Section 4.14.1). Colour and blink codes can be used to differentiate between various messages. This feature facilitates maintainability regarding monitoring, debugging and testing. For similar reasons, several single-colour LEDs are added to the PCB, each dedicated to the indication of most common activities and statuses (see Section 4.14.2).

Several software test routines are provided as part of this thesis project which facilitate debugging and testing of components and subsystems of the robot (see Section 5.2). Software is provided to test electronic subsystems, hardware components, signal lines and communication pathways (both wired and wireless). All software interacts with the user and reports the outcome of the tests in a way understandable for the educated human operator.

2.2.5 Improving producibility and ease of assembly

In terms of manufacturing and assembly, the following design aspects of the original µ-CAT are potentially improvable:

- the electronic circuitry consists of several hand-made boards;
- the circuit board connectors are not named, which can lead to confusion when connecting components;
- some circuit boards must be individually fastened on separate Plexiglass plates to keep them in place;
- parts of the hull require rework after manufacturing:

    - the front and back end caps are painted after 3D-printing, to make them waterproof;
    - the front and back end caps require manual rework (drilling, boring, sanding etc.) after manufacturing, to make the other parts fit;
- some components are glued in place, obstructing maintenance or replacement.

The fact that the circuit boards are handmade poses restrictions to the possibilities to integrate subsystems into the electronic design in a space-efficient and reliable way. Complex subsystems, such as the Arduino board, cannot easily be integrated onto a handmade board, due to the use of small and narrow-spaced components which are placed on both sides of the board, the use of “vias” (metal rivets contacting through several copper layers of a circuit board) and the fine circuit lines which often cause problems when attempting to produce them using non-industrial etching methods. The existing board layout is not space efficient and carrying out any upgrades on the circuitry is cumbersome.

In the scope of this thesis work, many handmade components and structures to house parts are being replaced by new designs which can be reproduced without excessive manual work. This includes the new PCB and several design features of the physical structure of the robot.

µ-CAT is a robot used in a scientific research environment and must therefore be ready to be continuously improved, updated and experimented with. This includes the necessity to be able to add new components or upgrade existing ones. For students, researchers and engineers working with µ-CAT, this poses problems because of the non-automized production methods employed on the existing circuit boards. During this thesis work, this is solved by designing a completely new PCB which can be manufactured by industrial methods. The results are presented in Chapter 3. In order to further guarantee that the robot is easy to reproduce, it must be ensured that the electric and electronic parts used are up-to-date and readily available. This is carried out in the course of redesigning the PCB. The IMU and the pressure sensor (see Section 4.15) are replaced by newer versions.

The new design of the robot features consistent 3D design files as part of this thesis project’ scope, as presented in Chapter 3. These files are free from errors, contain well-structured features and updated geometries, to reduce manual rework on integrated components after 3D-printing and to reduce the need to use adhesive bonds to form physical structures.

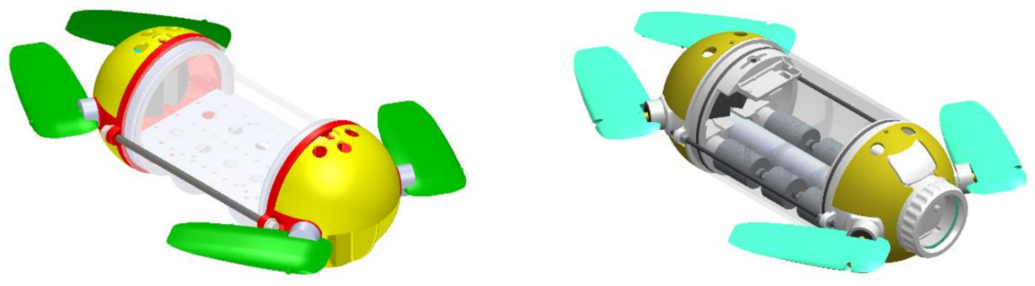
An additional 3D-printing method and different materials are used which can achieve better accuracy and highly water-resistant parts.

3 Mechanics, solid modelling and manufacturing

This chapter provides an overview on the design, manufacturing and assembly process for mechanics and hull.

According to Wang et al. [8, Ch. 2], the hull of an AUV requires a multitude of important considerations. Amongst these are the ability of the body to withstand the required environmental conditions (pressure, temperature, corrosion) and mechanical operating conditions (impacts and vibrations), as well as its practicality.

Appendix A 3.1 gives an overview on materials and their properties – but only empirical studies can finally answer all questions related to strength of materials. However, the aspect of practicality can be considered during the early design phase of the robot hull. Practicality includes ease of use (weights, shape), ease of maintenance (ease of assembly and disassembly, accessibility of connectors and user-interface elements) and ease of producibility (reduced need for manual rework).


(a) Old robot.

(b) New robot.

Figure 2. 3D CAD model of the robot (SolidWorks renderings).

3.1 Overview

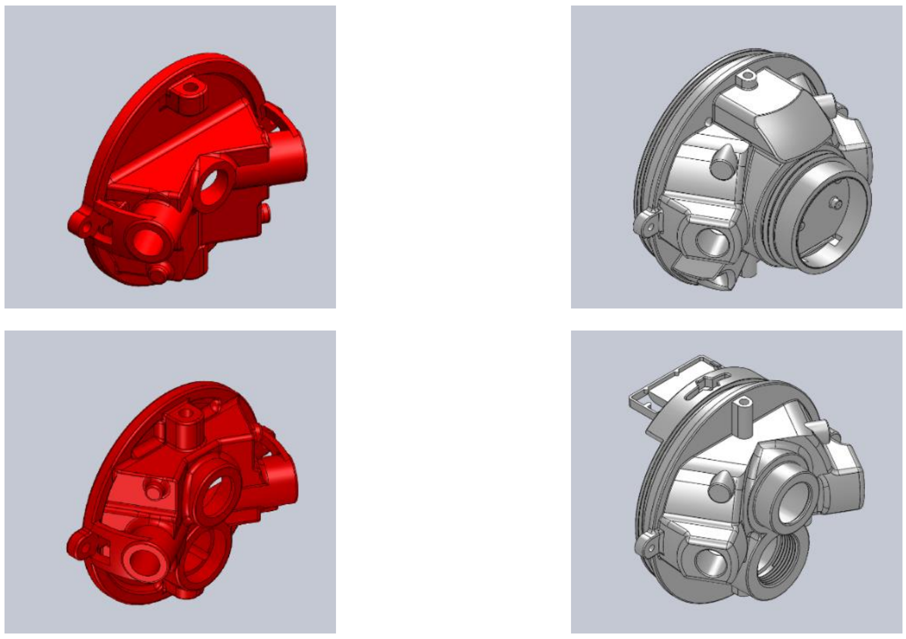
The new design of the robot is created in SolidWorks, and a comparison to the old design can be seen in Figure 2. The design file consists of 43 parts, contained in one root assembly, three main assemblies and two sub-assemblies. The files do not only include parts relevant for the physical design but also (to a certain degree of detail) electronic components, in order to plan their integration into the physical design. Some parts of the real robot are not reflected in the SolidWorks files.

The majority of work is done on the front and back end caps, to accommodate new electronic components, and on the front and back domes, protecting the aforementioned parts from mechanical impacts. As for the domes, they also reduce drag: Spherical and cylindrical or similar shapes with dome- or bullet-shaped front have a much lower drag coefficient than flat faces [36].

SolidWorks parts, their codes, names and the production methods (if applicable) are shown in Appendix A 3.1. The new design features 32 newly designed and 3 updated files, of which 32 are designed by the author of this thesis.

3.2 Main modifications

The main design work is applied to two parts: the front end cap and the back end cap. These parts are shown in comparison with the old robot in Figure 3. The yellow domes covering these parts are also redesigned to fit to the new caps.



(a) Old robot.

(b) New robot.

Figure 3. Main design changes on the robot (SolidWorks renderings). Front end cap (top) and back end cap (bottom).

3.3 Buoyancy analysis

In order to save energy during operation, it is important for the robot to be able to keep its depth in water with minimal use of energy. This can be accomplished if the robot is neutrally buoyant. However, to avoid loss of the robot in case of unexpected cease of operation, passive self-surfacing is a desired feature, which means, the robot needs to be slightly positively buoyant.

According to Archimedes' principle, an object immersed in water is subject to an upward force which equals in magnitude to the gravitational force of the amount of water it displaces. This force counteracts the gravitational force of the object itself, as Figure 4 shows. In other words, if the average density of the object is equal to that of water, it is neutrally buoyant – the two force vectors add up to zero.

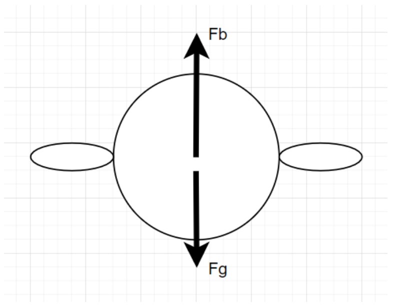


Figure 4. Gravitational force and buoyant force along the same line of action.

The following equations are scalars. The corresponding vectors with magnitude F_b and F_result act upwards (against gravitation), while the vector of F_g acts downwards.

$$F_\mathrm{b} = m_\mathrm{water} * g = \rho_\mathrm{water} * V_\mathrm{water} * g \tag{1}$$

$$F_\mathrm{g} = m_\mathrm{object} * g \tag{2}$$

$$F_\mathrm{result} = F_\mathrm{b} - F_\mathrm{g} = (m_\mathrm{water} - m_\mathrm{object}) * g \tag{3}$$

where:	F_b	[N]	buoyant upwards force acting on the object
	m_water	[kg]	mass of displaced water
	g	$[\frac{\mathrm{m}}{\mathrm{s}^2}]$	gravitational constant ($9.81\,\frac{\mathrm{m}}{\mathrm{s}^2}$)
	ρ_water	$[\frac{\mathrm{kg}}{\mathrm{m}^3}]$	density of water ($997\,\frac{\mathrm{kg}}{\mathrm{m}^3}$)
	V_water	$[\mathrm{m}^3]$	volume of displaced water
	F_g	[N]	gravitational force of the immersed object
	m_object	[kg]	mass of the immersed object
	F_result	[N]	resulting force on the object

Before production of the new robot, it is advisable to analyse its expected buoyancy characteristics, so that geometries can be adjusted or that the design provides the physical space to add fixtures for ballast elements, if needed.

To do so, each part in the SolidWorks design assembly is assigned a material with specific density. If the material type is known and exists in the SolidWorks material library, it is used as-is. However, densities of the 3D-printed part do not only depend on the type of material, but also on the density of the print – which is difficult to estimate without empirical testing. In cases where the density is unknown, the part is modelled in SolidWorks, then 3D-printed and its mass finally measured using a scale. The volume of the modelled part is obtained from SolidWorks and dividing its mass by this volume yields its density. This value is then used in a custom-made material type which is applied to the part. As a result, SolidWorks can now calculate the mass of the complete assembly (m_{object}).

As shown in the equations above (Eq. 1 – Eq. 3), it is necessary to know the effective volume of the robot to estimate its buoyancy characteristics – this means the volume enclosed within the surface which is in contact with the water. The interfacing surface surrounds all those part's solid volumes which are in contact with water. To obtain their values, some parts' geometries must be modified in SolidWorks to be completely solid (no closed cavities), while maintaining their outer surface shape. The obtained volume has the value of V_{water}. This is the volume of water displaced by the robot, and the mass of this body of water is calculated by using its density:

$$m_{\text{water}} = \rho_{\text{water}} * V_{\text{water}} \tag{4}$$

For the buoyant force of this amount of water to be equal in magnitude to the gravitational force of the object (neutral buoyancy), the mass of the object should be equal to the mass of water.

According to SolidWorks, the robot in its final design would have a mass of 1.075kg and hence be 500g too light to be neutrally buoyant. For the purpose of adjusting µ-CAT's buoyancy, it is equipped with two steel rods inside the hull to which several steel washers can be attached. These increase the mass of the robot without increasing its overall volume, hence increasing its overall density. With the mass of one M5 washer disk to be 4.58g (weight measured using real specimen), it requires 109 washers to get the robot neutrally buoyant. The physical design provides space for maximum 116 washers with a thickness of 1.7mm each. The design files include additional assemblies containing the washers. These are especially useful for analysis of roll stability, carried out in Section 3.4.

As an alternative to washers, the design includes box-like fixtures to hold other types of weight-adding elements inside the hull. These fixtures are two equal 3D-printed parts with several compartments for holding the weights. The boxes slide onto the battery tube and link to other features of the existing geometry to prevent rotation or linear movement during operation.

3.4 Roll stability

An object which is neutrally buoyant is not necessarily free from rotational moments acting on it. Due to the cylindrical base shape of µ-CAT, it might especially tend to rotate around its longitudinal axis (for example, when being disturbed by flows), so it seems reasonable to minimize instability about this axis especially. The undesired tendency can be counteracted by distributing the densities of materials on the robot such that its centre of gravity is vertically below the centre of buoyancy when the robot is in the natural position.

The centre of buoyancy of an object is coincident with the centre of gravity of the volume of water which it displaces. Since µ-CAT is completely submerged in water, the volume of displaced water has a shape equal to that of the whole vehicle. Assuming uniform density all throughout the volume of water, the centre of buoyancy of the vehicle is coincident with its centroid – the geometrical centre of a solid body with the same outer surface shape as the whole vehicle.

The buoyant force acts at the centre of buoyancy, while the gravitational force acts in opposite direction at the centre of gravity. If the centre of buoyancy is horizontally offset from the centre of gravity, they form a couple – a moment acts on the object. The magnitude of this torque depends on the magnitude of the opposed forces and the length of the moment arm between the two force vectors – i.e., the distance between them.

As described by Lautrup [37, p. 47], "[in] a fully submerged rigid body, for example a submarine, [… if] the centre of gravity does not lie directly below the centre of buoyancy, but is displaced horizontally, for example by rotating the body, the direction of the moment will always tend to turn the body so that the centre of gravity is lowered with respect to the centre of buoyancy. The only stable equilibrium orientation of the body is where the centre of gravity lies vertically below the centre of buoyancy. Any small perturbation away from this orientation will soon be corrected and the body brought back to the equilibrium orientation […]" (because frictional forces will prevent it from swaying indefinitely).

When designing an underwater robot, it should therefore be ensured that its centre of gravity lies vertically below its centre of buoyancy when the robot is in upright (neutral) orientation. This way, it can be guaranteed that the robot will return to this orientation if it gets rotated by external forces. The design must consider the following:

- any horizontal offset between the two points must be minimized;
- the centre of gravity must be shifted below the centre of buoyancy;
- the vertical distance between centre of buoyancy and centre of gravity determines roll stability – it should be long enough to counteract a rotation caused by external influences, but short enough for the robot to be able to actively roll around its axis.

The analysis of the locations of the centre of buoyancy and of the centre of gravity are carried out in SolidWorks as part of the design phase. To find the centre of buoyancy, the assembly (originally containing parts of different densities) is converted into solid filled objects of equal and homogenous densities. The position of the centroid is then computed by SolidWorks. This point is coincident with the centre of buoyancy, as explained above.

3.5 Structural components

The structural base frame of the physical design is formed by a set of six parts, as shown in Figure 5.

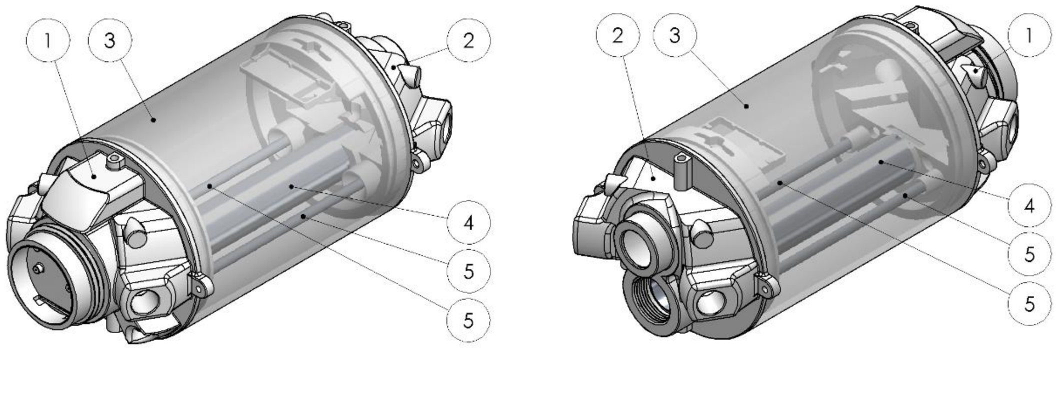


Figure 5. Parts forming the physical base structure (SolidWorks renderings). (a) Isometric front view; (b) isometric back view. (1) Front end cap; (2) back end cap; (3) transparent Plexiglass cylinder; (4) battery compartment; (5) a pair of M5 steel rods.

These parts define the base geometry of the robot and provide its initial mechanical stability. To additionally prevent them from twisting due to mechanical tolerances, supplementary parts are inserted (not shown above):

- two Plexiglass ballast dividers, sliding onto the battery compartment and M5 steel rods;
- two M3 steel rods pushed through provided holes on the exterior of front and back end cap and fastened with nuts.

The ballast dividers also give structural support against torsion of the main PCB, which is clamped between front and back end caps by inserting it into provided grooves. For ease of assembly, the grooves for the PCB and the holes for inserting the M5 steel rods are opening up towards the parts, so the parts centre themselves in these geometries during insertion. The outer M3 steel rods, when closed with nuts, additionally prevent the front and back end cap from sliding apart.

3.6 Waterproofing

The robot consists of three main parts surrounding the interior: the front end cap, the back end cap and the Plexiglass tube. Not only these three parts must be in waterproof contact with each other, but also all other components inserted into the front and back end caps which require openings on these 3D-printed parts: the camera, the photodiodes, the battery cap and the USB plug. The following subsections explain how waterproofing on these parts and components is accomplished.

The camera is located underneath a removable 3D-printed screw cap which holds a transparent flat circular Plexiglass piece and presses it against the body of the end cap. A custom-made silicone gasket is placed between end cap and camera cap to accomplish waterproof sealing. The gasket is cast using the method described in Section A 3.2.2.

Each photodiode is surrounded by a custom-made silicone-cast (see Section A 3.2.2) seal which is squeezed between the photodiode and the wall of the insert hole.

An M25-to-M20 steel reducer is screwed into the 3D-printed thread of the back end cap. Sealing between reducer and end cap is accomplished using Teflon tape along the thread. The steel battery cap has a M20 thread and fits onto the reducer. It comes off-the-shelf with an O-ring included which is squeezed between battery cap and reducer, protecting from leakage.

The USB plug consists of several parts – the main body and the cap. The cap is screwed onto the main body and seals itself by pressing the included rubber gasket onto the body when being tightened. The USB body is sealed against the 3D-printed end cap by pressing an O-ring between body and end cap when tightening the plastic hex nut on the USB body’s thread.

The transparent Plexiglass tube is sealed against the front and the back end caps with two O-rings which each sit firmly in a dedicated groove on the respective end cap.

The fin shafts each reach the motor clutch located in the interior of the hull through a nitrile oil rubber seal which prevents water from entering. The shaft sleeve holds the nitrile oil seal, and this sleeve has a very firm fit with the hull. For additional waterproofing, epoxy may be used to seal any gap between sleeve and hull.

3.7 Manufacturing

The new design incorporates several 3D-printed parts. Some of them require high geometrical accuracy, to host electronic and mechanical components and to be in close contact with parts to avoid leakages. These parts are the front and back end cap and the camera cap. They are produced using SLA (stereolithography). Other 3D-printed parts are the front and back dome, which do not need a tight fit, since they are flooded. For them, FDM (fused deposition modelling)[1] is the method of choice, since the involved material is more lightweight and cheaper. Front and back dome are supposed to protect underlying parts from damage, and in the case of collisions, they will break and must be replicable without causing high additional cost.

With µ-CAT being used in depths not more than 10 meters, the hydrostatic pressure exerted on the robot is small. Tests will reveal if the new material used for 3D-printing keeps its structural integrity during operation. Similarly, only empirical testing over a prolonged period can reveal the robot's ability to withstand other mentioned external influences. With the employed low-cost rapid-prototyping production method, the choice of materials is very limited. The existing µ-CAT has proven to withstand required pressures with 3D-printed materials. Waterproofing was accomplished by applying a rubber paint coating onto the robot end caps. The SLA 3D-printing material has higher density, is less porous and hence more water-resistant. The painting cover can therefore be omitted.

In-lab equipment is used for manufacturing the end caps, the camera cap, the domes and some of the gaskets. Processes and equipment are described in Appendix A 3.2.

3.8 Results and discussion

The new solid body design integrates all electronics, electromechanical and mechanical components in ways that facilitate assembly and handling of the robot. This chapter presents the results of the work carried out in SolidWorks. A photo of the new assembled µ-CAT can be seen in Figure 6.

[1] "Fused deposition modelling" and its abbreviation "FDM" are registered trademarks of Stratasys.

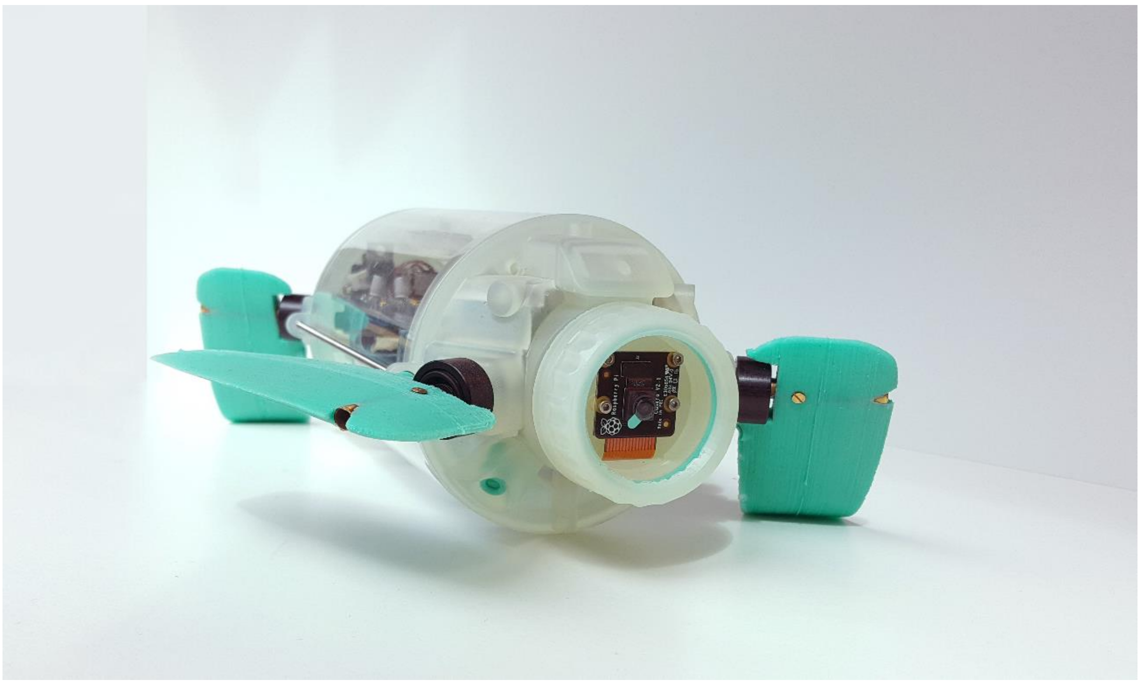


Figure 6. Assembled µ-CAT in its new design. Front and back domes are not present.

An overview of components integrated into the physical design is given in the following (Figure 7 and Figure 8).

Especially the redesign of the parts presented in Section 3.2 leads to several advantages over the previous robot design:

- Motors can be inserted more easily:
  - do not require manual rework (cutting) in order to fit;
  - slide in easily into the end caps (enough tolerances).
- Some parts need less or no glue in order to be integrated:
  - reed switch slides into the holder and needs no glue;
  - nut for holding the magnet clips into the holder and needs no glue;
  - 3D-printed thread in the back end cap provided for battery cap reducer.
- Main PCB does not require bolts to be fastened;
- Geometries are more accurate, hence mechanical fitting and tolerances are improved overall.

The front end cap now additionally holds two new parts, namely the camera module and the messaging LEDs' PCB. Since the camera does not have its own enclosure, a watertight protection mechanism is designed which allows for an easy attachment of the camera module and routing of its cable. The mechanism consists of a threaded cap, a seal and a transparent plexiglass insert (see Section 3.6). The cap needs no further fastening and holds firmly when screwed onto the body thread. The PCB for the LEDs does not require screws but is clipped into the holder.

The back end cap now additionally holds the new Bluetooth module (see Section 3.8.1) which clips into the holder.

The SolidWorks design files are improved regarding the following aspects:

- Materials with correct densities are assigned to all parts, therefore allowing precise calculation of the robot's mass and buoyancy characteristics.
- Errors and warnings present in features of the original files are eliminated.

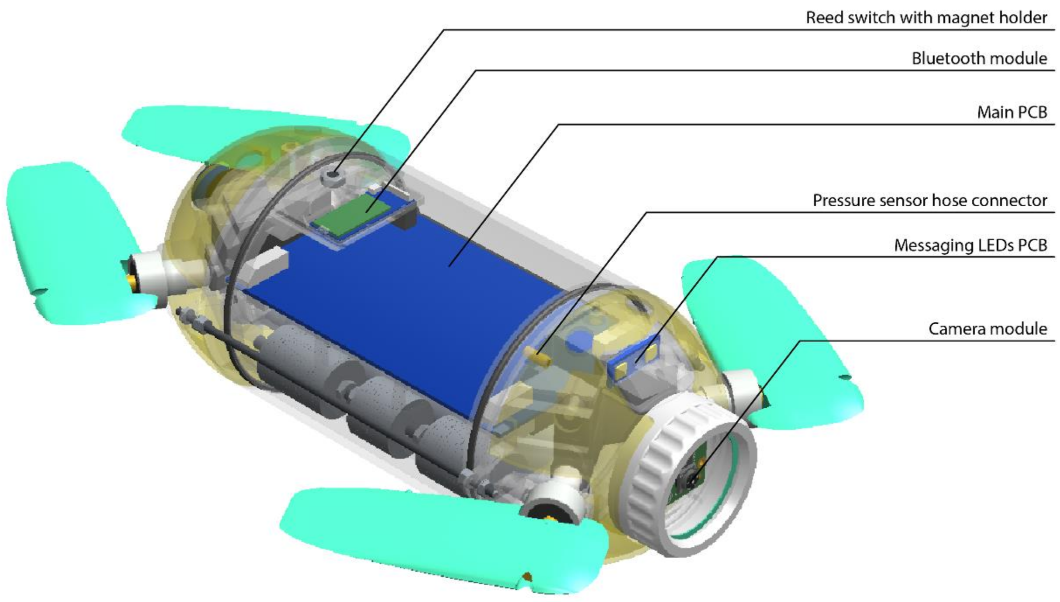

Reed switch with magnet holder
Bluetooth module
Main PCB
Pressure sensor hose connector
Messaging LEDs PCB
Camera module

Figure 7. Components integrated in physical design (SolidWorks rendering). The front and back dome and the front and back end cap are shown as half transparent.

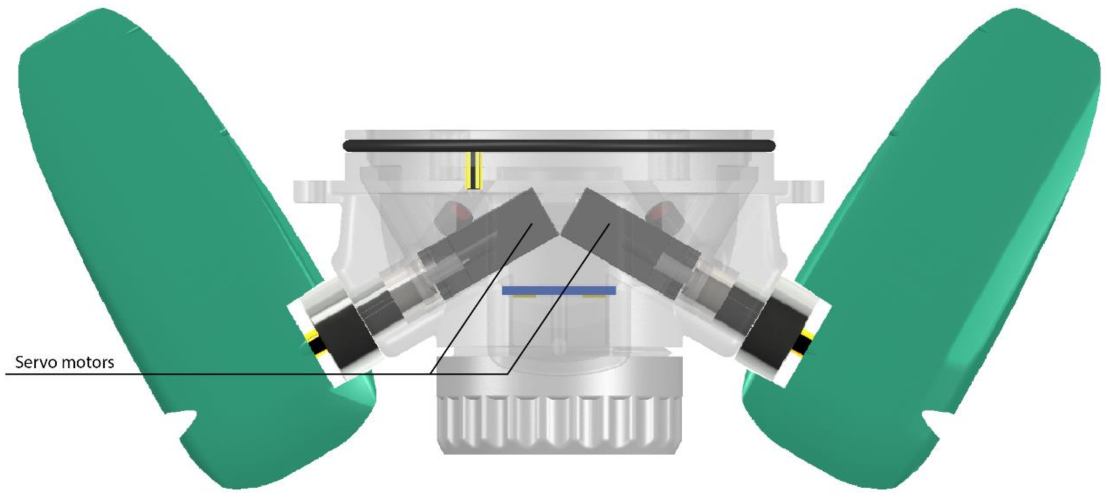


Figure 8. Motors integrated in physical design. (SolidWorks rendering). Top view of front cap assembly. The front end cap is shown as half transparent, the front dome is not shown.

3.8.1 Integration of electrical components

As part of the new design, one main aim was to improve integration of electrical components. Improvised methods of holding the components in place (glue etc.) are to be avoided. To give some examples of how these aims have been met, the integration of two components is presented in the following subsections, and Figure 9 gives an overview.

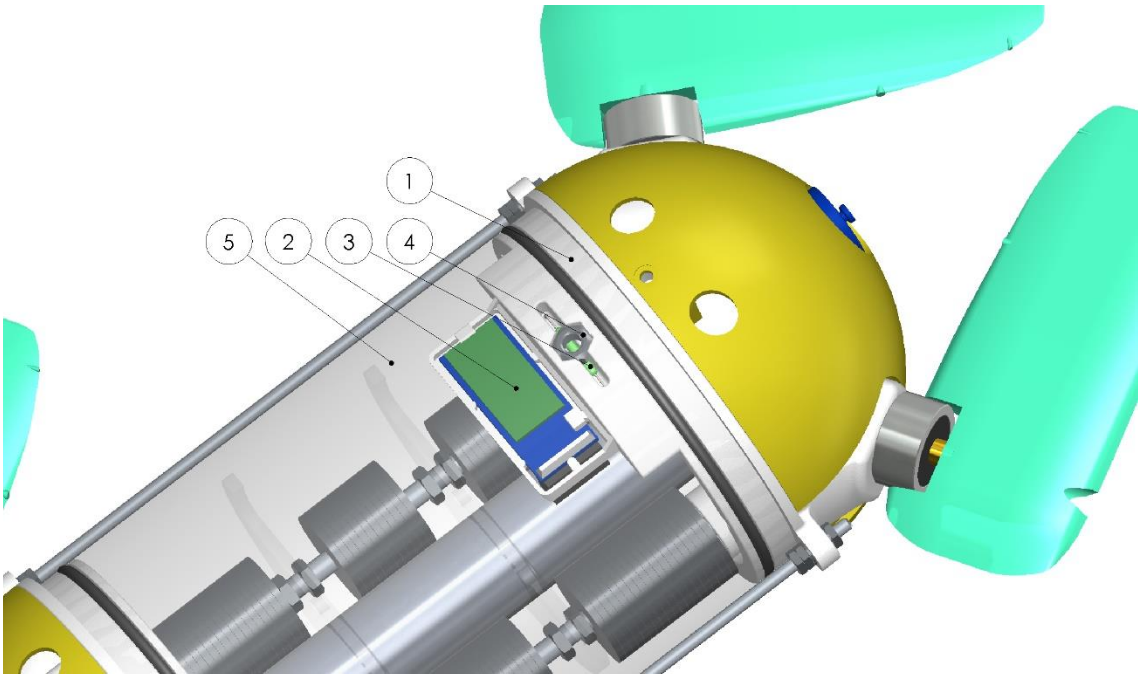


Figure 9. Integration of Bluetooth module and reed switch (SolidWorks rendering). (1) Back end cap; (2) Bluetooth module; (3) Reed switch; (4) M5 hex nut; (5) transparent Plexiglass tube.

Placement of the Bluetooth module

The back end cap provides a holder which is suitable for the HC-05 breakout board or for the custom-made Bluetooth PCB. Assembly and disassembly are simple, since it is held only by clips. The positioning of the Bluetooth module is on the highest possible place inside the hull. When the submarine is placed in the water and not operated, it stays on the surface, as proven in Section 3.8.3. In this state the user can communicate with the vehicle using Bluetooth without removing it from the water. It has been found experimentally that the Bluetooth device is reachable if it is less than 5cm below the water surface. With the knowledge obtained in Section 3.8.3, the distance between the Bluetooth antenna and the water level can be found. It is estimated to be only 3.66mm (see Figure 10), which is expected to provide good connectivity to the Bluetooth module. More importantly, due to the cylindrical shape of the Plexiglass tube, there is no water directly above the antenna, which is supposed to further ensure good connectivity.

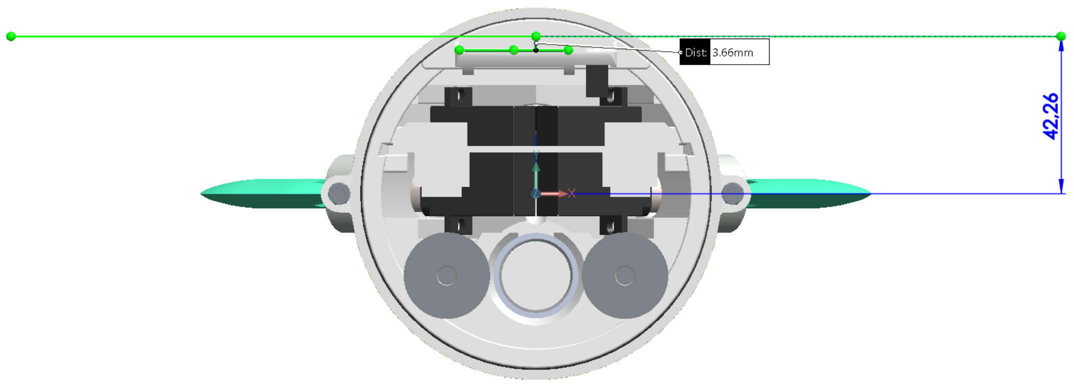


Figure 10. Bluetooth module positioning (SolidWorks rendering). Cross sectional cut through the robot assembly, showing water surface (upper green line) and Bluetooth module antenna plane (lower green line).

Placement of the reed switch

In contrast to the old µ-CAT design, the reed switch is now properly integrated into the physical body. Physical space is provided in the 3D-printed back end cap to host the reed switch, exactly matching its dimensions. The legs are bent down by 90° and routed through dedicated channels. They can be bent around the lower part of the holder, and a connector or wires can be attached. A hex nut is placed on top of the reed switch which snaps into a dedicated recess and holds the magnet when placed on the outside of the Plexiglass tube above the reed switch.

3.8.2 Roll stability

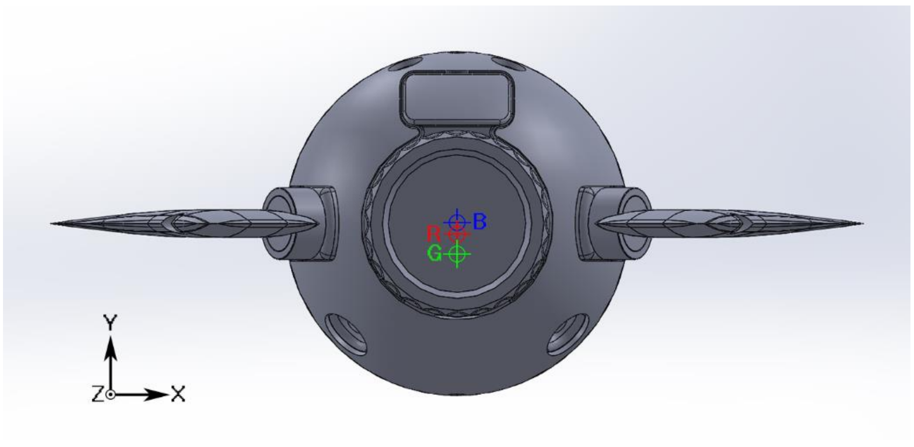


Figure 11. Analysis of buoyancy (SolidWorks rendering). Position of centre of buoyancy (B), centre of gravity without washers (R), centre of gravity with 100 washers (G) on x-y plane.

Figure 11 shows the result of the analysis carried out in SolidWorks. The batteries are included in the computations. Their placement in the lower half of the vehicle lowers the centre of gravity, therefore working in favour of roll stability. By adding 100 steel washer disks onto the provided rods, the centre of gravity moves roughly 5.6mm downwards, hence increasing roll stability. The centre of buoyancy B then lies significantly above the centre of gravity G, offset vertically by 9mm. The horizontal offsets of all centres of masses from the vertical centre plane along the x axis are negligible, therefore no torque is expected to occur about the longitudinal axis (z) when the vehicle is in its natural position (as shown).

In the most extreme case, the vehicle may be rotated by $\pm 90°$ about its z axis. The moment acting on the vehicle about its z axis is then:

$$M_z = F_\mathrm{g} * |BG| = 0.1351\mathrm{Nm} \tag{5}$$

This torque will turn the vehicle until the natural position is restored. While turning, the moment diminishes, with the moment arm shortening as a sine function of the angle of rotation, as the robot gets closer to its neutral orientation.

3.8.3 Self-surfacing

The recommended number of washers (100 pieces) is smaller than the calculated number, to achieve slightly positive buoyancy and have the robot slowly floating upwards in a case of malfunction (i.e., when it cannot actively surface). Using equations Eq. 1 – Eq. 3 in Section 3.3, a resulting upwards force of $F_\mathrm{result} = 0.437\mathrm{N}$ can be expected with 100 washers added. The number of washers is variable, and it not only determines the time it takes for the submarine to passively surface, but also the final depth position it takes when surfacing. This final position yields the equilibrium between gravitational and buoyant force, which is reached when the mass of the displaced water equals to that of the vehicle including the washers. Using SolidWorks, it is found that this mass is $m_{\text{object,corr}} = 1529.71\text{g}$. The mass of displaced water when completely submerged is slightly larger ($m_{\text{water}} = 1574.27\text{g}$). The equilibrium condition occurs when approximately 97% of the robot is submerged (Figure 12). This is the position in which the robot eventually comes to rest without activity.

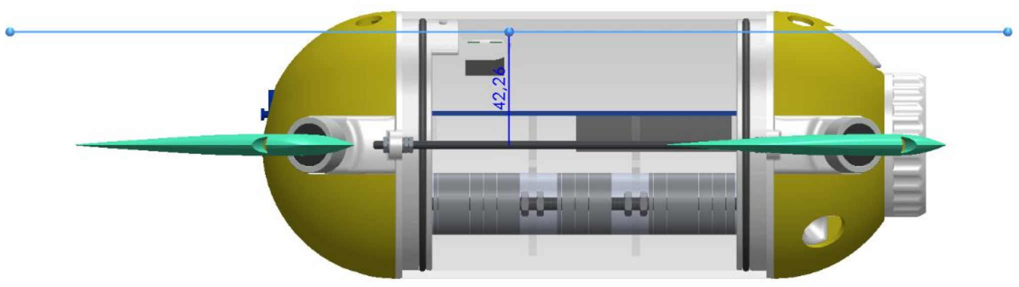


Figure 12. Estimation of water surface level (SolidWorks rendering). Reached in force equilibrium, with 100 washers of ballast added.

3.8.4 Ease of production and assembly

Manual rework on many parts is improved, and assembly of parts is easier than before, as stated in the beginning of Section 3.8.

The procedure for disassembling the robot in order to reach the PCB is shown in Appendix A 3.3. Back end cap assembly, battery compartment, ballast rods with ballast washers, Plexiglass weight dividers and outer steel rods form one assembly and do not need to be taken apart for disassembling the robot. Cables connecting components in the front end cap assembly to the PCB are long enough, so the front end cap assembly can be slid out of the Plexiglass tube and the connectors on the PCB can be reached and disconnected.

3.8.5 Future work

In the future, the holder for the Bluetooth module in the back end cap should be improved, to avoid too thin walls around the module. Due to the 3D-printing process, walls tend to bend, which might compromise a tight fit of the Bluetooth PCB.

The physical design of each insert channels for the photodiodes should be slightly improved, to hold the photodiode and its surrounding seal in place from both sides, thus preventing it from sliding inwards or outwards.

Another improvement concerns ease of assembly: To facilitate self-centring of the PCB in the grooves of the front end cap, the Plexiglass weight dividers which slide over the battery compartment will have grooves to hold the PCB from the sides. The PCB needs to slide into these parts freely, so electronic components too close to the edge of the PCB must be moved.

4 Electronics

The work related to electronics includes the development of new circuit schematics, the modification of existing circuit schematics and the design of a PCB implementing all circuitry. These tasks are carried out in Autodesk Eagle, a commonly used CAD software for industrially produced circuit boards.

Future versions of the manufactured PCB should include silkscreen-printed references to parts in the schematic diagrams, to facilitate manual soldering. This, however, may lead to a loss of clarity on names of connectors, LEDs and test pads which are required for use-cases after the manufacturing of the PCB has finished. An alternative way is to provide a separate printed sheet with positions and names of all parts marked which can be placed on top of the PCB between the soldering steps.

4.1 Main modifications compared to original µ-CAT

The following table (Table 5) gives an overview on all implemented features covered in this chapter, indicating for each of them to which group it belongs – newly developed or existing – and, if it belongs to the latter group, whether it has been taken from the given design "as-is" or whether it has been (at least in parts) updated ("U") by a newer version or has been modified ("M"). The design of a feature is marked as new if it is not copied from existing products and if it is not derived from schematic layouts provided to the author of this thesis. Therefore, some features are marked as new although they have been present on the original µ-CAT robot but used some different working principle, and some features are marked as previously existing although they have not been used on the previous design of µ-CAT but were taken from external sources.

It should further be noted that features marked as "Existing" might take over the existing schematic diagram but the routing of their circuit lines might be modified in the process of implementing their layout to the new PCB.

Table 5. Features implemented in the scope of this thesis project. M: modified; U: updated. *: The signals provided on the PCB consider the possibility to implement a modified design of the Bluetooth module circuitry. More details in Section 4.10.1. **: The existing design itself is not implemented, but only pin headers are provided. FTDI: Future Technology Devices International.

Feature	**New**	**Existing**
Provision of different voltage rails		M
Powering 5V rail alternatively through USB		●
Logic power supply unit for 5V logic	●	
Logic power supply unit for 3.3V logic	●	
Motor power supply unit		M
Raspberry Pi		**
Arduino Mini		M
Camera		●
Bluetooth module	*	●
Shared UART bus (incl. multiplexing, level shifting)	●	
Communication from Arduino to Raspberry Pi (incl. level shifting)	●	
Tone detectors		●
Frequency-shift keying module		**
OR logic between input signals for frequency-shift keying	●	
RGB LED	●	
Various status LEDs	●	
IMU circuit		U
Pressure sensor circuit		U
Motor drivers circuit		●
Motor feedback circuit		●
Voltage measurement circuit		●
FTDI (USB-to-serial)		●
Reed switch		●
Test pads	●	
Design of PCB	●	

4.2 Overview of components, subsystems, signals and busses

The new electronic circuitry can be broken down into several functional blocks. Most of these elements are individually identified on the new PCB by their names. The following table (Table 6) gives an overview of all implemented elements. For each element, its placement on the new PCB is indicated (top or bottom side), and it is shown on which busses it actively participates and which GPIO (general-purpose input/output) lines it provides or requires. The latter covers only those lines which are routed to digital GPIO pins of one of the microcontrollers. Bus and GPIO signals can be either on a 3.3V or on a 5V level.

The subsequent block diagram (Figure 13) accompanies Table 6 and clarifies the connection topology of all functional elements which participate on one of the busses or employ GPIO signal lines.

Table 6. Functional elements, their placement side and their signals. FSK: frequency-shift keying.

	PCB side		**Signals used**						
Element	**Top**	**Bottom**	**I²C (3.3V)**	**I²C (5V)**	**SPI (5V)**	**GPIO (3.3V)**	**GPIO (5V)**	**UART (3.3V)**	**UART (5V)**
USB connector		●							
Logic power supply (3.3V, 5V)	●	●							
Motor power supply (6V)	●						●		
Raspberry Pi Zero W		●	●			●		●	
Photo diodes connectors		●							
Tone detectors	●	●					●		
Frequency-shift keying module	●						●		
Switch between sources of FSK	●								
External LEDs connector	●								
Arduino	●			●	●		●		●
FTDI	●							●	
Bluetooth connector	●					●		●	
Multiplexer		●				●			
Onboard RGB status LED	●					●			
IMU	●		●			●			
Pressure sensor	●		●						
Motor drivers		●		●					
Motor voltage connectors	●								
Motor encoder connectors		●			●				
Voltmeter		●							
Reed switch connector	●								
Multipurpose level-shifter	●								
I²C level shifter		●							
Test pads	●								

Note that the SPI (serial peripheral interface) bus includes GPIO signals (chip select; “CS”) by default.

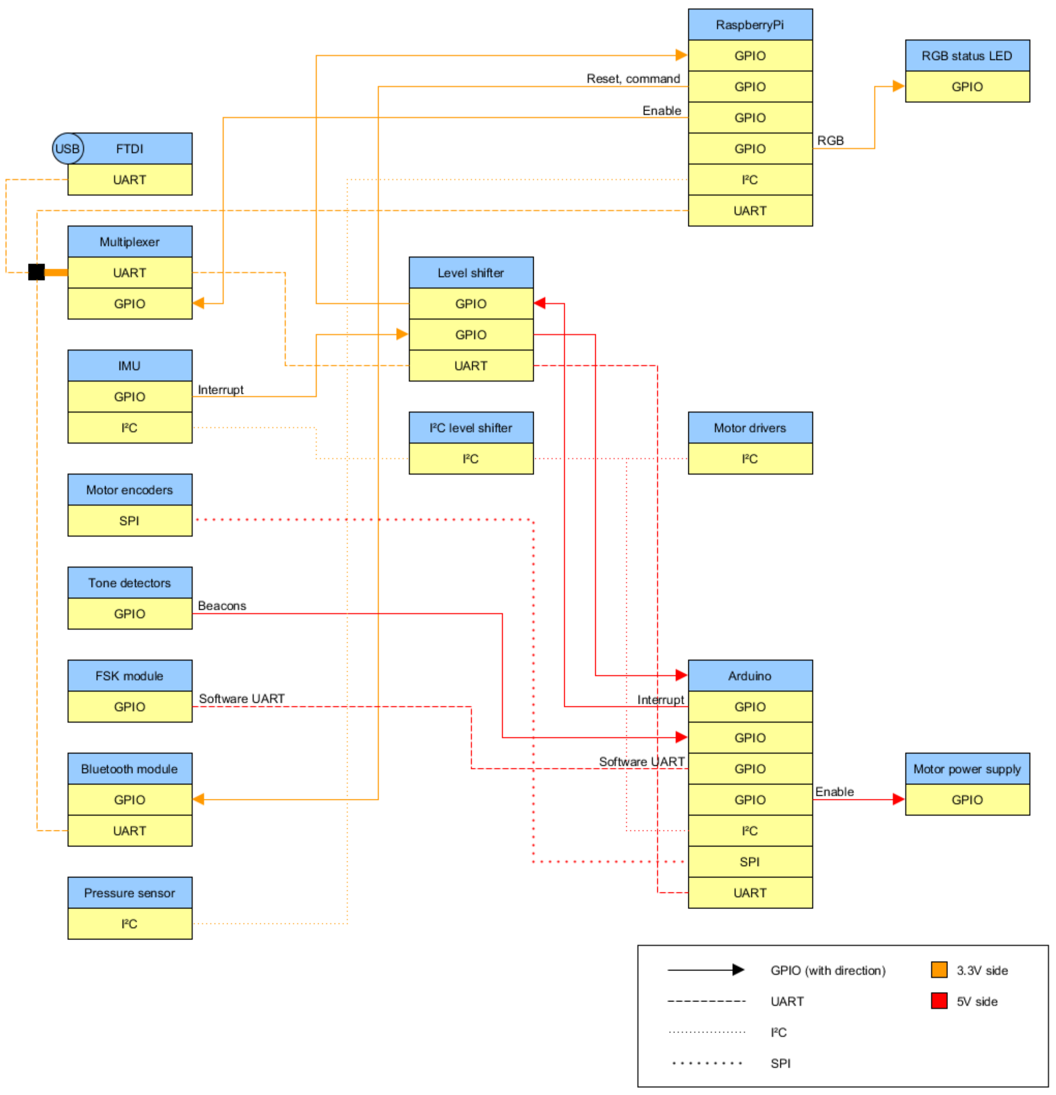


Figure 13. Functional elements and their signals (GPIO and busses).

4.3 Resistor value and power calculations on LEDs

When choosing the current-limiting resistor for any LED used in the new design, the operating principle of the LED in general must be understood. An LED operates at a given nominal forward voltage with a corresponding forward current, as given in the datasheet. The values vary depending on type and colour of the LED. The brightness of an LED depends on the operating point, but the absolute maximum forward current must not be exceeded when driving the LED.

The following equations are used to calculate the required resistor value for an LED and the power consumption of the network comprising of LED and resistor:

$$R = \frac{U_{\mathrm{CC}} - U_{\mathrm{LED}}}{I} \tag{6}$$

$$P = U_{\mathrm{CC}} * I \tag{7}$$

where:			
	R	[Ω]	resistance of in-series resistor
	U_{CC}	[V]	voltage across network of resistor and LED
	U_{LED}	[V]	forward voltage of LED (from datasheet)
	I	[A]	forward current through resistor and LED
	P	[W]	total power consumption of LED and resistor

The calculated values of resistors may not be available on the market. In such cases, the closest available resistor value is chosen. With a resistor value different from the initially calculated one, the forward current differs and can be calculated as follows:

$$I_{\mathrm{real}} \approx \frac{U_{\mathrm{CC}} - U_{\mathrm{LED}}}{R_{\mathrm{chosen}}} \tag{8}$$

This value is just an approximation because the voltage drop across the LED (U_{LED}) also changes with the current through it. The approximation is sufficient if the chosen resistor value is close to the previously calculated one.

Results of these calculations for all LEDs are given in Appendix A 2.1.

4.4 Voltage rails

The main purposes of the existing voltage rails (see Section 2.1.3) do not change during the upgrade of the robot. A new 3.3V rail is implemented for powering several sensors and other components. A second 3.3V rail becomes available which provides voltage through the voltage regulator onboard the Raspberry Pi computer. This is used directly to power the camera, the multiplexer and the general-purpose level shifter's low side. The only device driven by this voltage through the GPIO pins of the Raspberry Pi is the RGB LED. The 3.3V generated within the USB-to-serial chip is not used. The 5V rail additionally powers the Raspberry Pi module, the frequency-shift keying module including the attached OR logic and the high sides of the level shifters. USB power can still be used as an alternative source for the 5V rail. The 6V rail now does not only power the motors, but also the high-power messaging LEDs installed on the front end cap.

4.5 Supplying 5V alternatively through USB

A circuit is implemented which allows to override the battery voltage and use USB supply for the 5V logic instead. This allows programming of the Arduino microcontroller without having any batteries inserted. The circuit consists only of the Schottky diode D2 with maximum forward current of 3A and maximum reverse voltage of 20V. According to specifications [38], [39], the voltage of the USB supply can vary between 4.75V and 5.5V. If D2 is forward-biased (conducting), this voltage drops by 0.5V on the load side across the diode. If the reed switch is open (no magnet placed) while powering through USB, the Buck converter is enabled, and its output provides 5V (assuming sufficient battery voltage). In this case, D2 is reverse-biased (Anode potential equal to or lower than Cathode potential) and no power is provided through USB at all. Consequently, if the robot should be powered through USB, the reed switch must be closed (magnet attached). If the magnet is not in place, it must be expected that no voltage will be supplied from USB.

4.6 Improving power supply

One task is to extend the time span of uninterrupted operation of the vehicle. The original robot can remain operational for approximately four hours. Being powered by two Li-Ion cells of type 18650 in series, each with a capacitance of 2200mAh at 3.7V, the totally provided energy by a fully charged set of batteries is 16.28Wh. Assuming the vehicle remains operational until the batteries are fully drained (which, in reality, it does not), a mean power consumption of max. 4.07W for µ-CAT can be derived.

With the new components added, especially the Raspberry Pi computer with camera (see Section 4.8.1) and the relatively powerful messaging LEDs (see Section 4.8.2), the battery lifetime is expected to be shortened. In order to counteract this undesired effect, a closer look must be taken on the power supply units used in µ-CAT.

For estimating battery lifetime, the power consumptions of individual consumptions P_i could each be calculated as the product of the supply voltage of a component and its (typical) current consumption during operation. The total power consumption of the completely assembled robot can then be calculated by adding individual power consumptions P_i of all components:

$$P_i = U_{\mathrm{CC},i} * I_i \tag{9}$$

$$P_{\mathrm{in}} = \sum_{i=1}^{n} P_i \tag{10}$$

The real power consumption of a component depends largely on its mode of usage. A more practical approach for estimating the average power consumption of µ-CAT is to run the final µ-CAT software and measure the current consumption I_{B} right after the batteries (voltage U_{B}). The power consumption is then calculated as:

$$P_{\mathrm{in}} = I_{\mathrm{B}} * U_{\mathrm{B}} \tag{11}$$

In order to achieve a perceivable saving in energy consumption, it is obvious to aim for the voltage rails to which the heaviest consumers are attached. These are the $5\mathrm{V}$ rail, mainly due to the Raspberry Pi computer and the camera with their total current demand during operation of approximately $150\mathrm{mA}$ on that rail, and the 6V rail because of the four servo motors with a total stall current of up to $2\mathrm{A}$ (all motors stalled) and a current demand of approximately $550\mathrm{mA}$ during operation, according to previously conducted measurements. In the executed experiments for selecting the voltage rail which should be powered by the new power supply unit, the high-power messaging LEDs on the $6\mathrm{V}$ line are not considered, because they had not been chosen at the time when the experiments were designed.

4.6.1 Buck converter instead of linear series regulator

Buck converters and voltage regulators are used to convert between two DC voltage levels, in particular to change a DC input signal to a lower level. These converters generate an output voltage on a stable value, independently from the input voltage provided and the output current drawn. When converting from a higher to a lower voltage, voltage dividers made of a network of two resistors are another alternative. Their disadvantage lies in the high heat dissipation due to the voltage drop across the resistors and the current drawn from the supply through the upper resistor. Apart from that, voltage dividers consume current even with no load connected.

In linear voltage regulators, the excess power, due to the voltage drop between input and output voltage, is transformed into heat and therefore wasted. The higher the voltage drop across the regulator, the higher the heat dissipation. In fact, the dissipation of heat follows directly from generating the desired voltage drop. If the input current of the voltage regulator is equal to its output current, its efficiency can be calculated as follows:

$$\mu = \frac{U_{\mathrm{out}}}{U_{\mathrm{in}}} = \frac{P_{\mathrm{out}}}{P_{\mathrm{in}}} \tag{12}$$

The dissipated heat is then:

$$P_{\mathrm{heat}} = (U_{\mathrm{in}} - U_{\mathrm{out}}) * I = P_{\mathrm{in}} - P_{\mathrm{out}} \tag{13}$$

Typically, linear voltage regulators are ICs (integrated circuits) comprising of a Darlington transistor array or MOSFET (Metal-oxide silicon field-effect transistor) configuration and an operational amplifier including an output transistor which drives the Darlington array, along with overheat and short circuit protection. The transistors are operated in their active, i.e., linear region, where their resistance is linear to the respective input signal. The operational amplifier compares the actual output voltage to a fixed reference voltage and changes the resistance of the driving output transistor which, in turn, determines the amount of current provided through the Darlington or MOSFET network [40]. Obviously, driving transistors in their active region leads to dissipation of heat due to their internal resistance, and the heat increases with the required current on the load side.

Buck converters are known for their higher efficiency compared to linear voltage regulators, mainly because they produce less heat. The buck converter is controlled by a transistor switch and a diode and stores energy in an inductor. The transistor is only driven in its saturated region (fully on) or fully off, and therefore does not dissipate so much heat (only during the moment of switching). Since there is ideally no loss of power in the form of heat (if the transistor's switching heat dissipation is neglected), the input power is equal to the output power, and a step-down in voltage therefore leads to a step-up in current.

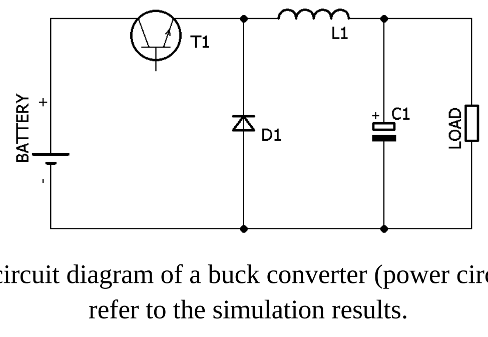


Figure 14. Basic schematic circuit diagram of a buck converter (power circuit only). The marked points refer to the simulation results.

As shown in Figure 14, the basic buck converter circuit consists of a switching transistor (T1) and a flywheel circuit, comprising of diode D1, inductor L1 and capacitor C1. The operating principle is based on two phases which repeat indefinitely at high frequency. These two phases are (1) the one where the transistor is conductive (on) and (2) where the transistor is not conductive (off). In the beginning of phase 1, since the inductor opposes the change of current flow, it restricts the emerging flow of current through it. During this time, it stores energy in a magnetic field, and the voltage across the inductor is highest because the capacitor has reached the end of its discharge cycle (lowest possible potential on the positive side of the capacitor). During phase 1, current is provided to the load and to the capacitor through the inductor. Over time, while the magnetic field builds up, the current flow through the inductor increases and the voltage drop across the inductor decreases due to the potential across the capacitor rising.

$$u = U_\mathrm{B} * e^{-t*\frac{R}{L}} \tag{14}$$

$$i = \frac{U_\mathrm{B}}{R} * \left(1 - e^{-t*\frac{R}{L}}\right) \tag{15}$$

where:	u	[V]	momentary voltage drop across the inductor
	U_B	[V]	supply voltage from battery
	e		Euler's number (≈ 2.718)
	t	[s]	time
	R	[Ω]	load resistance
	L	[H]	inductance of the inductor
	i	[A]	momentary current through the inductor

But the capacitor will not reach the full supply voltage because phase 2 will take place before it is charged up completely. During phase 2, the transistor is open (off) and the flow of current into the inductor is rapidly stopped. Current to the load is still provided, since the inductor releases current stored in its magnetic field in the direction in which it has been flowing previously, hence through the load. [41] The capacitor helps supply the load additionally. It smoothens at all instances the voltage ripples caused by the linearly changing current supply from the inductor.

The switching frequency of the transistor is fixed, but the duty cycle D varies depending on the required voltage drop $\Delta U = U_\mathrm{out} - U_\mathrm{in}$, in particular:

$$U_\mathrm{out} = D * U_\mathrm{in} \tag{16}$$

$$\Delta U = U_{in} * (D - 1) \tag{17}$$

The transistor gate driver (not shown in above schematics) takes care of matching the duty cycle to the output voltage requirements by comparing the instantaneous output voltage with an internal reference voltage created by the standard silicon bandgap. The inductance of the inductor must be chosen according to the switching frequency of the transistor and the power demand of the load. Knowing the duration of a pulse, the output power can be expressed as energy provided per pulse. This is the energy to be transferred during every switching pulse of the converter. The datasheet of a specific buck converter IC which includes already the switching transistor, as it is in the case of the module used in this project, usually provides the required value for the inductor.

Disadvantages of buck converters compared to linear voltage regulators are the higher noise level they generate due to the switching of the transistor, their higher cost factor and the larger requirement of components for assembling the circuit, leading to a larger footprint on the board. The switching noise may be critical in some applications where highly sensitive analogue measuring equipment is used but can mostly be reduced to an acceptable level by using appropriate capacitors on the output of the converter and at the inputs of noise-sensitive circuitry after the converter. The higher cost factor is still minimal and therefore not significant. The larger footprint of the circuitry may pose a challenge in space-critical applications, but µ-CAT offers plenty of room for hosting a large mainboard – only the original version of the robot did not use the available space sufficiently.

Design of the buck converter circuit

After carrying out several experiments as shown in Appendix A 1.1, the buck converter LM2596S is chosen to replace the original voltage regulator on the 5V logic voltage supply rail of µ-CAT.

This chip provides up to 3A output current, which leaves enough headroom with the expected current of less than 1A on the 5V rail. Not driving the buck converter at its limit improves output voltage stability and decreases noisiness. The chip features an inverted enable pin (“ON/OFF”) which can be used to switch the device off. The quiescent current consumption of the disabled (idle) buck converter chip is expected to be around 80µA, but not more than 200µA under normal temperature conditions [42, Ch. 7.9]. Applying a conservative calculation, the robot can remain in switched off state for 458 days, with initially fully charged batteries.

The current drawn by the enable pin is very small, so the reed switch can be used directly on that pin, to switch the robot on or off.

The buck converter circuit is designed according to the recommendations in the datasheet [42, p. 1] and is – apart from the values of some capacitors – equal to the design of the buck converter used in the experiments (see Figure 15).

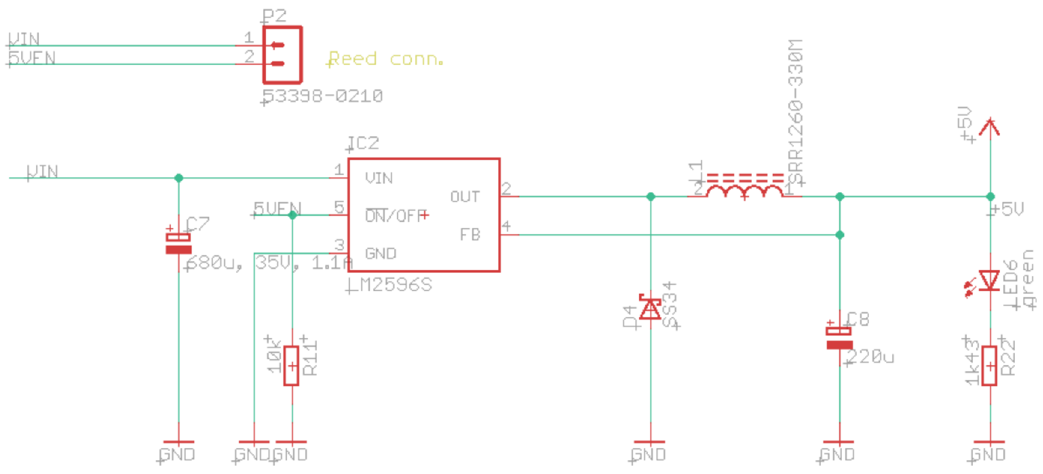


Figure 15. Implemented buck converter circuit.

Integration of the reed switch

The robot can be turned on and off without the need of having a physically accessible switch on the outside of the hull, which would require additional waterproofing. Instead, a reed switch is used inside the robot which reacts to the magnetic field of a magnet placed close to it, on the outside of the vehicle. This reed switch has been, in the original design, glued to the inside of the hull, and an iron hex nut has been used to hold the magnet in place.

In the new design, the same reed switch and the same principle for holding the magnet are used, but the physical integration of the switch is improved (see Chapter 3.7). As for the wiring of the switch, it connects the positive battery voltage signal (“VIN”) to the enable (“ON/OFF”) pin of the buck converter chip LM2596S. Since the enable pin of the 6V voltage regulator uses inverted logic, the buck converter chip is switched off when the reed switch is closed, i.e., when the magnet is placed. In order to keep the buck converter chip switched on while the reed switch is open, its enable pin is pulled down to “GND” through a 10kΩ resistor (R11 in Figure 15). In order to turn the robot off, the magnet must be placed.

4.7 Motor power supply unit

The motors are supplied with 6V from the MIC29302WT voltage regulator, taken over from the original µ-CAT design. This chip takes the battery voltage as its input. It provides max. 3A output current [43, p. 1], which is enough for the motors even when they are stalled. The dropout voltage is as low as 370mV when drawing the maximum output current of 3A, otherwise even lower. This means that the device provides a stable output voltage if the input voltage is at least 6.37V – which is given even with drained batteries. The chip features an enable ("EN") pin which can be used to switch the device on or off. This pin is pulled low through resistor R26 and routed to the Arduino microcontroller's pin "PC3". A high signal on this pin enables the voltage regulator. Having control of this pin through the Arduino allows for the robot to be tested and debugged without providing power to the motors.

4.8 Capability upgrades

In the following subsections, the implemented upgrades related to the improvement of functional abilities of µ-CAT are listed. This includes the Raspberry Pi Zero W, the camera and the advanced optical messaging system.

4.8.1 Integration of Raspberry Pi Zero W single-board computer and camera

In order to be able to upgrade the new design with future versions of the Raspberry Pi Zero module, this computer is integrated by providing a pin header, rather than copying the Raspberry Pi layout into the PCB layout. The 40-pin header is placed in a position that allows the camera ribbon cable to lead towards the camera module without deflecting from the geometrical centre axis of the robot and to avoid a too narrow bend radius of the ribbon cable when passing the front motors.

The camera module (Raspberry Pi camera version 2.1) connects to the Raspberry Pi module with a ribbon cable. Therefore, the electrical connections to the camera are provided solely by the Raspberry Pi computer, and no connection must be provided on the PCB itself.

4.8.2 Implementation of advanced optical messaging

The complete connection topology of the optical messaging system is shown in Figure 16 and Figure 17 and described in the following subsections.

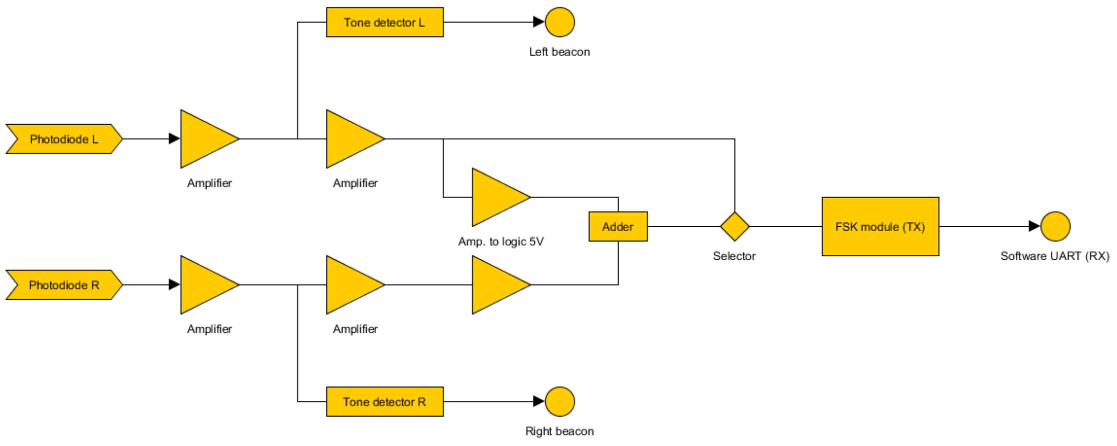


Figure 16. Block diagram of topology for FSK demodulator (incoming signals).

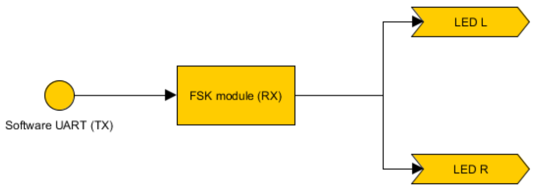


Figure 17. Block diagram of topology for FSK modulator (outgoing signals). “LED L” and “LED R” symbolize the messaging LEDs.

Basic components

The photodiode converts light into electrical current, and this current is amplified by the subsequent circuitry, then fed into the tone detectors and into the FSK (frequency-shift keying) modules. Two photodiodes are used to distinguish between optical signals to the left and to the right of the robot. The photodiodes are installed on the front hull of the robot, and two simple pin header connectors are provided on the PCB, as close as possible to the photodiodes, to keep wire runs short.

Each tone detector circuit detects beacon signals on 48kHz coming from one of the photodiodes. Depending on which photodiode is active, the robot can distinguish between a signal coming from the left or from the right. At the core of the tone detector circuitry, the tone decoder IC LM567 is implemented. It provides a transistor switch which is closed whenever it receives an input signal within the specified passband [44, p. 1]. The tone detectors' outputs (“LEFTBEACON” and “RIGHTBEACON” in Eagle) are routed to two Arduino digital input pins.

Integration of frequency-shift keying circuit

As an application report by Texas Instruments states, “[in] telecommunications and signal processing, frequency modulation (FM) is encoding of information on a carrier wave by varying the instantaneous frequency of the wave. Digital data can be encoded and transmitted via carrier wave by shifting the carrier's frequency among a predefined set of frequencies—a technique known as frequency-shift keying (FSK).” [45, p. 1] In the case of µ-CAT, the source of information are external light signals, and the receivers are the same photodiodes used for the beacon signal detection. Decoded messages are sent to the Arduino using GPIO lines and the *SoftwareSerial* library (software-emulated version of UART). The module can also encode signals fed from the Arduino board using *SoftwareSerial*, and they are sent out in the form of light using the powerful messaging LEDs installed to the front end cap of the robot (see Chapter 3.7).

The frequency-shift keying circuit has still been in development at the time of implementation on the new PCB. Changes to the circuitry might still be required. Therefore, the module is regarded as a black box. However, since its purpose and interface are known, it can be implemented in the form of an add-on component.

Switching between signal sources

The frequency-shift keying module can be set to react either on only the signals supplied from the left photodiode or from both. If the jumper “RCVMIX” (indicated as “RCV” on the PCB) is in default state, the frequency-shift keying module is fed by both photodiode signals (left and right). When selecting the mixed signal, the amplified individual signals from the two photodiodes are routed through an adder (logic OR) circuit, comprising of three N-channel MOSFETS (in two chips type FDC6401N). The first two transistors are routed in parallel configuration, with common drain, and each of them receives one of the incoming signals on its gate. Whenever at least one of them switches on, the common output signal (drain) is low. This output signal is routed to the gate of the third transistor which inverts it.

The FSK module expects a digital (binary) message signal, which means that the MOSFETs are supposed to act as switches (saturated mode). The MOSFETs need a voltage level of 2.5V or higher (maximum 12V) between gate and source to be on [46, p. 2] and a voltage level of less than 0.5V to be off [46, p. 3].

The tone detector circuit is known to work with at least (typically) 20mV (RMS) input voltage signal [44, Ch. 8.4], which corresponds (for a sine wave signal) to a peak-to-peak level of 28.28mV:

$$U_{\mathrm{pp}} = U_{\mathrm{RMS}} * \sqrt{2} = 20\mathrm{mV} * \sqrt{2} = 28.28\mathrm{mV} \tag{18}$$

This value is therefore defined as the minimum required signal peak-to-peak amplitude for the system to be operational. Each channel of this signal (called "TO_RCVR_L" and "TO_RCVR_R") is amplified twice before reaching the adder circuitry.

The first amplifier (IC5B and IC9B in the schematic diagram) has a voltage gain of approximately 40. This corresponds to a valid peak-to-peak value of minimum 1.205V. The output capacitor (C33 and C48, respectively) achieves that the centre of the output signal (its average) is 0V – it is an AC signal. Accounting for this offset, the extreme (peak) DC signal voltage values for the minimum valid signal on that amplifier output can be calculated:

$$U_{\mathrm{low}} = -\frac{U_{\mathrm{pp}}}{2} = -\frac{1.205\mathrm{V}}{2} = -0.6025\mathrm{V} \tag{19}$$

$$U_{\mathrm{high}} = \frac{U_{\mathrm{pp}}}{2} = \frac{1.205\mathrm{V}}{2} = 0.6025\mathrm{V} \tag{20}$$

The upper peak value must always be high enough to guarantee a saturated on-state of the first MOSFET stage in the adder circuitry, and the lower peak value must always be in a valid range to achieve that the first MOSFET stage is off. The lower half of the signal can be clipped off (minimum peak value will be 0V) since it is insignificant when interpreted as a frequency-shift-keyed message. The upper peak value should be greater than or equal to 2.5V, as mentioned above. To achieve this, an additional op-amp is used before the adder circuitry. The dual-channel operational amplifier TS972IPT is initially chosen and used in non-inverting configuration, as shown in Figure 18. With the negative input rail referenced to ground, the lower half of the input signal is clipped off, and the upper half has a magnitude of 0.6025V for the minimum required valid signal.

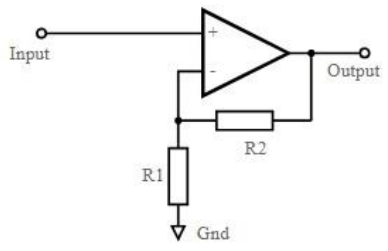


Figure 18. Operational amplifier in non-inverting configuration.[1]

The required voltage gain is:

$$A_{\mathrm{V}} \geq \frac{2.5\mathrm{V}}{0.6025\mathrm{V}} = 4.15 \qquad (21)$$

The values of resistors R1 and R2 in above schematic diagram define the amplification factor (voltage gain) of the circuit. The equation is:

$$A_{\mathrm{V}} = 1 + \frac{R2}{R1} \qquad (22)$$

A ratio of R2 to R1 can be found which accomplishes the required voltage gain. The resistors should be in the range of kiloohms, to avoid excessive currents and overheating of the operational amplifier chip. Available resistor values are $\mathrm{R1} = 10\mathrm{k}\Omega$ and $\mathrm{R2} = 39\mathrm{k}\Omega$, which yields a voltage gain of $A_{\mathrm{V}} = 4.9$. This drives the subsequent MOSFETs safely into saturation, even for the valid worst-case signal:

$$U_{\mathrm{out}} = A_{\mathrm{V}} * U_{\mathrm{in}} = 4.9 * 0.6025\mathrm{V} = 2.95225\mathrm{V} \qquad (23)$$

The problem with the chosen op-amp TS972IPT in the given scenario is its inability to guarantee linear amplification for input voltages below $U_{\mathrm{DD}} + 1.15\mathrm{V}$ (common mode input voltage range [47, Ch. 3]) – which is in this case $1.15\mathrm{V}$ (for $U_{\mathrm{DD}} = 0\mathrm{V}$). The smallest valid input signal, as discussed above, ranges from $0\mathrm{V}$ to $0.6025\mathrm{V}$. A more suitable operational amplifier IC with the same pinout and footprint is MCP6022, which has a common-mode input range of values starting from $-0.3\mathrm{V}$ [48, p. 3].

[1] Image source: https://www.watelectronics.com/wp-content/uploads/Non-Inverting-Operational-Amplifier-Circuit.jpg

Integration of messaging LEDs

The OSRAM DURIS S8 GW P9LR32.EM is a white high-power LED. The new design implements two of them, mounted on a separate PCB (including their resistors), which is installed in the front end cap (see Chapter 3.7). This PCB is connected to µ-CAT's main PCB using a wire harness with plugs for two-pin SMD connectors on both ends. (The messaging LED PCB has been designed by Jaan Rebane.)

Electrical characteristics, chosen resistor values and calculated power consumptions can be found in Appendix A 2.1. Calculations are based on equations given in Section 4.3 and values from the datasheet [49, p. 11]. The LED is driven below nominal voltage, at 5.5V. In case this product becomes unavailable, similar versions exist for replacement [50], [51].

4.9 Integration of Arduino Mini 05

The Atmel/Microchip ATmega328P microcontroller is the CPU used on the Arduino Mini 05 board. For a reliable and shock-proof operation, integration of the Arduino layout directly to the new PCB instead of providing pin headers for the Arduino module seems preferable. This decision is reasonable in the case of the Arduino microcontroller, since no significant updates on the 8-bit AVR can be expected in the future. In regard to this topic, a more extensive discussion on space-efficiency, maintainability and reliability is made in Section 4.21.

4.10 Communication with the outside world

In this subsection, features are presented which relate to the possibility to interface the microcontrollers. This includes debug messages sent from the microcontrollers to an external client, instructions and program code sent from an external client to the microcontrollers.

The new design provides the following communication interfaces:

- USB serial from and to Arduino, converted to UART by the FTDI[1] chip;
- Bluetooth serial from and to Arduino, converted to UART by the Bluetooth module;
- WLAN connection with Raspberry Pi via its onboard WLAN module;
- Bluetooth connection from and to Raspberry Pi via its onboard Bluetooth module.

The last two features are not further described in this subsection, since they are integrated properties of the Raspberry Pi module. However, it should be noted that a connection to the Arduino microcontroller can be established through the wireless interfaces provided by the Raspberry Pi, because the Raspberry Pi and the Arduino can communicate with each other through UART (see Section 4.11).

4.10.1 Integration of the FTDI chip

The design of the original µ-CAT features a Sparkfun breakout board [52] holding the FTDI chip FT232. The same chip is implemented in the new design, with the circuit design derived from the original breakout board. It translates between the USB protocol (which offers half-duplex communication via two symmetric signal lines) and the UART buffered full-duplex communication standard. The UART port of the chip is routed to the Arduino UART port via a general-purpose level shifter to translate between the internal logic voltage level of the FTDI chip (3.3V) and the logic voltage level of the ATmega328P (5V). Details on this level shifter can be found in Section 4.19.

4.10.2 Integration of the standalone Bluetooth module

The main purpose of the Bluetooth module is to connect µ-CAT wirelessly to a computer or to a smartphone in order to get debugging messages from Arduino while the robot is operating. Since the Bluetooth module cannot communicate while the robot is under water, it is desirable to place it as high as possible inside the robot's body (see Section 3.8.1), so it can communicate while the robot is floating near the water surface. This allows debugging right before diving and immediately afterwards and does not require the operator to get the vehicle out of the water. Placing the Bluetooth module near the

[1] Abbreviation for the company name Future Devices Technology International; commonly used for USB-to-UART translator chips.

highest point inside the hull makes it necessary to connect it to the PCB using a wire harness. For this purpose, a surface-mount 8-pin connector is placed on the PCB to which the Bluetooth module can be easily plugged. The other side of the wire harness is connected to the Bluetooth module using pin headers.

The chosen Bluetooth module is the EGBT045-MS, placed on a breakout board similar to HC-05. Since the original board HC-05 does not expose all relevant pins of the EGBT045-MS, a self-designed breakout board is used instead (PCB design by Jaan Rebane).

The following table (Table 7) lists the provided signals on the new PCB and their intended connection to the Bluetooth module EGBT045-MS or, wherever applicable, their connection to the breakout board HC-05. Exact pin numbers for the signals connecting to the microcontrollers can be found in the tables of provided test pads in Section 4.20.

Table 7. Routings between Bluetooth module and new PCB. Left arrow: to the Bluetooth module; right arrow: from the Bluetooth module. *: Through resistor 2.2kΩ on HC-05 breakout board.

Bluetooth module			**µ-CAT main PCB**		
HC-05 pin	**EGBT045-MS pin(s)**	**Direction**	**Signal name (Eagle)**	**Connections**	**Purpose**
STATE	25/32	→	BT_STATE	LED “BTS”; Raspberry Pi	Get information on connection status
RXD	2	←	BT_RX	Arduino TX	Send data
TXD	1	→	BT_TX	Arduino RX	Receive data
GND	13		GND	Common GND	Logic ground
VCC	12		+3V3	3.3V supply	Logic supply 3.3V
EN	34*	←	BT_CMD	Raspberry Pi	Select role (Master/Slave); put into command mode
-	24/31	→	BT_STATE1	LED “BTS1”	Read status of the Bluetooth onboard LED
-	11	←	BT_RST	Raspberry Pi	Reset Bluetooth module

The “BT_CMD” signal, in combination with a reset of the module through “BT_RST”, can be used to put the Bluetooth module into AT command mode. In this mode, it can be configured using AT commands being sent via the UART interface. It should be noted that, in order to configure the Bluetooth module, Arduino and Raspberry Pi must collaborate: The Raspberry Pi controls the GPIO pins of the Bluetooth module, while the Arduino can access its UART port. The “EN” pin of the HC-05 breakout board is wired to the “CMD” pin (34) of the EGBT045-MS through a buffer resistor. Pushing the button on HC-05 performs the same action as setting the “EN” pin high (see Figure 19).

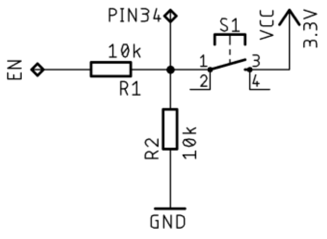


Figure 19. Routing of enable pin on the HC-05 breakout board. “PIN34” refers to pin 34 of the EGBT-045MS Bluetooth module, “EN” and “GND” are exposed pins on the HC-05 breakout board.

By default, the module is in a Slave mode and appears to be transparent, acting like a Serial connection cable, as stated in the Bluetooth profile SPP (serial port profile). It needs no reconfiguration; entering the command mode is usually unnecessary. Therefore, the module can be operated without making use of those pins which are not exposed on the HC-05 breakout board.

4.11 Sharing the UART bus

The UART bus offers one-to-one communication between two devices. One line establishes the unidirectional connection for messages sent from the first device to the second and the other line vice versa. The protocol implements full-duplex buffered communication at various baud rates.

As described in the previous sections, several devices must now be able to communicate with the Arduino microcontroller via the UART bus. This calls for an extension of the native protocol implementation. In the new µ-CAT design, a multiplexer chip is used to switch between the three communication partners of the ATmega328P microcontroller. Similar approaches have been used by other engineers in the past [53]. The ATmega328P supports baud rates up to 250kbps, and the employed multiplexer has been verified to handle baud rates up to 115.2kbps – the value commonly used when flashing program code to the ATmega328P microcontroller. Since all three communication partners operate on a 3.3V logic level (and Arduino on 5V), a level shifter must handle the voltage translations (see Section 4.19).

The following block diagram (Figure 20) illustrates the connection topology of the shared UART bus, including the pin designations used on the devices.

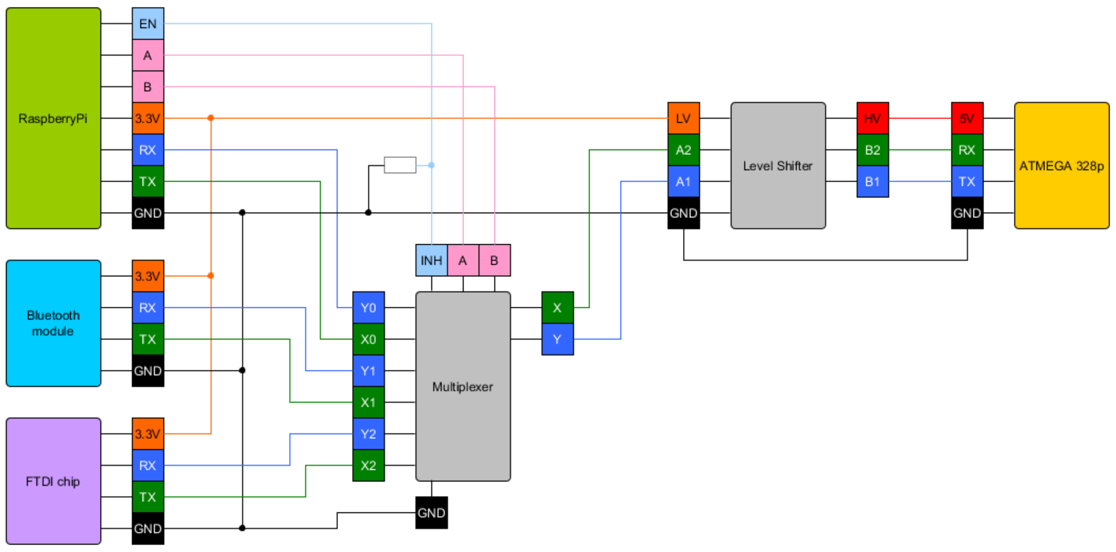


Figure 20. Circuit diagram for multi-device UART connectivity to ATmega328P.

4.11.1 Integration of the multiplexer chip

A multiplexer is a universal logic functional element used in electronic designs to route one or more incoming signals to one or more selected pins, depending on the address setting. Analogue multiplexers handle analogue voltage ranges and route them to the corresponding outputs using MOSFET switches. Digital multiplexers expect discrete signals with the level either being high or low. Due to their different internal architectures, an analogue multiplexer can have its pins put into high-impedance mode – meaning that the routing is completely interrupted. Digital multiplexers, however, have a defined default output state, which makes it inappropriate for use with the UART bus. The digital multiplexer also has a dedicated input and output side, and it expects the signal on the input being the signal source (driving), while the output is the signal sink (driven) – meaning that a digital multiplexer is a unidirectionally operating device. In the topology employed to realize a shared UART bus, the device to which the RX and TX lines of various clients need to be routed (the ATmega328P) is the driving side of one of these lines (namely, the TX line), and therefore a digital multiplexer cannot be used.

The analogue switch multiplexer MC14052B is implemented in the new circuit design of µ-CAT. It is digitally controlled (through address and enable), but the signals to be routed can be of any nature and – more importantly – of any direction. Depending on the 2-bit address setting, the chip routes each signal in its singular pair of common pins to a corresponding signal in one of four other pairs of pins. The ATmega328P UART pins RX and TX are connected to the common pins of the multiplexer, and the UART interfaces of the three potential communication partners – the FTDI chip, the Raspberry Pi and the Bluetooth module – are connected to the other routing pins of the multiplexer.

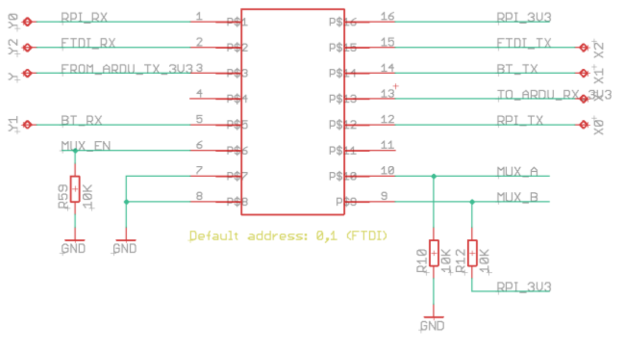


Figure 21. Multiplexer schematic diagram.

Figure 21 shows the routing of the multiplexer pins. The routing table for the multiplexer is given in Appendix A 4.3.

4.12 Communication line from Arduino to Raspberry Pi

A GPIO status line is provided between Arduino physical pin 25 and Raspberry Pi physical pin 7. It can be used by the Arduino microcontroller to send simple messages (in the form of sequences of digital high and low states) to the Raspberry Pi. This way, the Raspberry Pi can be informed about critical or exceptional system states on low-level components which are monitored by the Arduino. The logic level translation from Arduino's 5V level (signal called "ARD2RPI_HV") to the Raspberry Pi's 3.3V level (signal called "ARD2RPI_LV") is done by the level shifter described in Section 4.19. The status LED "HWINT" lights up whenever this signal line is high.

4.13 Performing a reset of the Arduino microcontroller

It is possible to reset the Arduino microcontroller via the integrated pushbutton. However, this button is not accessible when the robot is fully assembled. Therefore, an alternative way of resetting via the GPIO pin 11 on the Raspberry Pi is provided by a signal line called "ARDU_RST_LV" which opens the digital NPN transistor T1, connecting "GND" to Arduino's reset pin. Resetting the Arduino through the Raspberry Pi makes it possible to upload code to the Arduino using the Raspberry Pi's UART port. To perform this upload, the Raspberry Pi must set the multiplexer UART routing accordingly (see Section 4.11).

4.14 Implementation of status LEDs

With the aim to improve user interaction with µ-CAT during operation and debugging, several status LEDs are integrated on the PCB. Each of them carries a printed name on the circuit board. In the following subsections, Table 8 lists the RGB LED and Table 9 all single-coloured LEDs including their functions. Electrical characteristics, chosen resistor values and calculated power consumptions can be found in Appendix A 2.1.

All LEDs except for LED11 are placed on the top layer of the PCB which points upwards when installed on the robot.

4.14.1 Implementation of RGB status LED

The RGB LED CLP6C-FKB consists of three separate LEDs in one die. Each of them has its individual forward voltage and needs its individual resistor, as calculated by the equation above.

The purpose of the RGB LED is to have a high flexibility in being able to colour-code messages from the robot to the user, covering a wide range of different topics, including system status and warning or error codes. The definition of colour codes used on the RGB LED is up to the engineer and can be programmed through code executed on the Raspberry Pi.

For highest flexibility and best PWM (pulse width modulation) output, the natively capable PWM pins [54] on the Raspberry Pi are used for the RGB LED.

Table 8. Onboard RGB status LED.

Eagle name	**Printed name**	**Colour**	**Indication**
LED5-G	RGB LED	green	User-defined, implemented by software.
LED5-R		red	
LED5-B		blue	

The resistors for the LED are chosen such that, assuming a 100% PWM duty cycle, each LED inside this chip is driven at about 50% of its nominal forward current, which is expected to yield approximately 50% of the nominal luminous intensity of the RGB LED [55, p. 5]. As tests have shown, this is sufficient for low-distance signalling.

As a side note, depending on the required distance between communication partners, the RGB LED offers potential for usage as a messaging device from robot to robot. Since µ-CAT is now equipped with a capable camera, colour codes emitted by the RGB LED on a robot could be used as signals for communication between robots working in a swarm.

4.14.2 Implementation of single-colour LEDs

For the single-colour LEDs, a strict colour-coding scheme is used: Green LEDs indicate regular mode of operation, yellow or blue LEDs indicate special events (messages or certain status flags), and red LEDs indicate faults.

In order to save power, the in-series resistor values for single-coloured LEDs in µ-CAT's design are chosen such that the LEDs are driven at low brightness, at around 10% of their mid-range forward current values. Experimental tests have shown that this brightness is enough for a clear indication. For all single-coloured LEDs, the desired forward current is about 2mA.

Table 9. Onboard status LEDs.

Eagle name	**Printed name**	**Colour**	**Indication**
LED1	3V3	green	On: 3.3V voltage regulator provides output.
LED2	TX	yellow	Toggle: data flow from FTDI UART.
LED9	RX	yellow	Toggle: data flow to FTDI UART.
LED3	HWINT	yellow	Toggle: data flow from Arduino to Raspberry Pi via "ARD2RPI_HV".
LED4	BATLOW	red	On: battery voltage below critical level.
LED6	5V	green	On: 5V buck converter provides output.
LED8	6V	green	On: 6V voltage regulator provides output.
LED10	BTS	blue	On: Bluetooth module is connected.
LED11	BTS1[1]	blue	Flash 1Hz: Command mode. Flash 2Hz: Data mode.

Locations of LEDs on the PCB can be found in Section 4.21.

[1] Parallel to onboard LED on HC-05 breakout board.

4.15 Component updates: IMU and pressure sensor

To ensure the availability of components for this design upgrade and for the production of future copies of µ-CAT, the IMU and the pressure sensor are updated with newer versions. The new devices perform the same functions as the old ones but may require new software drivers and a different electrical connection.

The original IMU is upgraded with a newer version from the same manufacturer – TDK InvenSense ICM-20608-G. The device's features are similar to those of the original IC. The reason for this update is to maintain availability of components used on µ-CAT, to guarantee future producibility.

The IMU includes a triple-axis MEMS gyroscope with digital output for angular velocities about all three axes, an accelerometer with digital output of linear accelerations along all three axes and a thermometer [56, Ch. 2]. The device can be interfaced using I^2C or SPI [56, Ch. 2.3]. On µ-CAT's design, I^2C is used, and the device is a slave on that bus. Up to two of such chips can be placed on the same I^2C bus, and their addresses are selected by setting the state of the address pin "AD0" [56, Ch. 6.2]. Whenever the sensor performs a new measurement, it stores it in its internal FIFO (First in, first out) buffer and sets a flag which can be read using the interrupt status pin signal on pin "INT".

The new pressure sensor MS5837-02BA has a very compact footprint (3.3 x 3.3 x 2.75mm – approximately one quarter of the area of the old pressure sensor) and is therefore very easy to physically integrate into the new design. The sensor covers pressures ranging from 300mbar to 1200mbar – which allows sensing in diving depths up to approximately 2m in water. It withstands pressures of up to 10bar (91m depth). The given resolution of 13cm at sea level [57, p. 1] means that it can detect changes as small as 0.0153mbar, which corresponds to a resolution of more than 65000 discretized values per 1bar of pressure – a resolution of around 0.15mm in water. The accuracy describes the maximum bias (offset) from the correct value to be expected in a single measurement under specific environmental and electrical conditions, and under the conditions in which µ-CAT is operated, it is stated to be within a corridor of $\pm 2\text{mbar}$ [57, p. 3] which corresponds to roughly 2cm in water. The device is interfaced using the I^2C protocol.

There are several advantages of the new over the old sensor, apart from the reduced footprint size. The first one to mention is the lower production cost and effort. The old sensor requires an operational amplifier and an ADC (analogue-to-digital) converter chip in order to have enough accuracy when reading the output voltage using an analogue input pin of the Arduino. The new one does not require these components, since it comes with a microcontroller integrated, providing digital data. Secondly, with the elimination of these additional components, the power consumption is expected to be reduced.

4.16 Integration of motor drivers (I^2C)

Four ICs of type DRV8830 – one per motor – are used to drive the servo motors through an internal H-bridge. These chips can be interfaced via I^2C bus. Each chip provides two address pins which can be in one of three states: high, low or open. This way, up to nine addresses can be set for this type of device on the I^2C bus. When implementing the motor drivers into the new design, care is taken that the original addresses of the devices are kept, so that no software-sided modification is required. The devices can sense the current consumption of the load if external shunt resistors are placed. This feature is currently not used on µ-CAT, but space for the respective shunt resistors is provided on the circuit board. With the feature omitted these resistors should have a value of 0Ω, otherwise their values are calculated based on the voltage drop of 200mV (according to the chip's reference voltage) to be created when the current threshold value is reached [58, Ch. 7.3.3]:

$$R_{\text{ISENSE}}\ [\Omega] = \frac{0.2\ [\text{V}]}{I_{\text{LIMIT}}\ [\text{A}]} \tag{24}$$

With a measured current consumption of approximately 138mA per motor during regular operation and an expected current consumption of around 500mA per motor in stall condition, a current threshold of 400mA for each motor would be a safe choice to detect stalling. Using above equation, this would yield a value of 0.5Ω for each of the resistors R13, R14, R15 and R16 (see Figure 22). Since the dedicated "FAULT" pin of the chips is not routed on the PCB, the flag bits D0 and D4 set in the I^2C register 1 [58, Ch. 7.6.1.2] would need to be used in order to detect the fault condition by software.

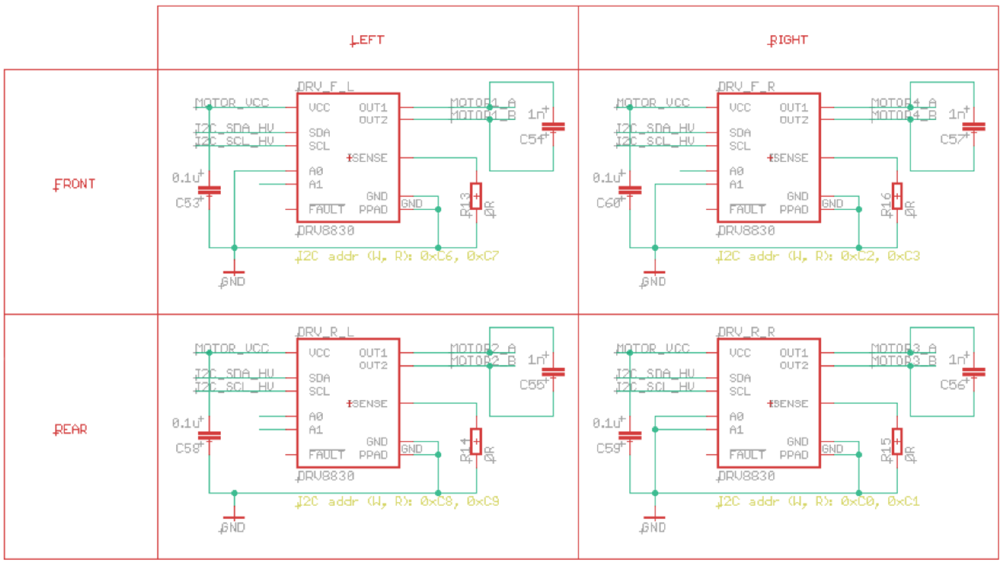


Figure 22. Motors drivers' schematic diagrams.

The chips' surrounding circuitry (Figure 22) is developed in accordance with the prescriptions in the datasheet [58, Sec. 5], and they are integrated into the new layout of the PCB as per recommendation [58, Sec. 10.2]. The latter requires a specific footprint (Figure 23) which guarantees sufficient size of the thermal pad and which allows this pad to extend seamlessly into the surrounding ground fill of the PCB. With a proper heat-sinking pad provided, the chip can provide up to 1A peak output current without overheating [58, Ch. 3].

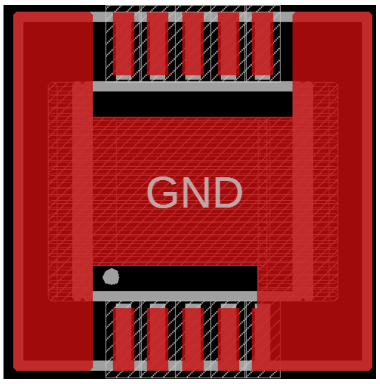


Figure 23. Custom-made Eagle footprint for DRV8830 (Eagle rendering).

The described implementations refer to PCB design version 26.

4.17 Integration of motors

The servo motors from the original μ-CAT are used in the upgraded version. Improved placement of motor connectors to the PCB allows for shorter wires from the motors. This is expected to improve EMC (electromagnetic compatibility) characteristics and provides a cleaner overview when working on the robot.

The outputs of the motor driver chips are routed to four pin header connectors with two pins each. These are SMD connectors, positioned near the corners of the PCB, to keep wire runs to the motors short.

To obtain feedback from the motors, they have been modified by the engineers of Centre for Biorobotics and have been used with this modification already on the original version of µ-CAT. The team expressed their wish to keep this manual alteration of the motors in the upgraded version. The original motors (off the shelf) are equipped with an internal microcontroller, which allows for sending a position command to the motor, and an internal potentiometer as a feedback sensor to get the motor to the setpoint. This motor, when used as-is, does not provide any feedback information to the user, so no external control algorithms can be applied.

In the course of the modification of the motor, the rotary Hall effect sensor AMS AS5040 has been added as a replacement for the original potentiometer. The internal microcontroller has been removed, and interfacing signal lines to the Hall effect sensor have been provided, so that the feedback information is available to a user – who can now apply custom-made control algorithms.

The Hall encoder interface lines consist of five signals, including 5V and ground. They use a three-wire SPI interface which contains a chip select (“CS”) line. To connect the motor encoders, SMD connectors are chosen and placed on the top layer of the PCB, near the corners, to keep the wire runs to the motors short.

The described implementations refer to PCB design version 26.

4.18 Integration of battery voltage measurement

In order to measure the battery voltage with the Arduino microcontroller, a simple voltage divider circuit is implemented, taken over from the original µ-CAT design. The voltage divider is clamped between the battery voltage line (“VIN”) and the ground. The output line from the voltage divider is called “BATTERY_MEASURE” and routed to analogue input pin ADC6.

The output voltage of the voltmeter can be calculated as follows:

$$U_{\mathrm{out}} = U_{\mathrm{B}} * \frac{R_{\mathrm{L}}}{R_{\mathrm{L}}+R_{\mathrm{H}}} \tag{25}$$

where:	U_{out}	[V]	output from voltage divider
	U_{B}	[V]	supply voltage from battery
	R_{L}	[Ω]	resistor between supply and output (R28)
	R_{H}	[Ω]	resistor between ground and output (R29)

With the resistor values used on µ-CAT ($R_{\mathrm{H}} = 47\mathrm{k\Omega}, R_{\mathrm{L}} = 5.6\mathrm{k\Omega}$), the output voltage of the voltage divider is always around 10.6% of the supply voltage, therefore ranging up to approximately $0.72\mathrm{V}$ for the nominal value of the batteries. With the resistor values used, the battery voltage value can be calculated from the voltage divider output voltage using the following equation:

$$U_{\mathrm{B}} = \frac{U_{\mathrm{out}}}{0.106} \tag{26}$$

The sensing range of the Arduino microcontroller is set to match with this range by using the internal analogue voltage reference of the ATmega328P, which is $1.1\mathrm{V}$. With approximately $0.79\mathrm{V}$ being used as the maximum value (full batteries), this corresponds to almost 72% of the full range, providing a precision of 734 distinct values to measure battery voltage. In case the robot is powered through an external power source with slightly higher voltage, for the purpose of testing, the design leaves headroom for measuring this voltage without destroying the microcontroller. The maximum measurable input voltage is $10.37\mathrm{V}$.

The correspondence between the analogue input value x with range $(0; 1023)$ and the output voltage from the voltage divider is expressed as follows:

$$U_{\mathrm{out}} = 1.1\mathrm{V} * \frac{x}{1023} \tag{27}$$

From the analogue input value x, the battery voltage can be calculated using following equation:

$$U_{\mathrm{B}} = 1.1\mathrm{V} * \frac{x}{1023 * 0.106} = 0.010\mathrm{V} * x \tag{28}$$

4.19 Implementation of level shifters

Commonly used voltage levels on devices are 3.3V and 5V. In the Eagle design, most signal lines with voltage levels of 3.3V carry the suffix "_LV" (low voltage), the ones with 5V levels the suffix "_HV" (high voltage). Whenever two devices with different logic voltage levels must communicate with each other, a level shifter can be used to take care about the translation between these voltages. This helps protect the device with the lower voltage level from overvoltage. If the voltage level on one side of the level shifter is 0V (low), the other side automatically is also 0V (low). If the voltage on one side of the level shifter is high, the other side must also become high, but the voltage must correspond to the desired voltage level of this side. The voltages to be used by the level shifter are usually defined by applying the levels to reference pins, one on each side of the level shifter.

Instead of using a dedicated voltage level shifter IC, a network of two resistors and a Schottky diode can be used, as suggested in the Bluetooth module datasheet [59, Fig. 3]. Since the new PCB design requires level shifting between more than two devices, using a chip is more space-efficient.

In this new circuit design, two different level shifter ICs are used, as explained in the following subsections.

TXB0104 is a digital CMOS bidirectional 4-channel voltage-level translator with automatic detection of the direction of a signal. This allows either of the two sides to be the driving side, while the other side follows accordingly. Whether a side is driving or driven is independent from its state (high or low).

Since I^2C uses strong pullup resistors (1kΩ), TXB0104 is not applicable for translating voltages on the I^2C bus because of the low drive strength of the chip's outputs [60, Ch. 8.3.5]. Therefore, a different chip is used to translate the voltages on the I^2C bus: PCA9306. This IC has been especially developed for the purpose of handling translations on the I^2C bus. Its operating principle is based on an N-channel MOSFET switch used on each signal line routing. Additionally, a diode is placed from source to drain. The gate of the transistor is always high (connected to the low-side positive voltage), therefore the transistor conducts whenever its source voltage potential falls below that of the gate. The source of the transistor is connected to the low-side bus signal, and its drain to the high-side bus signal.

- In the default state, both sides are high (due to the external pullup resistors).
- When the low-voltage side drives the bus low, the transistor becomes conductive (source potential below gate potential), and due to its very small internal resistance, the potential on the drain (high-voltage side signal) becomes low.
- When the high-voltage side drives the bus low, the internal diode becomes forward-biased for one moment, and its knee voltage demands the voltage on the low-voltage side signal to become only slightly higher (around 0.7V) than that of the high-voltage side. In this condition, the transistor becomes conductive, and the signal on the low-voltage side becomes equal to that on the high-voltage side (0V). The diode is now bypassed.

The level shifter can be disabled by opening jumper "I2CLVLEN" (indicated as "I2C_LVL_EN" on the PCB). This allows the use of the low-voltage I^{2}C bus without connection to the high-voltage side (Raspberry Pi and all I^{2}C devices except for Arduino and motor drivers).

4.20 Provision of test pads

The following tables list all test pads provided on the PCB, with their name indicated (printed) on the PCB in the left column. The last column relates to the signal connected to the respective test pad.

For six signals related to power supply signals the following test pads (Table 10) are provided (more than one pad with the same name might be present on the PCB):

Table 10. Provided power supply test pads.

Pad name	**Purpose**	**Eagle signal line name**
3V3	3.3V line from regulator	3V3
5V	5V line from regulator	5V
6V	6V line from regulator	MOTOR_VCC
GND	Ground (0V) reference	GND
USB5	5V line from USB port	USB_VCC
VIN	Battery voltage	VIN

For 11 miscellaneous signals, the following test pads (Table 11) are provided:

Table 11. Provided test pads for miscellaneous signals.

Pad name	**Purpose**	**Eagle signal line name**
PDL	Photodiode left	PHOTODIODE_L
PDR	Photodiode right	PHOTODIODE_R
RCL	Tone detector received signal left	TO_RCVR_L
RCR	Tone detector received signal right	TO_RCVR_L
TDA	Tone detectors combined amplified output	TD_OUT_AMP
TDL	Tone detector left output to FSK	TD_OUT_L
TDLA	Tone detector left amplified output to logic OR	TD_OUT_L_AMP
TDR	Tone detector right output to FSK	TD_OUT_R
TDRA	Tone detector left amplified output to logic OR	TD_OUT_R_AMP
V2L	Tone detector left centre voltage	VCC/2_L
V2R	Tone detector right centre voltage	VCC/2_R

Seven signals routed to pins of the Raspberry Pi computer have the following test pads (Table 12) on the PCB:

Table 12. Provided test pads for signals connecting to Raspberry Pi pins.

Pad name	**Purpose**	**Pin no.**	**Type**	**Eagle signal line name**
APL	Messages Arduino to Raspberry Pi (3.3V)	7	Input	ARD2RPI_LV
BTE	Bluetooth EN pin (enable)	13	Output	BT_CMD
BTR	Bluetooth RST pin (reset)	38	Output	BT_RST
BTS	Bluetooth STATE pin (connection status)	40	Input	BT_STATE
R3V3	Raspberry Pi-generated 3.3V output	1	Power	RPI_3V3
X0	UART TX (3.3V)	8	Output	RPI_TX
Y0	UART RX (3.3V)	10	Input	RPI_RX

23 signals routed to pins of the Arduino microcontroller have the following (smaller sized) test pads (Table 13) on the PCB:

Table 13. Provided test pads for signals connecting to Arduino pins.

Pad name	**Purpose**	**Pin name**	**Type**	**Eagle signal line name**
A7	Analog input pin (unused)	ADC7	A. Input	ADC7
APH	Arduino to Raspberry Pi (5V)	PC2	Output	ARD2RPI_HV
BLO	Battery low signal	PC0	Output	BATTERY_LOW_LED
BM	Battery voltmeter signal	ADC6	A. Input	BATTERY_MEASURE
CS1	Encoder Front-R. SPI select	PD7	Output	CS1
CS2	Encoder Rear-R. SPI select	PB2	Output	CS2
CS3	Encoder Rear-L. SPI select	PB1	Output	CS3
CS4	Encoder Front-L. SPI select	PB0	Output	CS4

Table 14. Provided test pads for signals connecting to Arduino pins [continued].

Pad name	**Purpose**	**Pin name**	**Type**	**Eagle signal line name**
FEN	FSK module enable	PC1	Output	FSK_ENABLE
FRX	FSK module receive	PD6	Input	FSK_RX
FTX	FSK module send	PD3	Output	FSK_TX
IMU	IMU interrupt signal 5V	PD2	Input	IMUINT_HV
LB	Left beacon signal	PD5	Input	LEFTBEACON
MEN	Motor power enable pin	PC3	Output	MOTOR_EN
MISO	SPI MISO bus 5V	PB4	Input	SPI_MISO
MOSI	SPI MOSI bus 5V	PB3	Output	SPI_MOSI
RB	Right beacon signal	PD4	Input	RIGHTBEACON
RST	Arduino reset pin	PC6	Input	ARDU_RST
RX	UART RX (5V)	PD0	Input	ARDU_RX
SCK	SPI clock (5V)	PB5[1]	Output	SPI_SCK
SCL	I²C clock (5V)	PC5	Output	I2C_SCL_HV
SDA	I²C data (5V)	PC4	Bus	I2C_SDA_HV
TX	UART TX (5V)	PD1	Output	ARDU_TX

In addition to the test pads, two ground fill areas (congruent; one in top, the other in bottom layer) are exposed along part of one edge of the circuit board. These are practical when measuring signals using an oscilloscope: the ground connector (usually a crocodile clamp) can be attached to them.

The physical locations of all test pads can be found in Section 4.21, Figure 24.

4.21 Considerations on modularity and reliability

With the aim to improve reliability in terms of vibration-tolerance and to avoid bad contacting due to corrosion, the overall design follows the principle to avoid pin headers or wire connections wherever possible.

On the previous version of µ-CAT, the following devices were installed as add-on components:

- motor drivers;
- FTDI chip;

[1] Through resistor R1.

- tone detectors;
- Arduino Mini 05 board;
- power supply PCB.

On the new version of the PCB, only the following devices are connected via pin headers:

- Raspberry Pi single-board computer;
- FSK module.

The decision to integrate devices into the layout rather than making them detachable involves a trade-off in terms of modularity. In case of water damage or failure of a components, the corresponding device cannot be easily replaced. However, since the production of a single PCB is cheap and does not necessarily involve manual methods anymore, this trade-off is acceptable, and the aim for reliability is of higher importance. Apart from that, the provision of test pads facilitates debugging and detection of the failed component which can then be manually replaced in a few relatively simple steps.

Another advantage of avoiding excessive modularity is the higher space-efficiency: devices can be placed more closely to each other, and with the avoidance of through-hole pin headers, both sides of the PCB can be used to a greater extent.

4.22 Using the main PCB

This section gives an overview on all features of the main PCB with which the user is most likely to interact during operation, debugging and assembly of the robot.

Test pads and most LEDs are located on the top layer of the board, to allow for easy debugging and monitoring. Figure 24 shows their locations on the PCB.

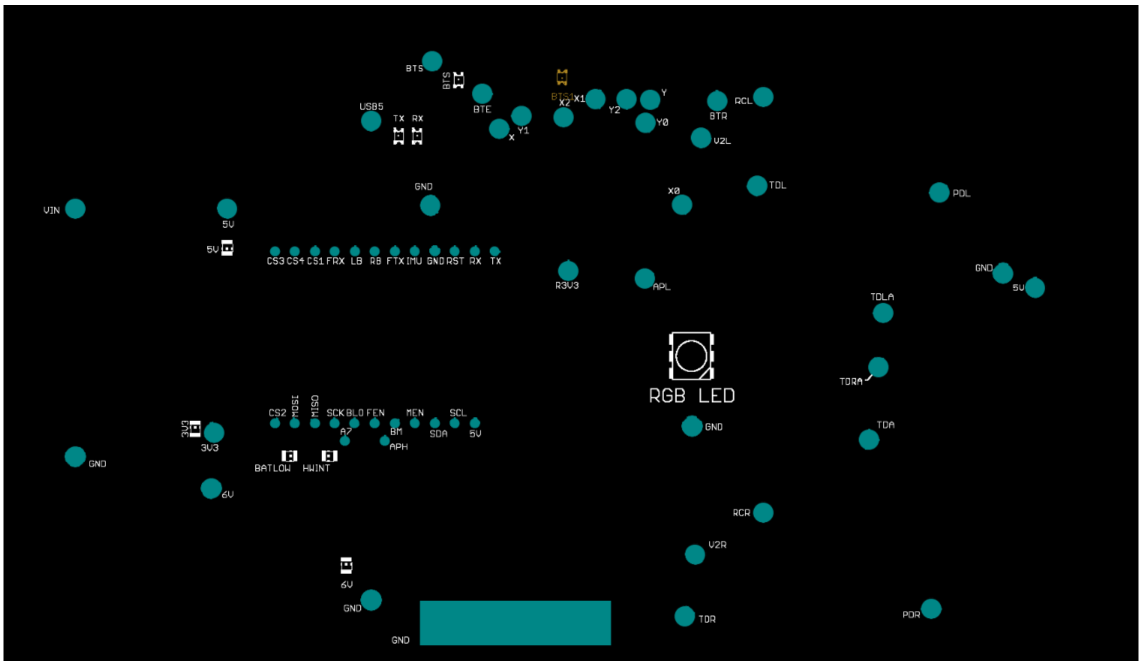


Figure 24. Test pads and LEDs (Eagle rendering). Test pads (green), LEDs on the top layer (white) and LEDs on the bottom layer (orange).

Connectors are and jumpers are located on both copper sides of the PCB. Their locations are shown in Figure 25 and Figure 26.

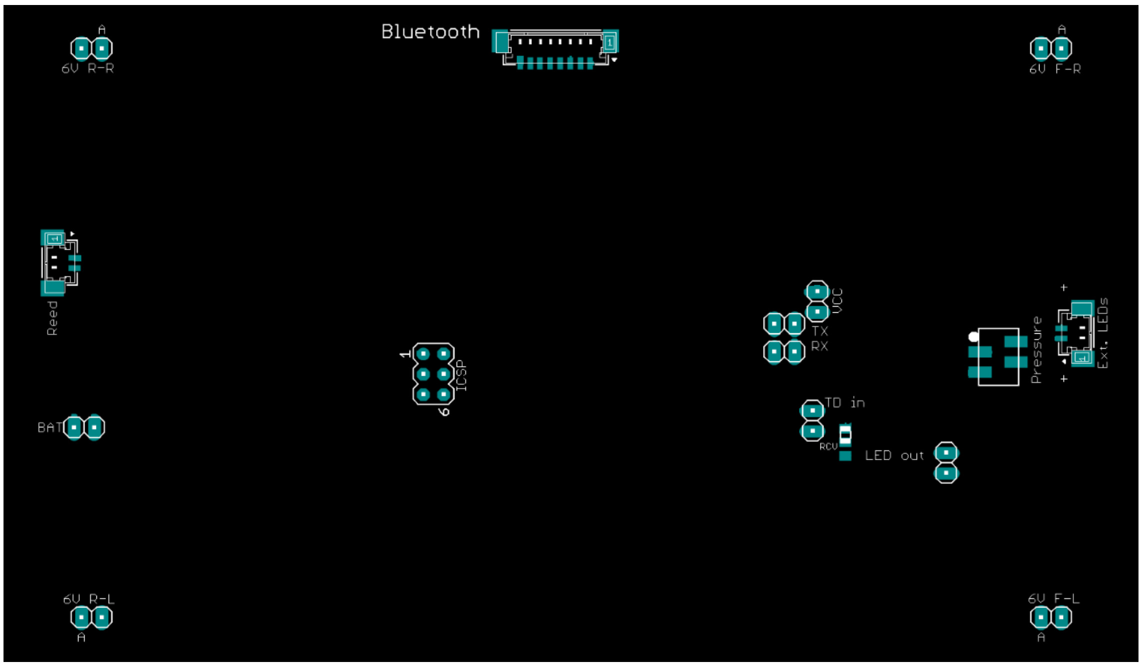


Figure 25. Connectors and jumpers on the top layer (Eagle rendering).

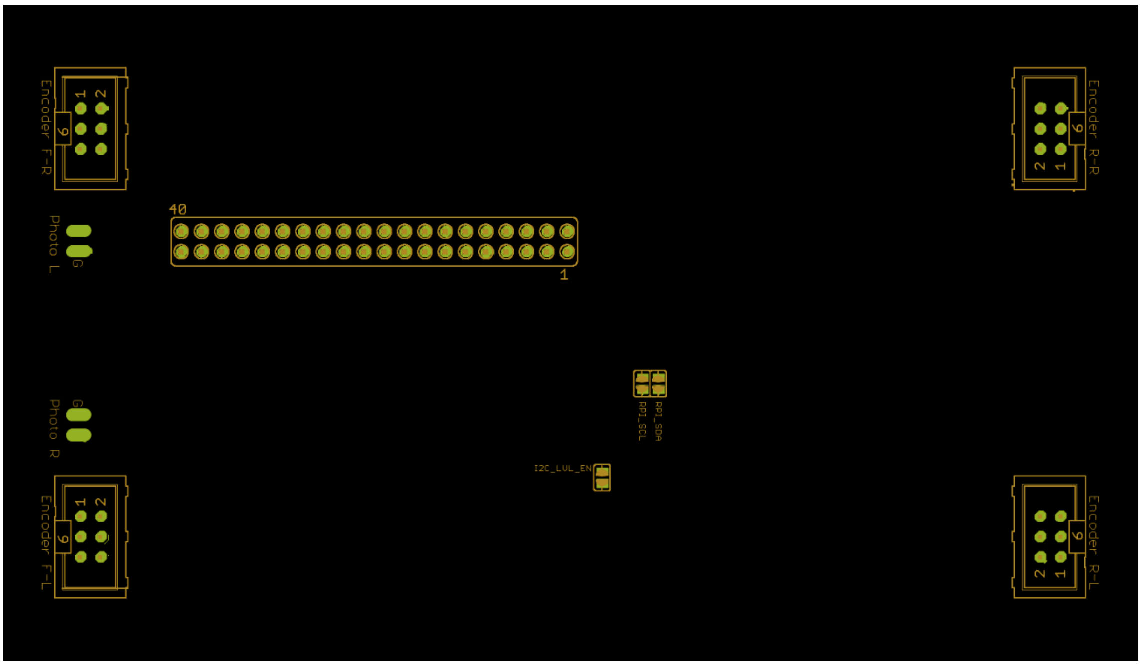


Figure 26. Connectors and jumpers on the bottom layer (Eagle rendering).

4.23 Results and discussion

All working features of the original µ-CAT robot are implemented in the new design and tested. The following subsections do not emphasize on these features, but list only significant improvements and changes to the robot. Figure 27 shows the result of the Eagle design work (referring to version 26 of the PCB), and Figure 28 shows top and bottom view of the real PCB (version 21), with some components not yet in place.

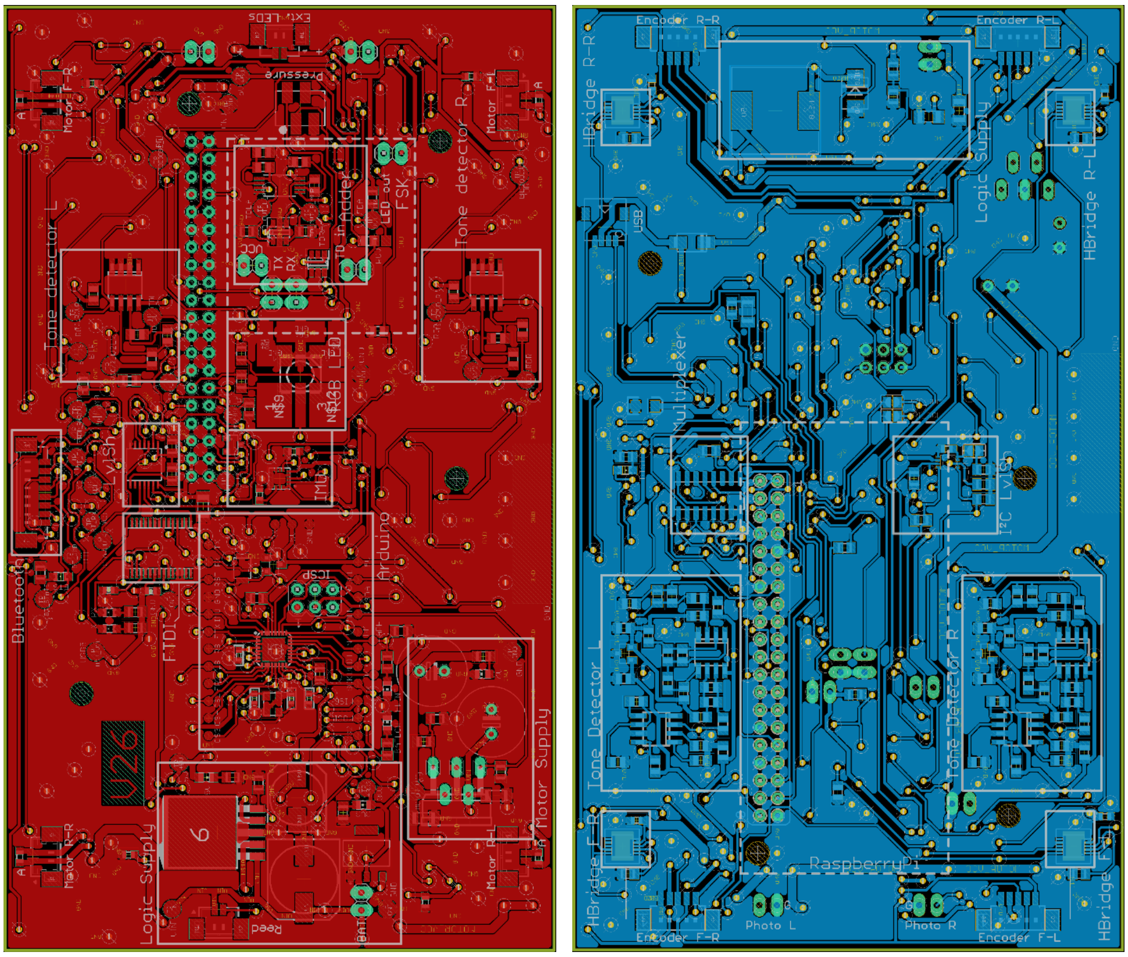


Figure 27. Eagle renderings of new PCB. Top view (left) and bottom view (right).

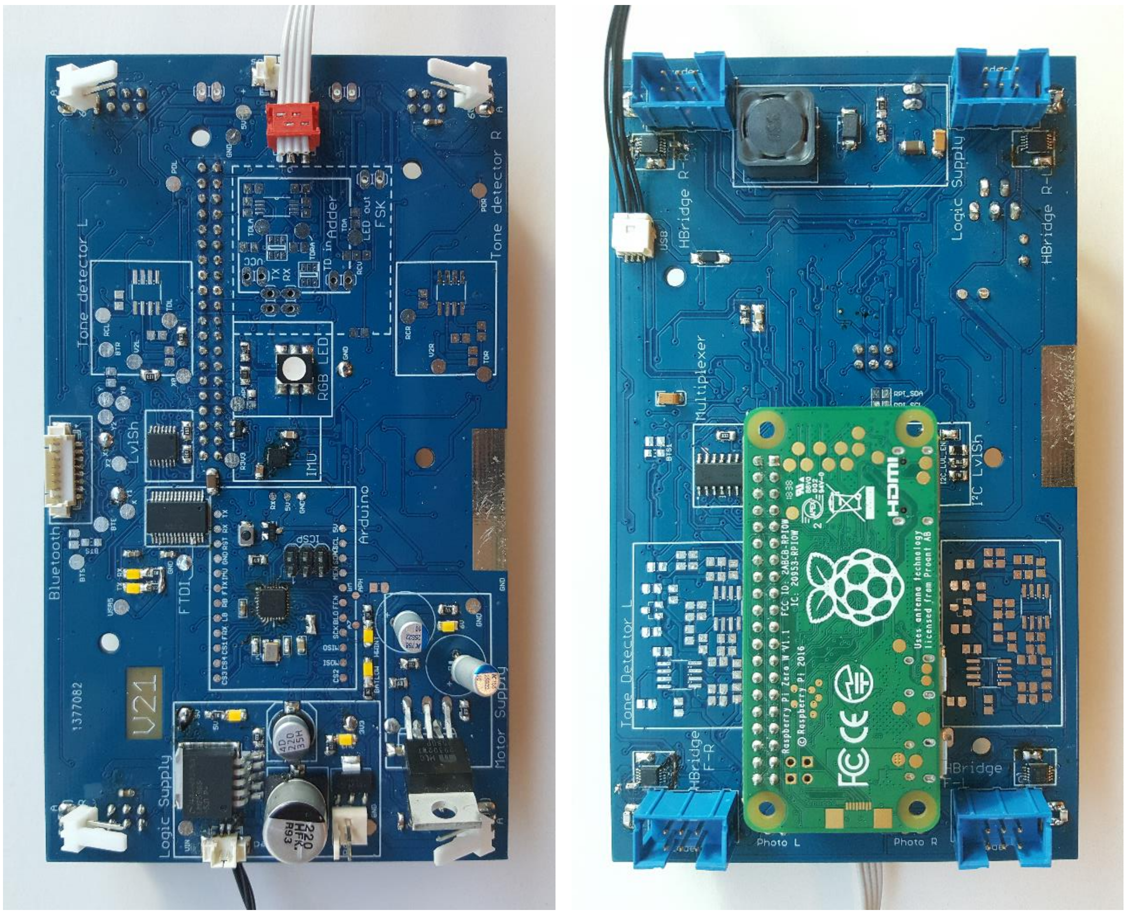


Figure 28. New PCB. Top view (left) and bottom view (right).

4.23.1 Power distribution, consumption and supply

Apart from the 6V power supply for the motors, µ-CAT is now equipped with a 5V buck converter circuit and a 3.3V linear voltage regulator supply on the new PCB.

Figure 29 shows the topology of power distribution lines of the updated robot, along with their names by which they are identified in the schematic diagram in Eagle.

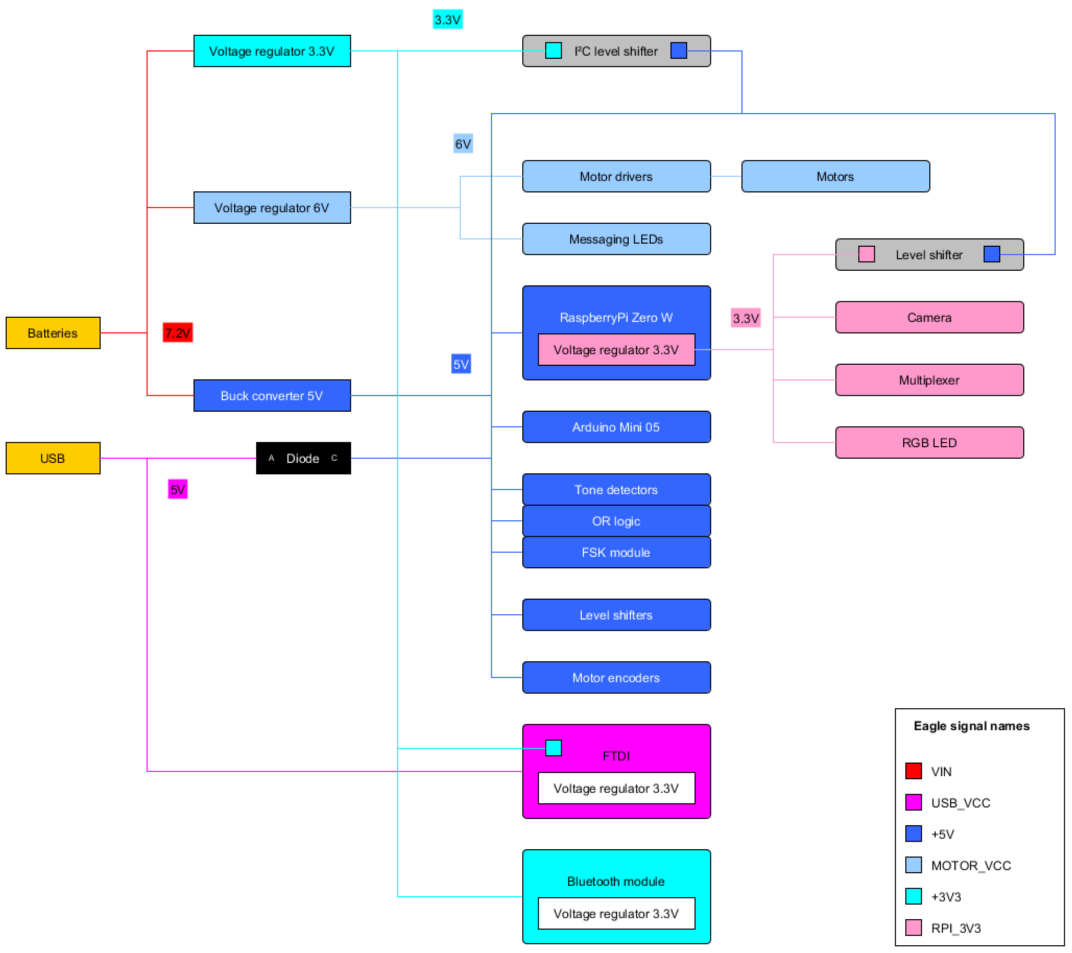


Figure 29. Block diagram of power distribution on the updated µ-CAT. The diode ensures that USB can power the 5V rail only if its voltage is higher than the supplied voltage from the buck converter. The minimum difference is equal to the knee voltage of the diode (~0.5V).

The power consumptions of old and new µ-CAT have been experimentally compared, as described in Appendix A 1.3. The following observations can be made:

- Under full load (including all new components added and running), the new robot consumes 5.669W. This yields a battery lifetime of approximately 2.9h, using the batteries from original µ-CAT, as stated in Section 2.1.2,
    - The old robot has a consumption of 3.434W with all devices powered. The consumption of the new robot has increased to 165%.
- Analysing the performance of the new PCB with the old one under comparable conditions (same devices connected):
    - Quiescent consumption of the new PCB is 79% of the old PCB without Raspberry Pi.

    - Quiescent consumption of the new PCB is 57% of the old PCB with Raspberry Pi. This shows the obtained power saving due to the buck converter on the 5V rail.
  - Comparing old robot under full load including a Raspberry Pi with the new robot under full load (additional hardware: messaging LEDs and Bluetooth module), the new robot consumes 3% less power.

Tests also show that the circuitry can be fully powered using only the USB supply.

Changes have also been made to the reed switch connection. Its connection principle is opposed to that used in the original µ-CAT design, and it reduces the probability of the magnet getting lost during operation of the robot. With the original µ-CAT design, if the robot loses the magnet while being in the water, it would switch off and remain passively floating – which would pose the risk of getting lost or destroyed. In the new design, the robot is operated without the magnet in place.

4.23.2 Integration of Arduino, Raspberry Pi Zero W and camera module

The original Arduino board layout is copied with modifications to the new PCB. The onboard voltage regulator is removed, and a slightly different CPU package version is used, which features a thermal pad underneath. Redundant pin access points are removed for higher space-efficiency. Instead of the through-hole pins used on the original Arduino board, SMD (surface mount device) test pads are used. An ICSP (in-circuit serial programming) pin header is added to provide a port for low-level programming of the microcontroller, for example, if a new bootloader is needed.

In cases where UART communication between Raspberry Pi and Arduino is not established, the Arduino can use a dedicated signal line to request certain tasks from the Raspberry Pi.

The following list summarizes the implemented features relating interfacing between Raspberry Pi and other components and subsystems on µ-CAT:

- Raspberry Pi interfaces the camera module;
- Raspberry Pi can reset the Arduino via a dedicated GPIO output pin (see Section 4.13);

- Raspberry Pi can communicate via the UART bus with Arduino (see Section 4.11);
- Raspberry Pi can be used to flash new software to the Arduino microcontroller (as a result of the previous two features combined);
- Raspberry Pi can communicate via the I²C bus with other I²C devices and with Arduino (if jumpers “RPISDA” and “RPISCL” are set, which are indicated as “RPI_SDA” and “RPI_SCL” on the PCB, respectively);
- Raspberry Pi can communicate with Arduino via Bluetooth;
- Raspberry Pi controls the multiplexer chip (see Section 4.11.1);
- Raspberry Pi drives the onboard RGB status LED (see Section 4.14.1);
- Raspberry Pi controls the standalone Bluetooth module (see Section 4.10.2);
- Arduino can send simple messages on a one-wire unidirectional signal line to the Raspberry Pi (see Section 4.12).

The addition of jumpers on the I²C line allows for the Raspberry Pi instead of the Arduino to be used for interfacing all sensors on that bus if the I²C level shifter is disabled by opening jumper “I2CLVLEN” (indicated as “I2C_LVL_EN” on the PCB).

With the implementation of the camera module, µ-CAT will now be able to perform visual servoing and object detection and avoidance, using software algorithms running on the Raspberry Pi. This greatly improves the level of autonomy of the robot.

4.23.3 Implementation of advanced optical messaging

The optical messaging system most typically covers three tasks:

- steer the robot left or right by using external light beacons;
- demodulate messages modulated onto the light signals;
- modulate messages onto a base signal and send them out via messaging LEDs installed on the front end cap of the robot.

Being equipped with powerful signal LEDs in the front end cap, the robots can now communicate back, either to the user or to other robots when working in a swarm.

The development of the messaging system is currently still ongoing (not part of this thesis scope), but the PCB provides all interface pins to attach the FSK module. Pin headers are provided on the PCB which can hold any frequency-shift keying PCB which has a compatible circuitry and is designed considering the given pin layout. This form of "partial integration" is less reliable than copying the layout to the PCB, but it offers more flexibility. This is necessary because the module is expected to undergo changes before it can finally implemented.

The tone detectors already exist on the original µ-CAT in the form of separate circuit boards. For the new PCB design, to avoid potential sources of bad contacting, their layout is integrated on the main PCB.

The same photodiodes used on the original µ-CAT design are implemented, but in a physically more advantageous configuration than before, now allowing a for a better distinction between optical signals from the right and from the left.

4.23.4 Communications

With integration of the new Bluetooth module, untethered messaging with the Arduino microcontroller becomes a possibility. The integration of Raspberry Pi Zero W allows additional for wireless communication methods.

With the integration of the multiplexer, one of three devices can communicate with Arduino using UART. The Raspberry Pi W handles the switching procedure by interfacing the multiplexer.

The integration of two level shifters allows for devices with two different voltage levels to communicate with each other on the same busses.

4.23.5 Improved user-interface

The RGB LED is implemented and ready to be used with PWM-capable pins on the Raspberry Pi. It offers possibility for µ-CAT to keep the user updated on system status. Colour-coded messages can be used to distinguish between a variety of messages.

With test pads on 47 signals, almost each component of the new PCB can be tested using a voltmeter or oscilloscope. This improves µ-CAT's usefulness as a teaching tool and allows for more efficient debugging procedures in case of malfunctions.

4.23.6 Future work

In the latest design version of the PCB (version 26), some connectors are replaced by SMD components, in order to save space and facilitate connecting of plugs. Since many of them are of the same type – Molex PicoBlade – a user might accidentally plug a component into the wrong connector during assembly. As part of future improvements, some additional time must therefore be invested for finding dissimilar connector socket types for different types of devices.

In the next design iterations, the fuse protecting the batteries from overcurrent must be integrated onto the PCB. A self-resettable polyfused can be used for that purpose. In the original design, a glass-body fuse has been placed inside the battery compartment. Since µ-CAT uses a new type of batteries now which are longer than the original ones, this fuse does not fit inside the compartment anymore. For now, it must be placed in line with the wire leading from the batteries to the PCB.

The next version of the PCB will have the input of the 3.3V regulator routed to the output of the 5V regulator, not to the battery voltage. This will prevent the 3.3V components from remaining powered when the reed switch is closed.

Other improvements concern ease of assembly: To facilitate self-centring of the PCB in the grooves of the front end cap, the PCB needs to slide into the Plexiglass weight dividers freely. To achieve this, electronic components close to the edge of the PCB must be moved. Apart from that, it is planned to mount one circuit board on the inside of each end cap. Each of them has connectors to the electronic components on one side and a slide-in connector on the other, where the PCB connects to when pushing the end caps against it. this way, mechanical and electrical contact are established at the same time.

5 Software

This chapter presents the results of the thesis project regarding software development. The first part presents methods to flash new code to the Arduino microcontroller, and the second part describes the test software created for testing the components on the new motherboard.

5.1 Programming the Arduino

While the Raspberry Pi Zero is equipped with a Linux operating system on a micro-SD card and can easily be programmed by connecting to it via SSH (secure shell), WLAN or by placing the micro-SD card into another computer, the methods of programming of the low-level microcontroller need to be considered in the electronic design of the Embedded System. The new PCB offers several methods to program the Arduino.

5.1.1 ICSP

The microcontroller can be flashed via its ICSP pin header. This method is not recommended for regular use, because the code will overwrite the bootloader on the AVR. Without the bootloader, it is not possible to program the microcontroller through UART. In order to enable programming via UART, a bootloader must be flashed to the microcontroller. This can only be done via ICSP. The Arduino IDE (integrated development environment) offers a possibility to do that.

For flashing code or the bootloader to the AVR, the ISP programmer must be connected to the ICSP pin header and the other side to a USB port of the PC. For flashing code, the function “Upload Using Programmer” can be selected in Arduino IDE from the “Sketch” menu. For flashing the bootloader, the option “Burn Bootloader” from the “Tools” menu can be used.

5.1.2 UART

Although UART is designed as a one-to-one communication protocol, the new PCB offers the possibility for one of several devices to communicate with Arduino via UART using the multiplexer. To flash code to the microcontroller via UART, the multiplexer is instructed to route the respective UART device port to the Arduino. This instruction can only be issued by the Raspberry Pi, and Python scripts are provided as part of this thesis work which accomplish the switching procedure.

UART via USB

The Arduino Mini 05 does not offer a USB port out of the box. An FTDI chip is installed on the PCB, allowing the Arduino to be programmed via USB, using the Arduino IDE. When doing so, the microcontroller is automatically reset shortly before the flash procedure starts. This is done by pulsing the “DTR” pin on the FTDI chip.

UART via Raspberry Pi

The most convenient method of programming the Arduino is via the Raspberry Pi, because the Raspberry Pi can reset the Arduino automatically in the right moment. However, this automatic reset cannot be executed when flashing the Arduino through the regular Arduino IDE. Instead, Python scripts running on the Raspberry Pi are provided as part of this thesis work which fully automate the compilation and upload process of any given Arduino sketch.

5.2 Test case automation

For the components on the new PCB, test software is developed as part of this thesis project. The purpose of this software is to verify proper function of a newly produced PCB and to troubleshoot during regular use. The software suite consists of several test case modules and an overlaying user interface. The test case files are stored on the SD card of the Raspberry Pi and executed from there. The software is organized in a modular way, with distributed files in different folders, where the file structure reflects the software architecture: each test case is in a separate folder which contains all required files for that test case. Shared files which are used by several test cases are not duplicated but placed in a folder at the root of the test case folder structure.

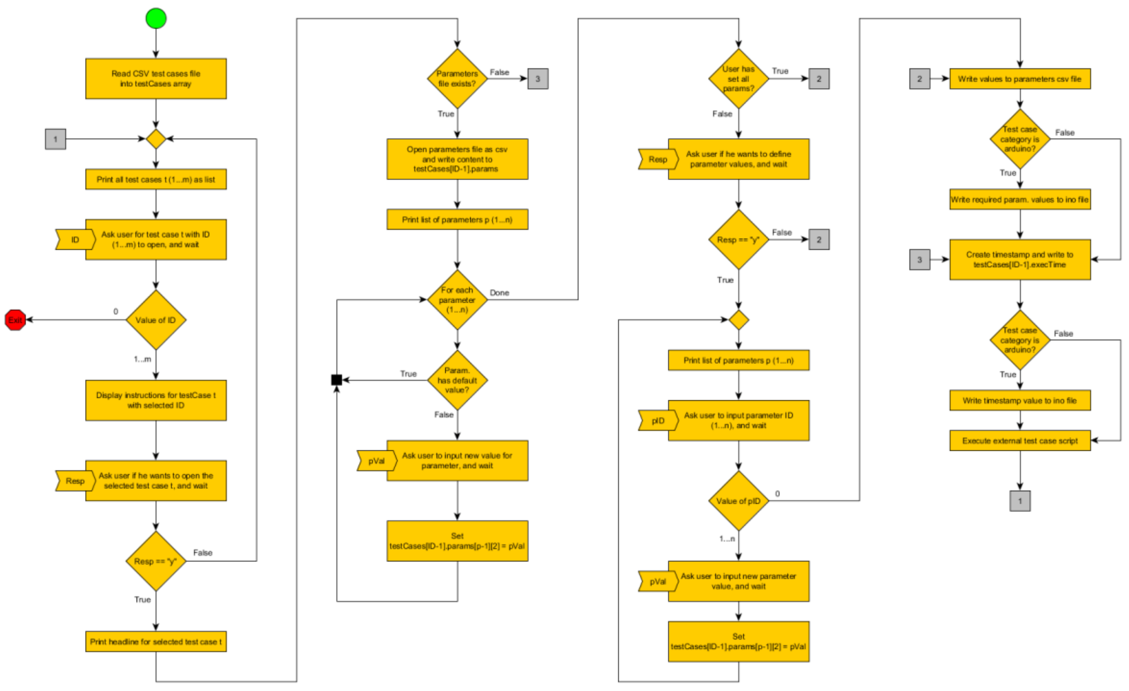
The following sections provide proof-of-concept of the test software suite, including a number of test cases. References to most relevant files can be found in Appendix A 4.4.

5.2.1 User interface

The overlaying user interface – a Python 3 script – facilitates the execution of test cases in the following ways:

- The end user does not need to manually locate the test case files. The user interface finds them as specified in a text file, and the user can start a test case through the user interface.
- The user interface displays a general description for each test case, stating its purpose and working principle, and provides instructions individual to each test case.
- The user interface takes care that parameters required for running a test case are defined and stores them for future executions of the test case.

The working principle of the user interface script is graphically presented in Figure 30, and the corresponding Python code can be found in Appendix A 4.4.2.



Read CSV test cases file into testCases array
1
Print all test cases t (1...m) as list
ID
Ask user for test case t with ID (1...m) to open, and wait
Exit
0
Value of ID
1...m
Display instructions for testCase t with selected ID
Resp
Ask user if he wants to open the selected test case t, and wait
Resp == "y"
False
True
Print headline for selected test case t
Parameters file exists?
False
3
True
Open parameters file as csv and write content to testCases[ID-1].params
Print list of parameters p (1...n)
For each parameter (1...n)
Done
True
Param. has default value?
False
pVal
Ask user to input new value for parameter, and wait
Set testCases[ID-1].params[p-1][2] = pVal
User has set all params?
True
2
False
Resp
Ask user if he wants to define parameter values, and wait
Resp == "y"
False
2
True
Print list of parameters p (1...n)
pID
Ask user to input parameter ID (1...n), and wait
Value of pID
0
1...n
pVal
Ask user to input new parameter value, and wait
Set testCases[ID-1].params[p-1][2] = pVal
2
Write values to parameters csv file
Test case category is arduino?
False
True
Write required param. values to ino file
3
Create timestamp and write to testCases[ID-1].execTime
Test case category is arduino?
False
True
Write timestamp value to ino file
Execute external test case script
1

Figure 30. Flowchart of the overlaying user interface Python script.

The list of test cases presented to the user contains only those test cases which are defined in the file “testcases.csv”.

When the user interface calls a test case, it passes the following arguments along:

- file name of the test case file to be executed, for example “a01.py”;

- code of the chosen test case, for example “a01”;
- name of the chosen test case, for example “UART_RPi”;
- date and time of execution (two arguments);
- name of the file containing the parameters, i.e., “params.csv”.

For test cases where the Arduino must be programmed with specific code, the Python script configures the Arduino code file to contain the user-specified parameters, wherever required. It also writes a timestamp to the Arduino code which is reported to the Raspberry Pi via UART upon execution of the Arduino code, so the Python script can detect if the flashing of the code has been successful.

5.2.2 Test cases

Test cases are divided into two groups – “Arduino” and “other”. The first involves the Arduino microcontroller, which means that a Python script will flash it with specific code in order to execute the test case. The latter does not need any collaboration with the Arduino and runs solely on the Raspberry Pi.

The following list (Table 14) shows the minimum required test cases for a thorough testing of the PCB.

Table 14. List of test cases and their mainly targeted unit(s) under test.

Code	**Main unit(s) under test**	**Flashes Arduino**	**User-interactive**	**Self-evaluating**	**Completed**
a01	UART (Raspberry Pi)	●		●	●
a02	GPIO lines: ARDU_RST_LV ARDU_RST_HV	●		●	●
a03	UART (FTDI/USB)	●	●		●
a04	Battery voltage measurement	●	●	●	●
a05	UART (Bluetooth)	●	●		●

Table 15. List of test cases and their mainly targeted unit(s) under test [continued].

Code	**Main unit(s) under test**	**Flashes Arduino**	**User-interactive**	**Self-evaluating**	**Completed**
a06	GPIO lines: ARD2RPI_HV ARD2RPI_LV	●		●	●
a07	Bluetooth command mode	●	●		●
a08	I²C addresses	●		●	●
a09	Motor drivers: write (and read fault)	●	●		
a10	SPI bus	●		●	
a11	Motor feedback sensors: read	●	●	●	
a12	Motors: drive and check	●		●	
a13	Pressure sensor: read	●			
a14	IMU: read: Gyroscope data Accelerometer data Temperature data	●		●	
a15	Beacon signals	●	●	●	
a16	FSK: read and write (incl. messaging LEDs)	●	●		
o01	RGB LED		●		●
o02	Camera: operation				
o03	Camera: object recognition		●		

The table shows for each test case if the microcontroller will be programmed, if the test requires user-interaction, if the Raspberry Pi script can determine the success of the test without relying on user feedback and if the test case has been completed by the time of this thesis submission. The latter can be determined based on two factors: (a) the test case scripts and codes have been finished; (b) the execution of the test case yields positive results. Note that a test case, in order to function properly, may require more devices to operate correctly than the one stated as the main unit under test. This also means that some components on the PCB do not require dedicated test cases to be tested.

The test cases provided are explained briefly in the following subsections. Some of them have already been executed several times in order to test the capabilities of the new PCB and to verify the correct operation of its components. For some test cases, conclusions drawn from the executed tests are stated in Section 5.4.1.

The individual test cases consist of a Python script and – in some cases – additional files, such as a parameter file and an Arduino sketch.

The basic structure of all test cases is similar. Being called from the user interface script (as described in Section 5.2.1) with several strings as arguments, the test case script executes the following functions in the stated order (their main purposes are given to their right, respectively, and indented functions are called from the previous promoted function):

Table 15. Overview of common functions used in test case scripts. Functions indicated in blue are used only in test cases which involve the Arduino.

Function	**Purpose**
`init()`	Stores passed command-line arguments in an array.
`getParameters()`	Reads parameters from csv file and stores them in an array.
`setup()`	Sets up GPIO Raspberry Pi pins and UART (if necessary).
`switch()`	Sets multiplexer routing between Arduino and Raspberry Pi.
`make()`	Compiles the Arduino code into hex file using *make*.
`flash()`	Uploads the hex file to the microcontroller using *avrdude*.
`reset()`	Resets the microcontroller before upload.
`execute()`	Runs the actual test routine.
`analyze()`	Compiles the test report.
`printParameters()`	Prints all parameters and their values into the test report.
`printResults()`	Prints specific test results into the test report.

The `execute()` method is individual to each test case. For some test cases, user-interaction is required, and the `execute()` function contains functionality to guide the user through the test procedure.

make is open-source software for compiling code into executables based on definitions stated in a document called “Makefile”. It “controls the generation of executables and other non-source files of a program from the program's source files” [61]. The script calls to `make` using a method that emulates the call being issued from the terminal. This involves the use of the *os* library: `os.system(<command>)` makes it possible to execute from the Python script any command that could be used in the command-line terminal – where `<command>` is the command or line of commands to be issued. In the case of *make*, the command is `make -f <path to Makefile>`. In order to obtain useful executables from the source code, the compiler must know the target hardware architecture – in this case it is the ATmega328P microcontroller. Performing compilation on a processor architecture which is different from the target architecture is referred to as "cross-compiling", and the compiler commonly used is GCC (GNU compiler collection) – more specifically, the program `avr-gcc` is being called. *make* detects the target hardware from the parameters specified in the "Makefile" and then invokes the actual compiler. The result of the compilation comprises of a set of executables with file ending ".o" and ".elf". `make` invokes `avr-objcopy` to copy these output files into another format, thereby creating ".hex" files.

avrdude is open-source software developed to flash AVR microcontrollers [62]. It takes the content of the executable hex file created by the compiler and flashes it into the EEPROM of the microcontroller.

Each test case includes its individual parameters file in a csv format. Before the specific test case script is started, the main user interface gives the user opportunity to specify the parameter values and writes them into the csv file. The test case script reads the parameters from the file and uses them. Parameters for each test case are different. For finalized test cases, their parameters are listed in the following Table 16. For many parameters, their purpose is self-explanatory. For others, it is described in the following subsections, under the respective test case.

Table 16. Parameters used in test cases.

Test case code	**Parameter name in the file**	**Unit**	**Type**
a01	`Baudrate`	bit/s	Integer
a02	`Repetitions`		Integer
	`Pulse time`	s	Float
a03	`Baudrate`	bit/s	Integer
	`Repetitions`		Integer
a04	`R_H`	Ω	Integer
	`R_L`	Ω	Integer
	`Analog reference`	V	Float

Table 16. Parameters used in test cases [continued].

Test case code	Parameter name in the file	Unit	Type
a05	`Baudrate`	bit/s	Integer
	`Repetitions`		Integer
a06	`Repetitions`		Integer
	`Pulse time`	ms	Integer
	`Off-time`	ms	Integer
a07	`Bluetooth data baudrate`	bit/s	Integer
	`Bluetooth module name`		String
	`Bluetooth module password`		String
	`Max. attempts per parameter`		Integer
	`Max. time per parameter`	ms	Integer
	`Bluetooth AT baudrate`	bit/s	Integer
a08	`Addr. H-bridge R-R`		Integer (hex.)
	`Addr. H-bridge F-R`		Integer (hex.)
	`Addr. H-bridge F-L`		Integer (hex.)
	`Addr. H-bridge R-L`		Integer (hex.)
	`Addr. Pressure sensor`		Integer (hex.)
	`Addr. IMU sensor`		Integer (hex.)
	`Max. attempts each`		Integer
	`Delay in attempts`	ms	Integer
o01	`Set PWM frequency`	Hz	Integer

Each test case displays a report to the user after finishing execution and appends the same report to a text file located inside the respective test case folder for future reference.

UART between Arduino and Raspberry Pi (“a01”)

The Arduino code listens to the Serial port on a user-defined baud rate and echoes each message back to the sender.

The Python script generates a user-defined number of random ASCII (American standard code for information interchange) strings and sends each of them to the Arduino. Upon reception of the responses, it compares the sent strings to the received ones and determines the success of the test.

Resetting the Arduino via the Raspberry Pi (“a02”)

The Arduino code detects if the microcontroller has been reset by writing a specific integer to the EEPROM (Electrically erasable programmable read-only memory) at a specific address upon its first execution, if this string is not yet present in the EEPROM. If it is, it indicates that the code runs for the second time, after which the EEPROM content is overwritten with a zero value. The Arduino reports via UART to the Raspberry Pi if the code runs for the first or the second time.

The Python script waits for the Arduino to be ready (the Arduino reports this via UART) and then resets the microcontroller. It waits for an answer via UART if the microcontroller has been reset. The test is repeated for a specified number of times, as per user-definition.

Parameter `Repetitions` defines how many resets should be performed; parameter `Pulse time` sets the duration of time the Arduino reset pin will be pulled low.

UART between Arduino and FTDI (“a03”)

The Python script instructs the user to open a serial terminal. It generates a user-defined number of random ASCII strings and asks the user to enter them. The Arduino code, upon reception of a string, echoes it back via UART. The Python script asks the user to enter the received string into the Python terminal. When the test ends, the Arduino reports to the Raspberry Pi via UART all strings it has received. The Python script then analyses the results and determines how many transmissions were successful from user to Arduino and how many from Arduino to user. The test cannot determine if data got corrupted on the way between Arduino and Raspberry Pi; it is therefore recommended to execute test case “a01” before and verify that the connection is robust.

Parameter `Repetitions` sets the number of strings the user will send to the Arduino during this test routine.

Battery voltage measurement (“a04”)

When instructed via UART, the Arduino reads the analogue input which is connected to the output of the voltage divider for the battery voltage measurement. Using a proportionality constant calculated in the beginning of code execution by using user-defined values for the resistors, it determines the corresponding battery voltage value from the analogue input value and sends it to the Raspberry Pi via UART.

The applied battery voltage is not random but set by the user upon specific instructions by the Python script. Henceforth, the Python script knows the correct battery voltage value and calculates a calibration coefficient which is written into the Arduino sketch. The Python script then flashes the Arduino with the modified code and repeats the test execution. In the end, the optimal proportionality coefficient is reported to the user, along with a comparison of test results before and after calibration.

Parameter `R_H` defines the value of the resistor between battery voltage and analogue input pin used on the PCB, and parameter `R_L` defines the value of the resistor between analogue input pin and ground. Parameter `Analog reference` sets the voltage value used as reference for the ADC (analogue-to-digital converter) of the microcontroller. The user must take care that this value corresponds to the microcontroller specifications and to the Arduino code. For example, if the user sets this value to 5, the Python code will not automatically remove the line `analogReference(INTERNAL)` from the Arduino code file. The user must do that manually.

UART between Arduino and Bluetooth module ("a05")

The test works like "a03", but the user is instructed to open a terminal to read data from and send data to the Bluetooth device.

The parameter `Repetitions` sets the number of strings the user is expected to send via the Bluetooth terminal.

Signal line from Arduino to Raspberry Pi ("a06")

In use cases where the UART interface of the Arduino microcontroller is not routed to the Raspberry Pi, a signal line from Arduino to Raspberry Pi is provided. While the Raspberry Pi can change the routing of the UART line to itself in order to report a critical system state to the Arduino, the Arduino does not have control of the multiplexer and must therefore rely on an alternative communication pathway in case of emergencies. Messages sent through this line are expected to be short and simple – a sequence of pulses, for example. Using a pre-defined number of pulses within a specified frequency range, the Arduino can instruct the Raspberry Pi to execute one of a few functions. The most likely to occur case is that the Arduino would need to instruct the Raspberry Pi to route the UART line between them so it can send required messages.

For this test case, the user can specify pulse length, duration of off time between pulses and the number of cycles. In each executed message cycle, the number of pulses is incremented by one. The Python script uses an event listener to check for rising edges on the GPIO pin. For each pulse sequence, it counts the number of detected rising edges and determines if the number corresponds to the respective sequence count.

Parameter `Repetitions` defines the number of pulse sequences to send, `Pulse time` defines the on-time of a single pulse, `Off-time` defines the off-time between pulses in a sequence.

Bluetooth AT mode configuration (“a07”)

The main purpose of this testcase is to check if the Raspberry Pi can reset the Bluetooth module into AT command mode and if the Arduino can interface with it using AT commands [63]. To start the test case, UART is routed between Arduino and Raspberry Pi first. After the Python code has verified the integrity and responsiveness of the Arduino program, it instructs the Arduino code to start. Arduino responds with a pulse on the “ARD2RPI” line, which triggers the Raspberry Pi to reset the Bluetooth module into command mode, send a confirm code via UART and then route the Bluetooth module to the Arduino. From this moment on, the Python script and Arduino cannot communicate any more via UART and must therefore rely on the “ARD2RPI” line from Arduino to Raspberry Pi. Arduino sets its serial baud rate to the value required for communication with the Bluetooth module and issues a series of commands to the Bluetooth module, while recording the responses. It signals its completion to the Raspberry Pi by pulsing “ARD2RPI”. At this time, it resets its baud rate to the value required for communication with Raspberry Pi. When the Python script detects the pulse, it resets the Bluetooth module back into data mode and routes UART between Arduino and Raspberry Pi. It instructs the Arduino to send the test report via UART and evaluates the results, printing and storing a test report.

The first three parameters in the file define new values to be set to the Bluetooth module during configuration: the new baud rate to be used during data mode, the new name of the module and the new password to be used during pairing. The fourth and fifth parameter relate to cases when the Bluetooth module does not respond to a command being sent to it: How often should the Arduino retry, and how much time should it wait between the retries. The last parameter, `Bluetooth AT baudrate`, defines the baud rate to be used for communicating with the Bluetooth module in command mode.

I²C detection (“a08”)

With all addresses of I²C bus participants correctly defined in the parameters file, the test routine scans the I²C bus using the default Arduino “Wire” library’s public functions `beginTransmission()` and `endTransmission()`. The latter returns an error code if a device does not respond as expected or does not respond at all. The test case report states the success of the test as the ratio of error-free responses to the total number of devices expected on the bus.

The first six parameters in the file define the I²C addresses of the bus participants as hexadecimal values. The last two parameters refer to the case when a device does not respond correctly: How often should the Arduino retry on each device, and how much time should it wait between the retries.

RGB LED (“o01”)

This test is fully user-interactive and requires the user to visually verify if the colour of the LED matches with the prescribed output. He also must verify the smoothness of transitions. By entering the results as answers to simple “yes/no” questions posed by the Python script, the code can compile a final test report.

The script uses the *pigpio* library [64] in order to make use of hardware PWM capabilities on the Raspberry Pi Zero [54]. Tests have shown that PWM generated by software emulation may cause flickering.

This test allows the user to specify a PWM frequency which should be used on the three pins to which the LED is connected. As stated in the *pigpio* library documentation [64], the employed function `set_PWM_frequency()` returns the real PWM frequency set for the respective pin, according to hardware limitations. The test report includes the desired and actual PWM frequency. The function `get_PWM_range()` returns the value that should be used on a PWM pin for a duty cycle of 100%, and `get_PWM_real_range()` returns the number of distinct values to be actually used by hardware (i.e., the resolution). For higher frequencies, the resolution drops. This may be observed as flickering during brightness transitions.

To summarize, a good choice of PWM frequency prevents flickering at duty cycles below 100% caused by a too low frequency and prevents flickering during transitions caused by a too low resolution which might occur if the frequency is too high.

5.3 Supplementary software

To facilitate handling of the new PCB, two additional software pieces are provided as part of this thesis project.

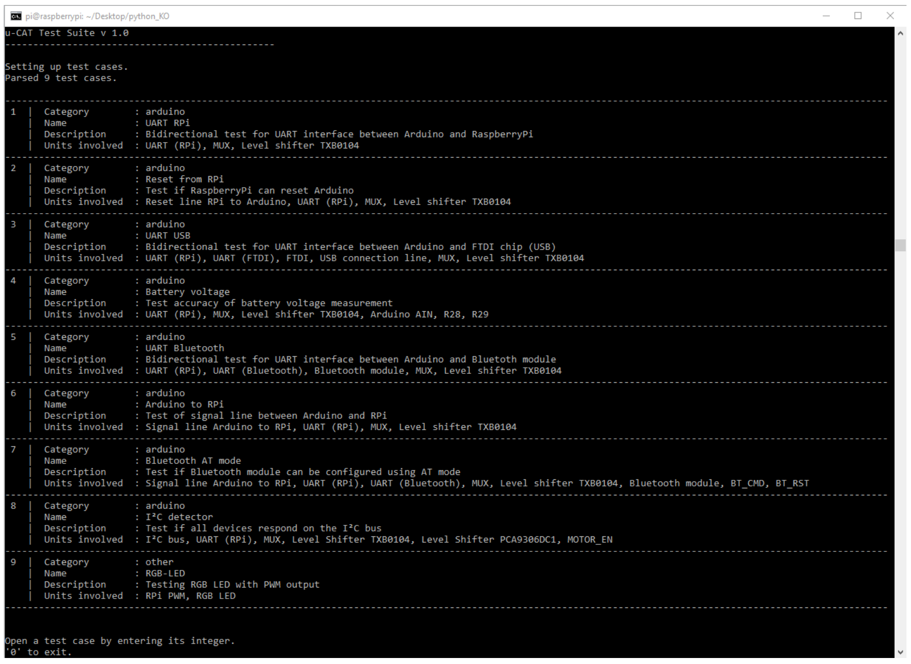
The Arduino code uploader, written in Python 3, can be used to upload any Arduino sketch from the Raspberry Pi to the Arduino microcontroller. When executed, it asks the user to enter the name of the ino file. The file must be located in a specific subdirectory. The path is `./arduino_test/<name>/<name>.ino`, where `<name>` is the string specified by the user. The Python script locates the file, then calls to the compiler (*make*), resets the microcontroller and executes *avrdude*.

The multiplexer switcher, also written in Python 3, facilitates the routing of UART to the Arduino. When executed, it presents a menu to the user with the three available UART clients – Raspberry Pi, FTDI chip and Bluetooth module. The user enters the respective integer, and the program instructs the multiplexer chip to set the routing accordingly.

5.4 Results and discussion

The following list summarizes all software features provided as part of the new robot design:

- Automated user-interactive test suite. The main user menu is shown in Figure 31.
- Ten test cases so far, automatically called by the test suite software. The software architecture provides a framework for software developers to implement more test cases, by following the existing software structure.
- Test results with generated reports. All test cases have been successfully executed and yielded positive results for the respective components on the new PCB.
- Arduino code uploader. Facilitates compilation and upload of Arduino software onto the microcontroller.
- Multiplexer switcher. Facilitates routing of Arduino’s UART bus to one of three devices.



```
pi@raspberrypi: ~/Desktop/python_KO
u-CAT Test Suite v 1.0
------------------------------------------------

Setting up test cases.
Parsed 9 test cases.

--------------------------------------------------------------------------------------------------------------------------------------------------------------
 1 |  Category        : arduino
   |  Name            : UART RPi
   |  Description     : Bidirectional test for UART interface between Arduino and RaspberryPi
   |  Units involved  : UART (RPi), MUX, Level shifter TXB0104
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 2 |  Category        : arduino
   |  Name            : Reset from RPi
   |  Description     : Test if RaspberryPi can reset Arduino
   |  Units involved  : Reset line RPi to Arduino, UART (RPi), MUX, Level shifter TXB0104
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 3 |  Category        : arduino
   |  Name            : UART USB
   |  Description     : Bidirectional test for UART interface between Arduino and FTDI chip (USB)
   |  Units involved  : UART (RPi), UART (FTDI), FTDI, USB connection line, MUX, Level shifter TXB0104
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 4 |  Category        : arduino
   |  Name            : Battery voltage
   |  Description     : Test accuracy of battery voltage measurement
   |  Units involved  : UART (RPi), MUX, Level shifter TXB0104, Arduino AIN, R28, R29
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 5 |  Category        : arduino
   |  Name            : UART Bluetooth
   |  Description     : Bidirectional test for UART interface between Arduino and Bluetoth module
   |  Units involved  : UART (RPi), UART (Bluetooth), Bluetooth module, MUX, Level shifter TXB0104
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 6 |  Category        : arduino
   |  Name            : Arduino to RPi
   |  Description     : Test of signal line between Arduino and RPi
   |  Units involved  : Signal line Arduino to RPi, UART (RPi), MUX, Level shifter TXB0104
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 7 |  Category        : arduino
   |  Name            : Bluetooth AT mode
   |  Description     : Test if Bluetooth module can be configured using AT mode
   |  Units involved  : Signal line Arduino to RPi, UART (RPi), UART (Bluetooth), MUX, Level shifter TXB0104, Bluetooth module, BT_CMD, BT_RST
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 8 |  Category        : arduino
   |  Name            : I²C detector
   |  Description     : Test if all devices respond on the I²C bus
   |  Units involved  : I²C bus, UART (RPi), MUX, Level Shifter TXB0104, Level Shifter PCA9306DC1, MOTOR_EN
--------------------------------------------------------------------------------------------------------------------------------------------------------------
 9 |  Category        : other
   |  Name            : RGB-LED
   |  Description     : Testing RGB LED with PWM output
   |  Units involved  : RPi PWM, RGB LED
--------------------------------------------------------------------------------------------------------------------------------------------------------------


Open a test case by entering its integer.
'0' to exit.
```

Figure 31. Main user menu of the test suite.

In terms of possibilities for a new software architecture, the implementation of two processors allows for code decoupling which keeps device interaction code separate and independent from high-level user definitions. The modular code structure of the test suite software illustrates this: The Python modules run on Raspberry Pi and include the user definitions for the test cases. The low-level code for the Arduino is configured by the test case scripts, and the Arduino only interfaces low-level devices. The analysis and presentation of test results lies again in the hands of the high-level Python code.

For mission use cases, the Raspberry Pi can store rules and policies, calculate navigation-related data and take care that certain actions are executed whenever a given condition is met. It sends its instructions to the Arduino via UART. The Arduino just executes low-level functions and does not need any “knowledge” of implemented policies or the overall mission.

This modularity may also play an important role in exceptional cases: For example, when the Arduino reports a critical battery voltage to the Raspberry Pi, the latter could, in accordance with the policies it holds, give commands to the Arduino microcontroller which cause the flippers to stop moving, so the robot would self-surface.

5.4.1 Representative test case results

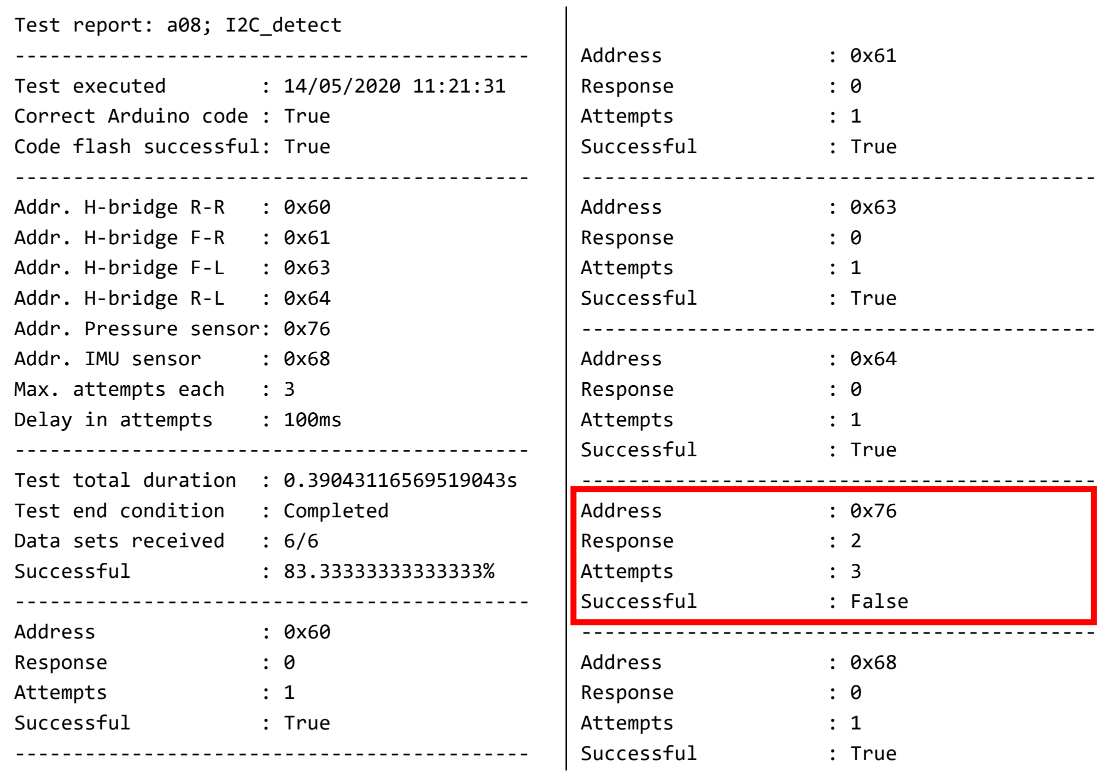
For debugging and verification of the new PCB and its components and for the purpose of verifying the working principle of the test software itself, all finished test cases (see

Section 5.2.2) have been executed at least once. The test software has been developed up to a state where it is fully functional, even though it has not yet been equipped with safeguards on user inputs. Some of the test results and conclusions drawn from them are presented in the following.

Example test result: I²C detection (“a08”)

A typical fault scenario for the I²C test is that an I²C device is not properly connected to the PCB. This is most likely to happen to the pressure sensor, since it uses a wire-to-board connection.

For the presented example, the pressure sensor is disconnected and then the test case executed. The test report looks like this:



```
Test report: a08; I2C_detect
--------------------------------------------
Test executed        : 14/05/2020 11:21:31
Correct Arduino code : True
Code flash successful: True
--------------------------------------------
Addr. H-bridge R-R   : 0x60
Addr. H-bridge F-R   : 0x61
Addr. H-bridge F-L   : 0x63
Addr. H-bridge R-L   : 0x64
Addr. Pressure sensor: 0x76
Addr. IMU sensor     : 0x68
Max. attempts each   : 3
Delay in attempts    : 100ms
--------------------------------------------
Test total duration  : 0.39043116569519043s
Test end condition   : Completed
Data sets received   : 6/6
Successful           : 83.33333333333333%
--------------------------------------------
Address              : 0x60
Response             : 0
Attempts             : 1
Successful           : True
--------------------------------------------
Address              : 0x61
Response             : 0
Attempts             : 1
Successful           : True
--------------------------------------------
Address              : 0x63
Response             : 0
Attempts             : 1
Successful           : True
--------------------------------------------
Address              : 0x64
Response             : 0
Attempts             : 1
Successful           : True
--------------------------------------------
Address              : 0x76
Response             : 2
Attempts             : 3
Successful           : False
--------------------------------------------
Address              : 0x68
Response             : 0
Attempts             : 1
Successful           : True
```

Figure 32. Test report for test case “a08”, with no pressure sensor connected. Highlighted: the non-responsive (missing) device.

The user can see that the property “Successful” did not reach 100%. Going through the individual address responses, it becomes apparent that address `0x76` did not respond, which is, according to the parameter listing in the beginning of the test report, the pressure sensor.

Example test result: Signal line from Arduino to Raspberry Pi (“a06”)

By executing this test several times, it was found that the minimum required pulse length is 60ms. The robustness can be significantly improved if the off time is even longer (90ms). These required lower limits are probably a result of capacitor charge and discharge curves appearing on the signal line when switching between states.

Example test result: RGB LED (“o01”)

Lower frequencies may lead to observable flickering due to the long on- and off-times of the PWM signal, while higher frequencies reduce the resolution of the PWM output too much. Running the test can help find the appropriate PWM frequency. The test result shows that the real range (resolution) is only 25 distinct values for a frequency of 8000Hz and above. Flickering can be observed in low brightness. To avoid flickering, a PWM frequency between 1000Hz and 5000Hz appears to be reasonable.

5.4.2 Future work

The list of required test cases has been presented in Section 5.2.2. The not yet implemented test cases will be scripted in the near future, to allow a complete thorough testing of the PCB.

The user interface and test cases require safeguards for user inputs which are not provided in the current versions.

6 Cost estimation

One main goal in the redesign of µ-CAT is to keep production cost low. This guarantees for µ-CAT to remain economically competitive in the class of small-scale robots, even after the design upgrade.

Software and hardware development cost for manhours are not included. Utilized IDEs are all freeware or no-cost open-source platforms and implemented libraries from third parties are licensed as open-source software and also free of charge.

Table 17. Cost estimation per unit.

Part(s), material or assembly	**Cost per robot, net [€]**
Physical structure	
Plexiglass tube	3.29
Plexiglass parts, laser-cut	7.60
3D-printed parts	51.50
Steel parts and misc.	16.98
Mechanical components	
Silicone flippers	5.37
Motor shaft assemblies	15.13
Electronic and electrical components	
Main PCB	12.00
Main PCB soldering	20.00
Raspberry Pi Zero W	12.00
Raspberry Pi camera	23.19
Bluetooth PCB and cable	21.00
LED PCB and cable	5.00
Motors	90.00
Electronic components	157.13
Total:	**440.18**

As Table 17 shows, the goal to stay below 500 € (see Section 2.2.1) has been met.

7 Summary

In summary, this thesis work presents the design and development steps which lead to the production of an upgraded version of µ-CAT, resulting in a robot which is more robust, easier to debug and has potential for field missions. During the design, care has been taken that these upgrades consider the cost margin and that they are relevant for rising the robot's significance in the field of small-scale underwater robotics. The latter has been accomplished by improving computational and underwater communication abilities, by providing a reliable hardware architecture for a collaboration of two processors and by providing means of debugging and monitoring, both in designed hardware and provided software. Apart from that, µ-CAT's producibility has been facilitated and user interfacing with the robot has been improved.

The design of a new motherboard and the redesign of the 3D-printed end caps form a central part of this thesis work. Integration of new components has been accomplished both on electronic and on physical level. In doing so, ease of assembly of the robot has been improved.

µ-CAT is now ready to be equipped with even more functionality – plans potentially include the addition of more sensing capabilities, such as bioinspired sensing. The extendable test software framework enables future users to implement their own test cases with ease. Henceforth, the implementation of new features is highly simplified due to the provision of this software. The aspect of extendibility is also important for µ-CAT as a teaching tool, being more approachable to students for implementation of new functionalities. On top of that, it can be used to learn about ROS for mobile robotics and about distributed control architectures which are enabled by the multi-processor configuration. Being more easily reproducible, many students can now work and experiment on µ-CAT. Regarding µ-CAT's applicability to field work, it has made an important step towards being used in a swarm and offers a higher level of autonomy due to the integration of the camera. The increased computational power allows for advanced use in various research areas, such as bioinspired locomotion or sensing.

Bibliography

[1] B. Allotta *et al.*, “The ARROWS project: Adapting and developing robotics technologies for underwater archaeology,” in *IFAC-PapersOnLine*, 2015, vol. 28, no. 2, pp. 194–199.

[2] “Arrows Project.” [Online]. Available: http://www.arrowsproject.eu/. [Accessed: 23-Mar-2020].

[3] T. Salumäe *et al.*, “Design principle of a biomimetic underwater robot U-CAT,” in *2014 Oceans - St. John’s*, 2014, pp. 1–5.

[4] D. T. Roper, S. Sharma, R. Sutton, and P. Culverhouse, “A review of developments towards biologically inspired propulsion systems for autonomous underwater vehicles,” *Proc. Inst. Mech. Eng. Part M J. Eng. Marit. Environ.*, vol. 225, no. 2, pp. 77–96, May 2011.

[5] R. Salazar, V. Fuentes, and A. Abdelkefi, “Classification of biological and bioinspired aquatic systems: A review,” *Ocean Eng.*, vol. 148, pp. 75–114, 2018.

[6] A. Raj and A. Thakur, “Fish-inspired robots: design, sensing, actuation, and autonomy—a review of research,” *Bioinspir. Biomim.*, vol. 11, no. 3, p. 031001, Apr. 2016.

[7] R. Salazar, A. Campos, V. Fuentes, and A. Abdelkefi, “A review on the modeling, materials, and actuators of aquatic unmanned vehicles,” *Ocean Eng.*, vol. 172, pp. 257–285, 2019.

[8] W. H. Wang, R. C. Engelaar, X. Q. Chen, and J. G. Chase, “The State-of-Art of Underwater Vehicles - Theories and Applications,” in *Mobile Robots - State of the Art in Land, Sea, Air, and Collaborative Missions*, 2009.

[9] S. A. Watson, “Mobile Platforms for Underwater Sensor Networks,” University of Manchester, 2012.

[10] A. B. Phillips *et al.*, “Agile design of low-cost autonomous underwater vehicles,” in *OCEANS 2017 - Aberdeen*, 2017, vol. 2017-Octob, pp. 1–7.

[11] D. Goldberg, “Huxley: A flexible robot control architecture for autonomous underwater vehicles,” in *OCEANS 2011 IEEE - Spain*, 2011.

[12] D. Kortenkamp, R. Simmons, and D. Brugali, “Robotic systems architectures and programming,” in *Springer Handbook of Robotics*, Springer International Publishing, 2008, pp. 187–204.

[13] K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru, and G. A. Demetriou, “Control architectures for autonomous underwater vehicles - IEEE Journals & Magazine,” *IEEE Control Syst. Mag. (Volume 17, Issue 6, Dec. 1997)*, 1997.

[14] K. Wang, M. Tan, and J. Zhang, “Design and Control of an Embedded Vision Guided Robotic Fish with Multiple Control Surfaces,” *Sci. World J.*, 2014.

[15] O. Hu, J. Liu, I. Dukes, and G. Francis, “Design of 3D swim patterns for autonomous robotic fish,” in *IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 2406–2411.

[16] J. Yu and C. Wei, “Towards development of a slider-crank centered self-propelled dolphin robot,” *Adv. Robot.*, vol. 27, no. 12, pp. 971–977, 2013.

[17] C. Niu, L. Zhang, S. Bi, and Y. Cai, “Development and depth control of a robotic fish mimicking cownose ray,” in *2012 IEEE International Conference on*

Robotics and Biomimetics, ROBIO 2012 - Conference Digest, 2012, pp. 814–818.
[18] W. Wang, D. Gu, and G. Xie, “Autonomous Optimization of Swimming Gait in a Fish Robot with Multiple Onboard Sensors,” *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 49, no. 5, pp. 891–903, 2019.
[19] W. Wang and G. Xie, “CPG-based locomotion controller design for a boxfish-like robot,” *Int. J. Adv. Robot. Syst.*, vol. 11, no. 1, 2014.
[20] D. Lachat, A. Crespi, and A. J. Ijspeert, “BoxyBot: A swimming and crawling fish robot controlled by a central pattern generator,” in *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006, BioRob 2006*, 2006, vol. 2006, pp. 643–648.
[21] W. Wang, J. Liu, G. Xie, L. Wen, and J. Zhang, “A bio-inspired electrocommunication system for small underwater robots,” *Bioinspiration and Biomimetics*, vol. 12, no. 3, 2017.
[22] H. Liu and O. Curet, “Swimming performance of a bio-inspired robotic vessel with undulating fin propulsion,” *Bioinspiration and Biomimetics*, vol. 13, no. 5, 2018.
[23] “Sepios: Nautical Robot.” [Online]. Available: https://sepios.org/#project. [Accessed: 15-May-2020].
[24] M. P. Möller *et al.*, “Focus Project Sepios (report),” 2014.
[25] J. Busquets *et al.*, “Low-cost AUV based on Arduino open source microcontroller board for oceanographic research applications in a collaborative long term deployment missions and suitable for combining with an USV as autonomous automatic recharging platform,” in *2012 IEEE/OES Autonomous Underwater Vehicles, AUV 2012*, 2012.
[26] C. Wang, Z. Hu, Y. Yang, L. Geng, and L. Wang, “Control system design for micro AUV based on open source hardware,” in *2018 IEEE International Conference on Information and Automation, ICIA 2018*, 2018, pp. 980–984.
[27] B. Meyer, K. Ehlers, C. Isokeit, and E. Maehle, “The development of the modular hard- and software architecture of the autonomous underwater vehicle MONSUN,” in *Proceedings for the Joint Conference of ISR 2014 - 45th International Symposium on Robotics and Robotik 2014 - 8th German Conference on Robotics, ISR/ROBOTIK 2014*, 2014, pp. 258–263.
[28] C. Osterloh, T. Pionteck, and E. Maehle, “MONSUN II: A small and inexpensive AUV for underwater swarms,” in *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, 2012, pp. 1–6.
[29] M. Sfakiotakis, R. Gliva, and M. Mountoufaris, “Steering-plane motion control for an underwater robot with a pair of undulatory fin propulsors,” in *24th Mediterranean Conference on Control and Automation, MED 2016*, 2016, pp. 496–503.
[30] K. H. Low, C. Zhou, G. Seet, S. Bi, and Y. Cai, “Improvement and testing of a robotic Manta Ray (RoMan-III),” in *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011*, 2011, pp. 1730–1735.
[31] Z. Wu, J. Liu, J. Yu, and H. Fang, “Development of a Novel Robotic Dolphin and Its Application to Water Quality Monitoring,” *IEEE/ASME Trans. Mechatronics*, vol. 22, no. 5, pp. 2130–2140, 2017.
[32] W. Wang, G. Xie, and H. Shi, “Dynamie modeling of an ostraciiform robotic fish based on angle of attack theory,” in *Proceedings of the International Joint Conference on Neural Networks*, 2014, pp. 3944–3949.
[33] “ROS.org | History.” [Online]. Available: http://www.ros.org/history/. [Accessed: 22-Mar-2018].

[34] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” in *ICRA 2009*, 2009.
[35] A. Gonibedu Dathatri, “An optical communication protocol between underwater robots,” Tallinn University of Technology, 2020.
[36] “Shape Effects on Drag.” [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/airplane/shaped.html. [Accessed: 13-May-2020].
[37] B. Lautrup, *Physics of Continuous Matter*. Boca Raton: CRC press, 2011.
[38] Compaq *et al.*, “Universal Serial Bus Specification Rev. 2.0,” 2000.
[39] Compaq *et al.*, “USB Engineering Change Notice: USB 2.0 VBUS Max Limit,” 2014.
[40] “What is Voltage Regulator and How Does It Work?” [Online]. Available: https://components101.com/articles/what-is-voltage-regulator-and-how-does-it-work. [Accessed: 13-Mar-2020].
[41] “Buck Converter: Basics, Working, Design and Operation.” [Online]. Available: https://components101.com/articles/buck-converter-basics-working-design-and-operation. [Accessed: 13-Mar-2020].
[42] “LM2596 datasheet.” [Online]. Available: http://www.ti.com/lit/ds/symlink/lm2596.pdf. [Accessed: 12-Mar-2020].
[43] “MIC29302 datasheet.” [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/20005685a.pdf. [Accessed: 06-Apr-2020].
[44] “LM567x datasheet.” [Online]. Available: http://www.ti.com/lit/ds/snosbq4e/snosbq4e.pdf. [Accessed: 24-Mar-2020].
[45] M. Patel, “Implementation of FSK Modulation and Demodulation using CD74HC4046A,” 2013.
[46] “FDC6401N datasheet.” [Online]. Available: https://www.mouser.com/datasheet/2/149/FDC6401N-77688.pdf. [Accessed: 17-Apr-2020].
[47] “TS971, TS972, TS974 datasheet.” [Online]. Available: https://www.st.com/resource/en/datasheet/ts971.pdf. [Accessed: 17-Apr-2020].
[48] “MCP6021/1R/2/3/4 datasheet.” [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/20001685e.pdf. [Accessed: 20-Apr-2020].
[49] “DURIS S 8 GW P9LR32.EM datasheet.” [Online]. Available: https://dammedia.osram.info/media/resource/hires/osram-dam-5589023/GW P9LR32.EM_EN.pdf.
[50] “DURIS® S 8, GW P9LR34.PM | OSRAM OS.” [Online]. Available: https://www.osram.com/ecat/DURIS® S 8 GW P9LR34.PM/de/de/class_pim_web_catalog_103489/global/prd_pim_device_4269724/. [Accessed: 14-Apr-2020].
[51] “DURIS® S 8, GW P9LR35.PM | OSRAM OS.” [Online]. Available: https://www.osram.com/ecat/DURIS® S 8 GW P9LR35.PM/de/de/class_pim_web_catalog_103489/global/prd_pim_device_5431473/#62a1fdff42f13778e2055fdc87c50fcb. [Accessed: 14-Apr-2020].
[52] “SparkFun FTDI Basic Breakout - 5V - DEV-09716 - SparkFun Electronics.” [Online]. Available: https://www.sparkfun.com/products/9716. [Accessed: 09-Apr-2020].
[53] C.-M. Lu, “Communication system for devices with UART interfaces,” US7650449B2, 2010.

[54] “Hardware PWM with Raspberry Pi Zero – Codecubix.” [Online]. Available: https://www.codecubix.eu/linux/hardware-pwm-with-raspberry-pi-zero/. [Accessed: 14-May-2020].
[55] “CLP6C-FKB datasheet.” [Online]. Available: http://www.farnell.com/datasheets/2003905.pdf?_ga=2.143229493.1223965175.1585138498-627741049.1504615003. [Accessed: 25-Mar-2020].
[56] “ICM-20608-G datasheet.” [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/03/DS-000081-v1.01.pdf. [Accessed: 09-Apr-2020].
[57] “MS5837-02BA datasheet.” [Online]. Available: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FMS5837-02BA01%7FA7%7Fpdf%7FEnglish%7FENG_DS_MS5837-02BA01_A7.pdf%7FCAT-BLPS0059. [Accessed: 23-Mar-2020].
[58] “DRV8830 datasheet.” [Online]. Available: http://www.ti.com/lit/ds/symlink/drv8830.pdf. [Accessed: 23-Mar-2020].
[59] “E-GIZMO EGBT-045MS hadware manual.” [Online]. Available: https://www.manualslib.com/manual/1499691/E-Gizmo-Egbt-045ms.html. [Accessed: 25-Mar-2020].
[60] “TXB0104 datasheet.” [Online]. Available: http://www.ti.com/lit/ds/symlink/txb0104.pdf. [Accessed: 25-Mar-2020].
[61] “Make - GNU Project - Free Software Foundation.” [Online]. Available: https://www.gnu.org/software/make/. [Accessed: 14-May-2020].
[62] “AVRDUDE - AVR Downloader/UploaDEr.” [Online]. Available: https://www.nongnu.org/avrdude/. [Accessed: 27-Mar-2020].
[63] “HC-05 AT commands,” 2011. [Online]. Available: http://www.linotux.ch/arduino/HC-0305_serial_module_AT_commamd_set_201104_revised.pdf. [Accessed: 08-May-2020].
[64] “pigpio library.” [Online]. Available: http://abyz.me.uk/rpi/pigpio/. [Accessed: 05-May-2020].
[65] Texas Instruments, “LM1117 800-mA Low-Dropout Linear Regulator 1 Features 3 Description.” 2016.
[66] “MS54XX datasheet.” [Online]. Available: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=MS54XX&DocType=DS&DocLang=English. [Accessed: 19-May-2020].
[67] P. Höjerslev, “Raspberry Pi Camera | 3D CAD Model Library | GrabCAD.” .
[68] “Hexagon Full Nut to DIN 934 ~ISO 4032 | 3D CAD Model Library | 3D ContentCentral.” .
[69] J. Head, “Comus GC2322 Reed Switch | 3D CAD Model Library | GrabCAD.” .

Appendix

A 1 Experimental results

This appendix section presents the experimental setups, procedures and results carried out in order to determine the applicability of the chosen buck converter IC LM2596S.

A 1.1 Choice of buck converter

A 1.1.1 Comparison of two buck converters

For a buck converter to be able to provide stable output voltage, the voltage difference between its input and output must be sufficiently high. In this regard, using a buck converter at the 3.3V rail would have made sense. However, the current demand on this rail is insignificantly low, and the increased cost and space of a buck converter would not be justified by the amount of saved energy on the small logic devices and sensors which operate on that rail.

Whether or not a buck converter can be used on the 5V or 6V rail must be found out experimentally, and two buck converters are being compared for this purpose: the HW-468 breakout board and LM2596S. The key question in this first experimental procedure is if the buck converter can provide the required output voltage at low battery levels (6.8V, as assumed by measurements for a drained battery).

The HW-468 is a fixed-output voltage converter (5V) which accepts variable input voltages ranging approximately from 8V to 24V, and LM2596S exists as fixed- and adjustable-output voltage versions. For this experiment, the adjustable version is used, but it has the same efficiency and operating characteristics as the respective fixed-output voltage version. Both buck converters are compared with respect to their potential usability on the 5V rail.

For this set of experiments, the following equipment is used:

▪ Load device:	Centre for Biorobotics proprietary (based on SPW20N60S5)
▪ Power supply:	Instek PSP-405

- Multimeter: Fluke 87V

The load device allows to create a specific load scenario by defining a certain value of current to be drawn.

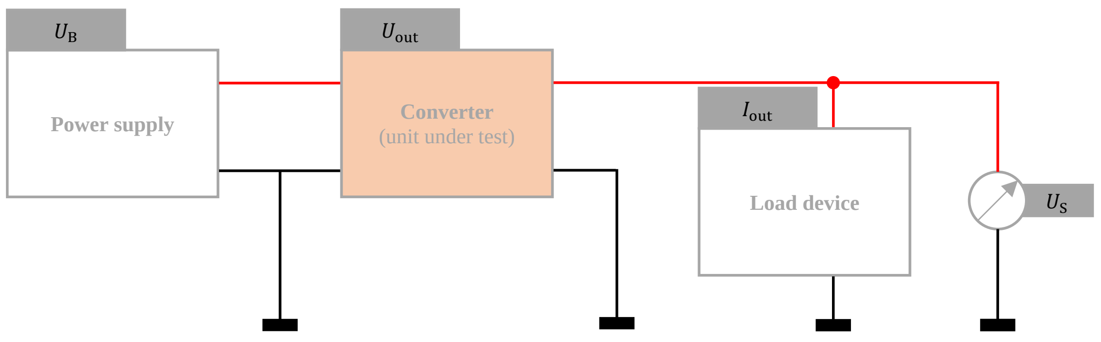


Figure 33. Experimental setup for sets 1 and 2.

Experiment 1: HW-468 applicability for 5V output

The input voltage of the buck converter required to drive the load at the desired current demand depends on the power supplied to the load. A current setpoint for the load was chosen (using the load device) and the input voltage was varied, starting from 35V (maximum value of power supply) down to a value where the buck converter could not provide a stable output of 5V any more (the measured stable value was 5.06V). If the value of supply voltage was significantly higher than the required supply voltage on µ-CAT (assuming drained batteries, i.e., 6.8V) at the instance when the output voltage fell below 5.06V, the converter would not be applicable for our purposes.

The experimental results show that even for a current demand as low as 0.4A on the output, the required input voltage is at least 8.27V, which is more than µ-CAT allows. This corresponds roughly to the information stated in the datasheet of the converter module (minimum 9V supply voltage required). Since the physical design of the robot (space restriction) does not allow to place more batteries in series (increasing the supply voltage) and since no other type of battery with a smaller form factor could be found which would be convenient to insert (cylindrical shape in order to maintain the tube and screw-cap design for water-proofing), there is no other conclusion to be made from this experimental result than to discard this buck converter module for our purposes.

Experiment 2: LM2596S applicability for 5V output

The buck converter IC is used on a breakout board featuring the flywheel circuit according to the recommendations in the datasheet [65]. In this test setup, the input voltage is kept at 6.8V and the output voltage of the buck converter initially adjusted to a fixed value of 5V, using the onboard multiturn trimmer. While observing the output voltage on the load, the current demand was increased in several steps, using the load device.

The experimental results show that the output voltage remains stable up to a current demand of 2A, which is enough for µ-CAT. Consequently, the device may be appropriate for our purposes. The results indicate that it would be worth testing the device regarding other aspects such as output noisiness and whether it could be used on the 6V rail.

A 1.1.2 Comparison of chosen buck converter and two voltage regulators

In this section, several experimental procedures and their results are presented which were carried out for the purpose of performance and efficiency comparison between the chosen buck converter LM2596S and the voltage regulators originally used on the robot to provide 5V and 6V to the mainboard, namely 29151-5.0 for 5V and 29302WT for 6V. The experiment was carried out in order to decide whether a buck converter should replace one or two of the originally employed voltage regulators in the updated design.

In all following experimental sets, the desired load current I_{out} [A] was set to a fixed value to simulate a certain load scenario. The battery supply current I_{B} [A] was observed during experimentation. For the linear voltage regulators, $I_{\text{B}} \approx I_{\text{out}}$, because of the power loss I the form of heat due to the voltage drop across the LDO (low-dropout regulator), for the buck converter, I_{B} is expected to be lower than I_{out} because of the law of energy preservation and due to the fact that the current must be stepped up while the voltage is being stepped down. The output voltage of the converter U_{out} [V] was fixed for to the linear voltage regulators, or it was set to a desired value before experimentation for the buck converter using the onboard potentiometer. During the experiments, its value was monitored as U_{S} [V]. Additionally, the temperature on the converter IC was sporadically measured and recorded as T [°C]. To be able to make a comparison of the noisiness of the output signal, snapshots were taken from the oscilloscope plots. In each set, measurements have been repeated several times, with minimum one minute of time between them, to monitor changes in performance over prolonged operation. The supply voltage to the converter circuit was set to $U_B = 7.4V$, to simulate operation of the device on the robot at average battery supply voltage over time (value determined experimentally before), assuming initially fully charged batteries.

For this set of experiments, the following equipment was used:

Load device:	Centre for Biorobotics proprietary (based on SPW20N60S5)
Power supply:	Instek PSP-405
Multimeters:	Fluke 87V (2x)
Oscilloscope:	Agilent DSO-X 3014A
Thermometer:	Fluke 561 IR

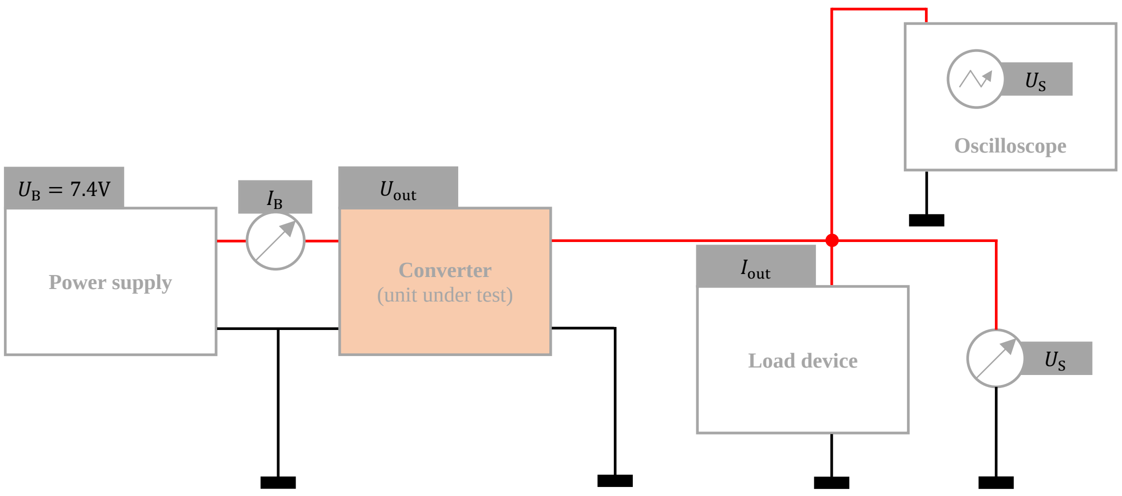


Figure 34. Experimental setup for sets 3 to 7.

Experiment 3: LM2596S vs. 29302WT; motors at stall current (6V, 2A)

If the buck converter can supply the load with stable 6V at a current demand of 2A, it can be used to drive the motors safely, even when they are all stalled at the same time.

In a first step, the performance of the originally implemented voltage regulator 29302WT has been assessed.

Table 18. Performance data for voltage regulator 29302WT.

t **[s]**	***I*****out** **[A] Set**	***I*****B** **[A] Observed**	***U*****S** **[V] Observed**	***T*** **[°C] Observed**	**Image**
0	2.004	2.077	5.762		
60	2.000	2.072	5.762	28.0	
120	2.000	2.071	5.745		
360	1.999	2.067	5.813	28.6	
600	1.997	2.067	5.824		Figure 35

The above data (Table 18) shows that the linear voltage regulator is able to provide 2A of output current at a stable output voltage of close to 6V, enough to get the motors out of the stall position.

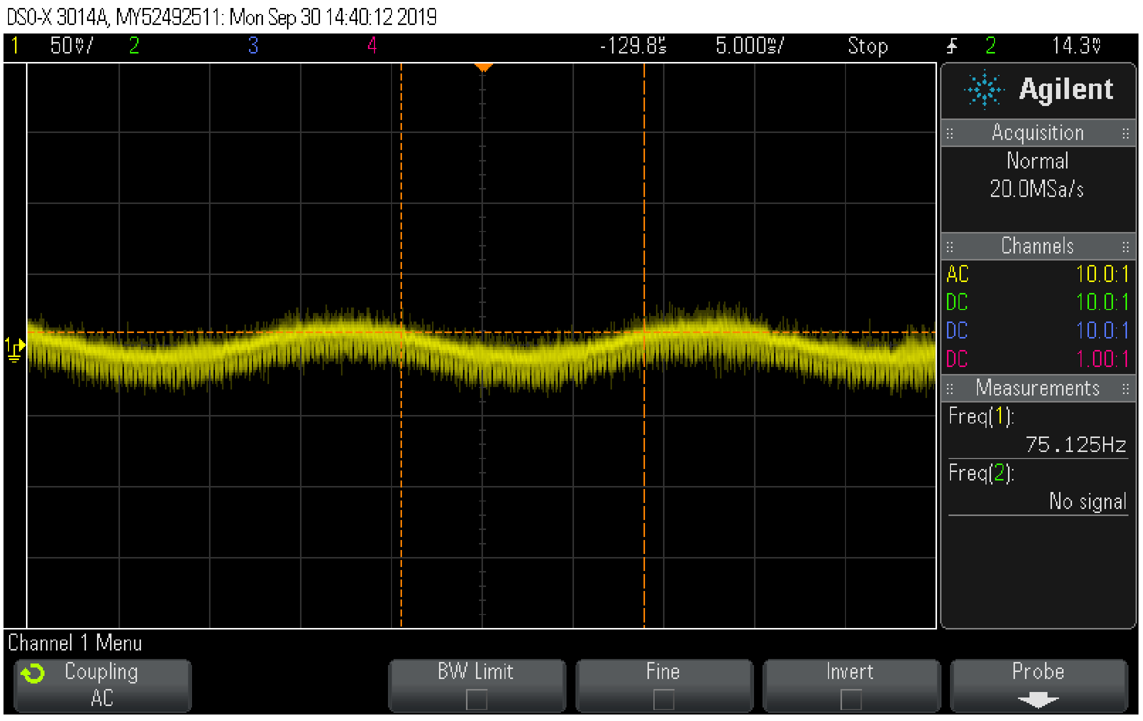


Figure 35. Signal on output of 29302WT at 2A, after 10 minutes of operation.

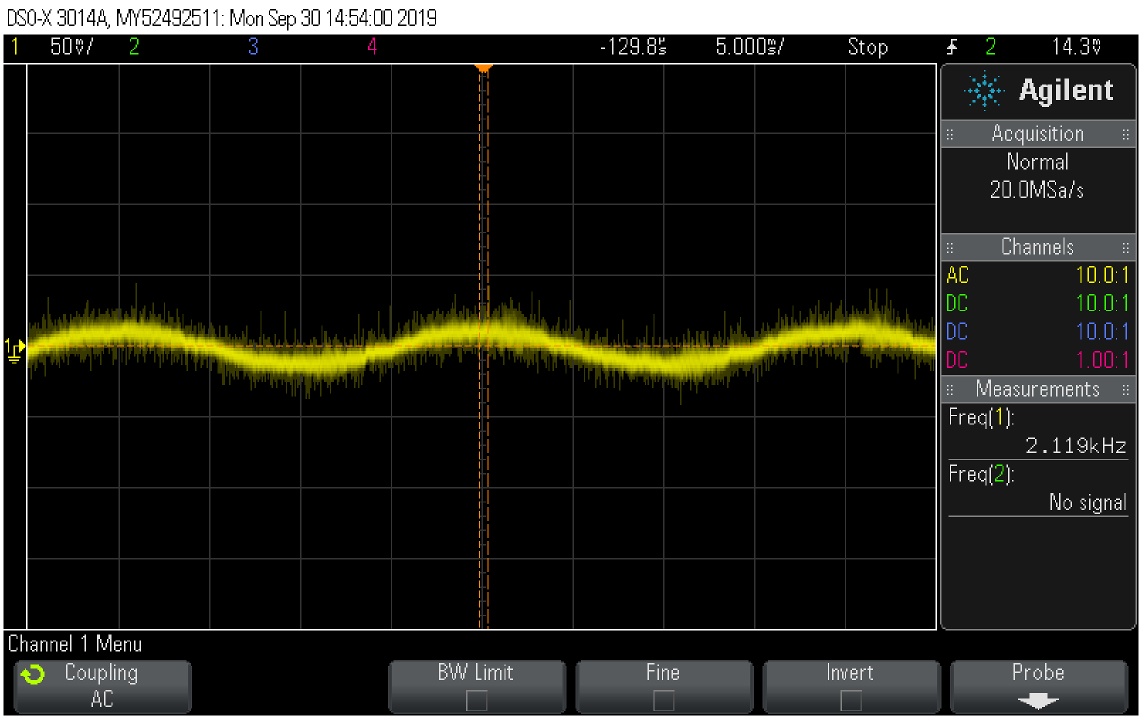
The high-frequency noise level remains in a corridor of approximately 30mV. The low-frequency noise (50Hz) with a magnitude of around 50mV occurs most likely due to the mains AC power supply, which means that it is insignificant for the real-life case on the robot.

After that, the buck converter LM2596S was tested, with the output voltage adjusted to $U_{\text{out}} = 6\text{V}$.

Table 19. Performance data for buck converter LM2596S.

t [s]	I_{out} [A] Set	I_B [A] Observed	U_S [V] Observed	T [°C] Observed	Image
120	2.003	1.825	5.387		
180	2.000	2.008	5.385		Figure 36
660	2.000	2.008	5.389		
840	1.998	2.006	5.379	72.0	
900	1.997	2.005	5.376	70.0	
1020	1.997	2.005	5.375		Figure 37
1080	1.997	2.005	5.374		

Above data (Table 19) indicates that the buck converter is not able to provide the demanded output voltage of 6V at the current demand of 2A. It may still be enough to get the motors out of stalling, but it cannot be guaranteed. Also, the heat dissipation on the buck converter IC is so high that it would require a massive heatsink, which should be avoided in the PCB design for the sake of space-saving.


DSO-X 3014A, MY52492511: Mon Sep 30 14:54:00 2019

Freq(1): 2.119kHz
Freq(2): No signal

Figure 36. Signal on output of LM2596S at 2A, after 3 minutes of operation.

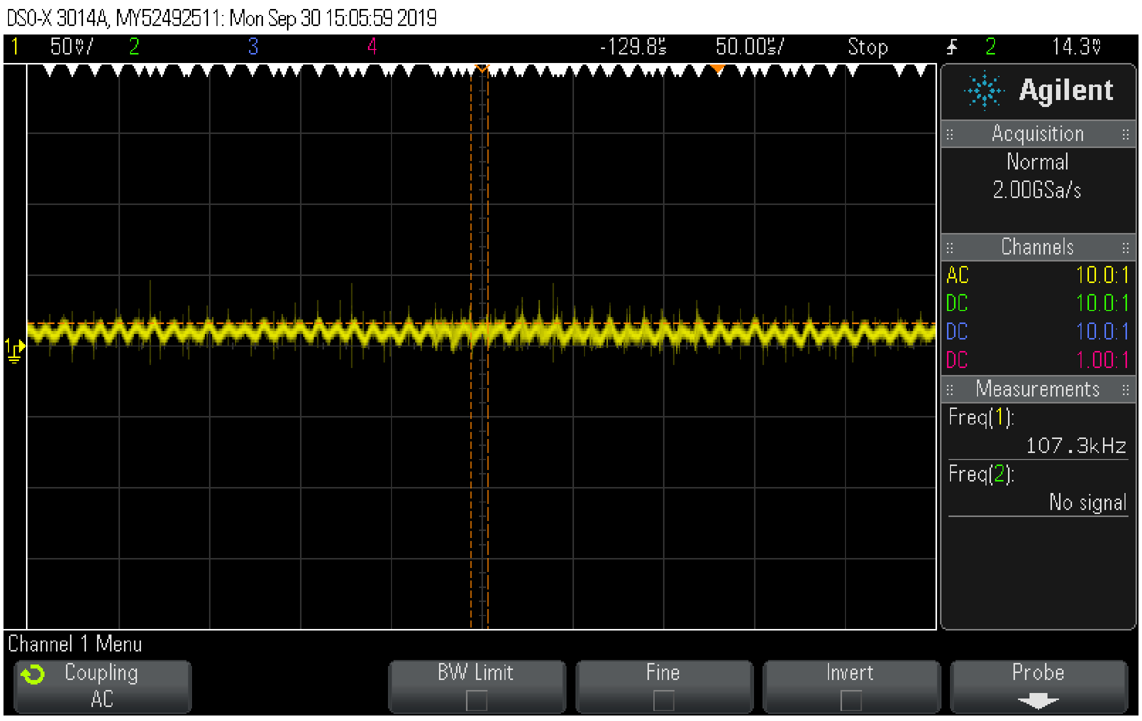


Figure 37. Signal on output of LM2596S at 2A, after 17 minutes of operation.

The output signal plotted on the oscilloscope (Figure 36 and Figure 37) shows that the high-frequency noise level is around 50mV, sometimes exceeding up to 70mV. As expected, the buck converter produces more noise than the LDO.

Experiment 4: LM2596S vs. 29302WT; motors (6V, 0.55A)

As previous measurements showed, the motors combined consume around 550mA at 6V during regular operation. The aim of this experimental set is to find out if the buck converter can provide a stable output to do so.

The originally existing voltage regulator 29302WT was tested first.

Table 20. Performance data for voltage regulator 29302WT.

t [s]	I_{out} [A] Set	I_{B} [A] Observed	U_{S} [V] Observed	T [°C] Observed	Image
0	0.557	0.575	5.896	30.0	
60	0.557	0.575	5.896		Figure 38
120	0.557	0.575	5.896		
180	0.557	0.575	5.896		
240	0.557	0.575	5.896		

The input power consumption is:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 0.557\text{A} = 4.12\text{W} \tag{29}$$

As Table 20 shows, the LDO provides stable output of nearly 6V to drive the motors at the demanded current. The heat dissipation on the device is moderate.

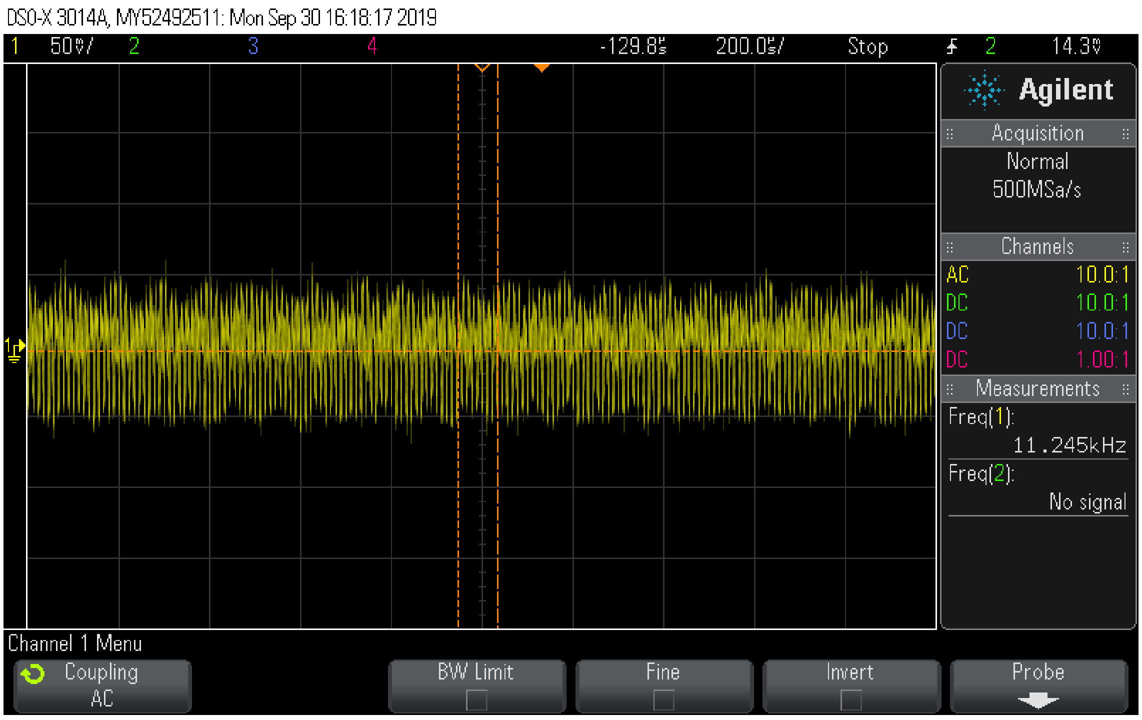


Figure 38. Signal on output of 29302WT at 557mA, after 1 minute of operation.

As Figure 38 shows, the high-frequency noise level is around 100mV.

The tests conducted on the buck converter are presented in the following.

Table 21. Performance data for buck converter LM2596S.

t **[s]**	I_{out} **[A] Set**	I_{B} **[A] Observed**	U_{S} **[V] Observed**	***T*** **[°C] Observed**	**Image**
0	0.557	0.544	6.000		
60	0.577	0.544	6.000		
120	0.577	0.544	5.990		Figure 39
240	0.577	0.544	5.990	38.1	

Indeed, the buck converter can maintain a stable output voltage at the desired current, as Table 21 shows. A mathematical comparison with the voltage regulator yields that the buck converter saves 2% of input power when used for driving the motors:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 0.544\text{A} = 4.03\text{W} \tag{30}$$

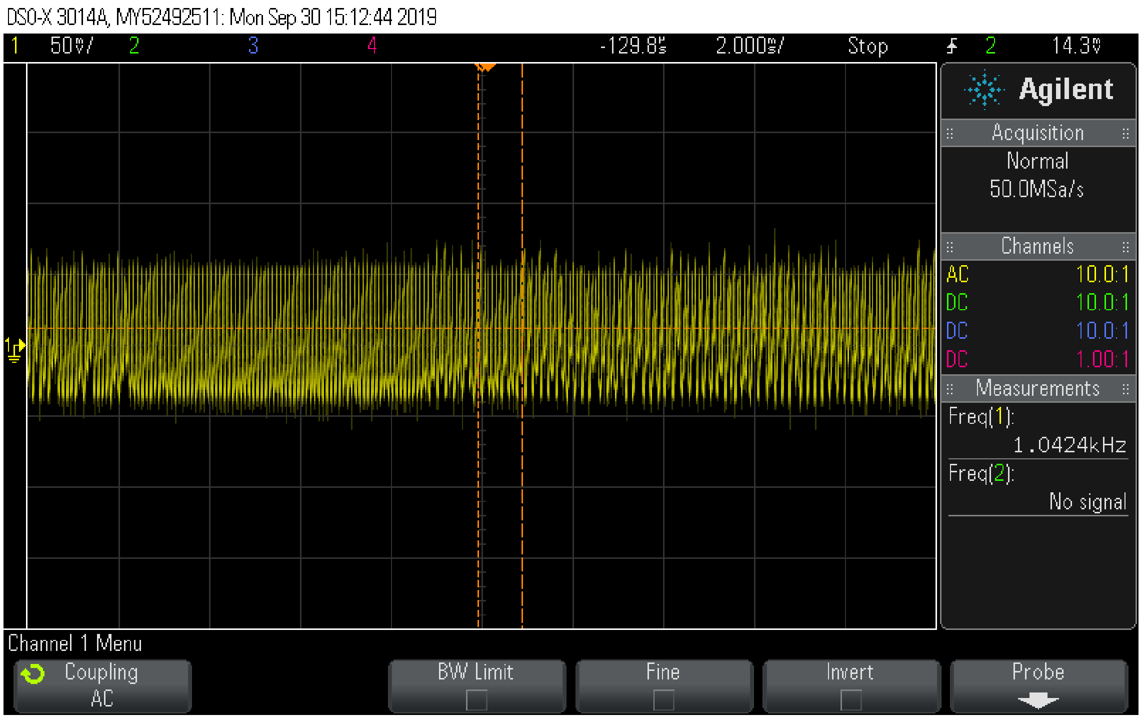


Figure 39. Signal on output of LM2596S at 557mA, after 2 minutes of operation.

As Figure 39 shows, the noise level is again higher than for the LDO, namely around 120mV.

Intermediate conclusion I

Although being suitable for driving the motors at 6V and with their current demand during regular operation with an insignificantly low value of saved input power, the buck converter can most likely not be used for driving the motors when they are stalled, since the output voltage on the buck converter drops and the device heats up.

Experiment 5: LM2596S vs. 29151-5.0; logic supply (5V, 0.25A)

Another option of using a buck converter instead of the LDO is for the 5V rail which drives the Arduino and the Raspberry Pi Zero. The current consumption at this rail has been found to be around 250mA at peak.

The results of the performance test on the existing voltage regulator 29151-5.0 are shown in the following.

Table 22. Performance data for voltage regulator 29151-5.0.

t [s]	I_{out} [A] Set	I_{B} [A] Observed	U_{S} [V] Observed	T [°C] Observed	Image
0	0.2516	0.268	4.482		
60	0.2516	0.268	4.483	28.0	Figure 40
120	0.2516	0.268	4.482		
180	0.2516	0.268	4.482	28.6	
240	0.2516	0.268	4.482		

Although the voltage regulator was used successfully on the original µ-CAT, it can be seen from Table 22 that it does not provide the desired voltage of 5V under the given input and load conditions.

The input power using the LDO is:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 0.268\text{A} = 1.98\text{W} \tag{31}$$

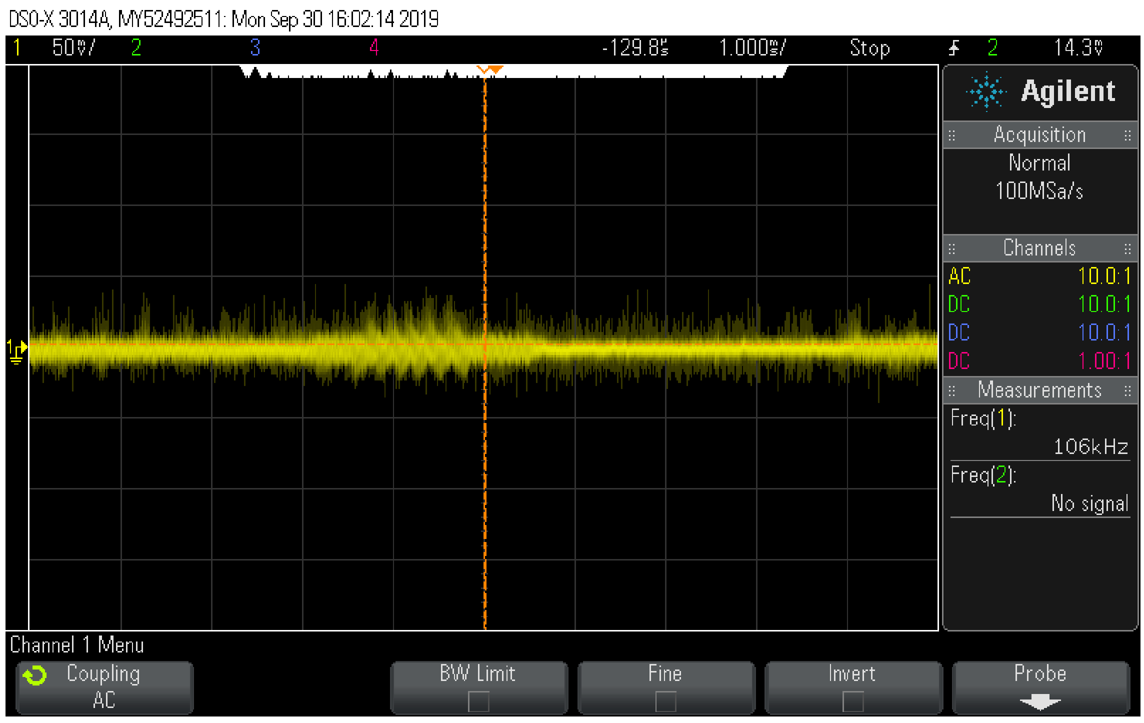


Figure 40. Signal on output of 29151-5.0 at 252mA, after 1 minute of operation.

Figure 40 shows that the noise level is around 30mV, with 60mV peaks.

In comparison, the performance results of the buck converter (which has been adjusted to 5V output) are given in the following:

Table 23. Performance data for buck converter LM2596S.

t [s]	I_{out} [A] Set	I_{B} [A] Observed	U_{S} [V] Observed	T [°C] Observed	Image
0	0.2516	0.228	5.022	29.0	
60	0.2516	0.228	5.022		Figure 41
120	0.2516	0.228	5.024	29.7	
240	0.2516	0.228	5.023		
300	0.2516	0.228	5.022		

According to the results in Table 23, the buck converter can supply stable 5V at the given current and input conditions.

The input power using the buck converter is:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 0.228\text{A} = 1.69\text{W} \qquad (32)$$

This amounts to a power saving of 15% compared to the LDO.

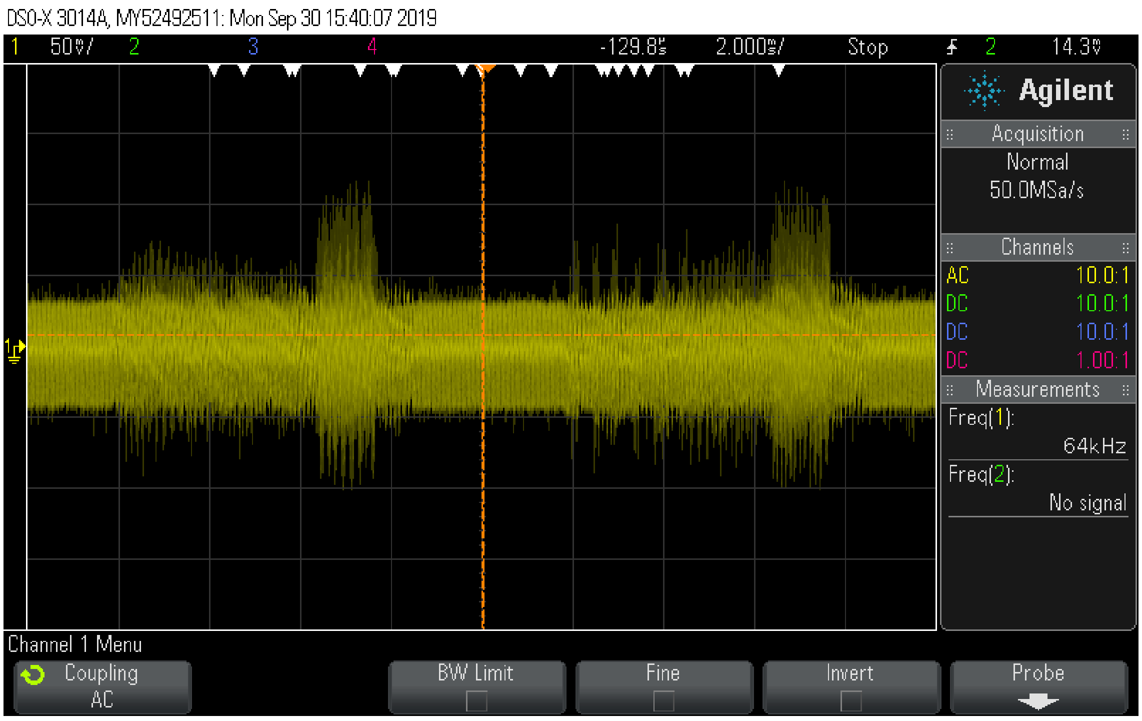


Figure 41. Signal on output of LM2596S at 252mA, after 1 minute of operation.

The buck converter, as seen from Figure 41, creates a much higher noise on the output than the voltage regulator. The magnitude ranges from 90mV to 200mV.

Intermediate conclusion II

With a power saving of around 15% compared to the LDO while maintaining the full output voltage, the buck converter seems to be the better choice. The high output noise might be problematic but can be dealt with by placing capacitors on inputs of critical components, such as sensors and microcontrollers.

Experiment 6: LM2596S vs. 29151-5.0; motors stalled (5V, 2A)

This experimental set and the next one take into consideration the possibility to drive the motors at 5V instead of 6V. Both linear voltage regulator and buck converter were tested under the corresponding conditions.

This experimental set was designed to determine the behaviour of the converters under stall conditions of all four motors, i.e., around 2A of output current.

For the voltage regulator, the experimental results are the following:

Table 24. Performance data for voltage regulator 29151-5.0.

t [s]	I_{out} [A] Set	I_{B} [A] Observed	U_{S} [V] Observed	T [°C] Observed	Image
0	2.004	2.097	4.660	35.0	
60	2.000	2.085	4.692	39.5	Figure 42
120	1.998	2.081	4.693	40.0	
180	1.998	2.077	4.694	40.0	
240	1.998	2.077	4.694		

As Table 24 shows, the voltage regulator does not fully reach the required output voltage.

The input power consumption of the voltage regulator for an average current consumption of 2.0834A is as follows:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 2.0834\text{A} = 15.42\text{W} \tag{33}$$

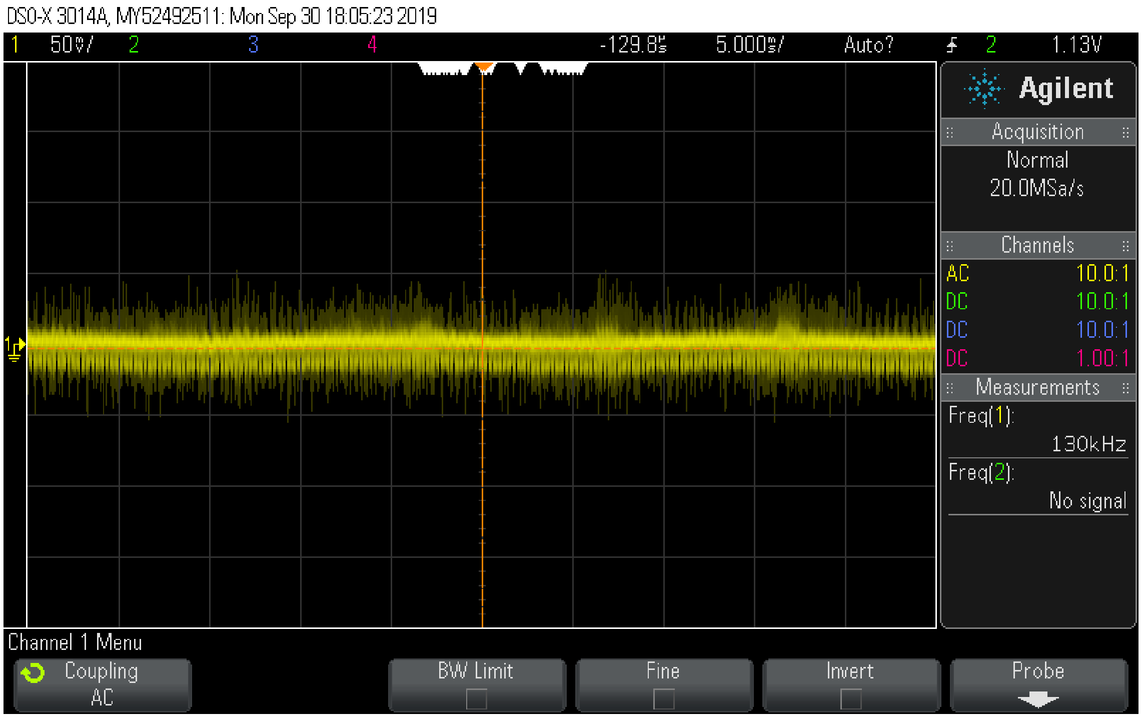


Figure 42. Signal on output of 29151-5.0 at 2A, after 1 minute of operation.

As Figure 42 shows, the noise magnitude is around 30mV and 90mV at peaks.

For the buck converter, the experimental results are shown in the following:

Table 25. Performance data for buck converter LM2596S.

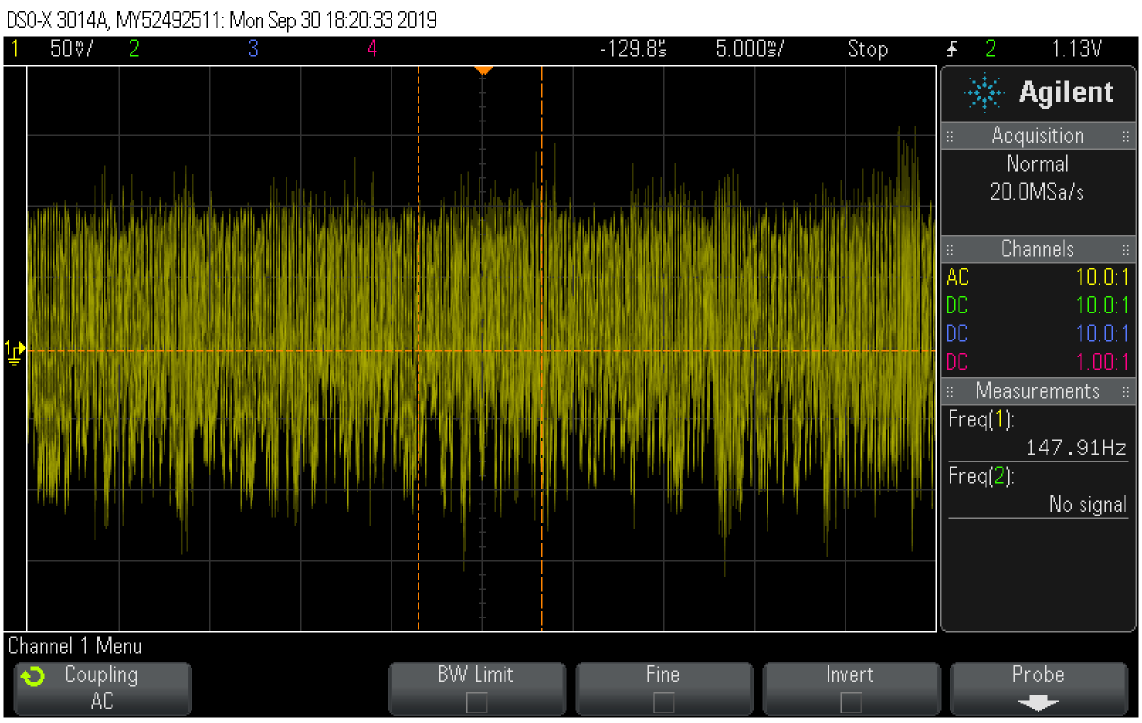
t **[s]**	***I*** I_{out} **[A] Set**	I_B **[A] Observed**	U_S **[V] Observed**	***T*** **[°C] Observed**	**Image**
0	2.004	1.920	5.024	39.0	
60	2.001	1.920	5.043	57.0	Figure 43
120	2.000	1.925	5.046	63.0	
180	1.999	1.924	5.045	72.0	
240	1.999	1.926	5.046	81.0	

The results in Table 25 show that the buck converter reaches and holds the nominal output voltage level under the given conditions, but it produces a significant amount of heat already after 4 minutes of operation, which may be addressed with a sufficiently dimensioned heatsink. This, however, could cause problems in terms of space-efficiency in the PCB layout.

The input power consumption of the buck converter with an average current of 1.923A is:

$$P_{\text{in}} = U_B * I_B = 7.4\text{V} * 1.923\text{A} = 14.23\text{W} \tag{34}$$

This corresponds to a battery power saving of almost 18% compared to the LDO.


DSO-X 3014A, MY52492511: Mon Sep 30 18:20:33 2019

1 50mV/ 2 3 4 -129.8µs 5.000ms/ Stop 2 1.13V

Acquisition: Normal, 20.0MSa/s

Channels: AC 10.0:1; DC 10.0:1; DC 10.0:1; DC 1.00:1

Measurements: Freq(1): 147.91Hz; Freq(2): No signal

Channel 1 Menu: Coupling AC; BW Limit; Fine; Invert; Probe

Figure 43. Signal on output of LM2596S at 2A, after 1 minute of operation.

Figure 43 clearly indicates a very high noise level for this mode of operation, ranging up to 300mV.

Experiment 7: LM2596S vs. 29151-5.0; motors (5V, 0.55A)

This last experimental set was designed to determine the behaviour of the converters under operating conditions of all four motors at 5V, i.e., around 550mA of output current.

For the voltage regulator, the experimental results are the following:

Table 26. Performance data for voltage regulator 29151-5.0.

t [s]	I_{out} [A] Set	I_B [A] Observed	U_S [V] Observed	T [°C] Observed	Image
0	0.558	0.576	4.915	26.6	
60	0.557	0.576	4.916	32.0	Figure 44
120	0.557	0.576	4.915	32.0	
180	0.557	0.576	4.915		
240	0.557	0.576	4.915	32.0	

The voltage regulator, according to the results shown in Table 26, can supply the required voltage at a stable level.

The power consumption on the input is:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 0.576\text{A} = 4.26\text{W} \tag{35}$$

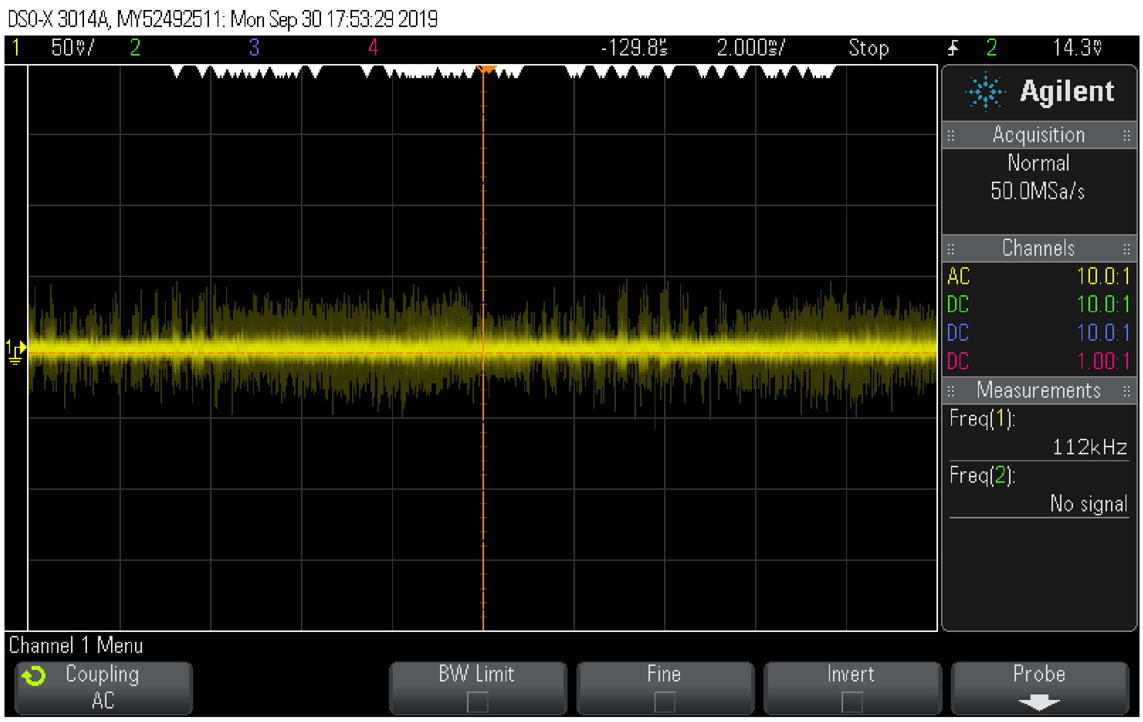


Figure 44. Signal on output of 29151-5.0 at 576mA, after 1 minute of operation.

The noise generated on the output of the regulator is in magnitude of 20mV, with peaks of about 80mV.

For the buck converter, the experimental results are the following:

Table 27. Performance data for buck converter LM2596S.

t **[s]**	$\boldsymbol{I_{\text{out}}}$ **[A] Set**	$\boldsymbol{I_{\text{B}}}$ **[A] Observed**	$\boldsymbol{U_{\text{S}}}$ **[V] Observed**	***T*** **[°C] Observed**	**Image**
0	0.557	0.474	5.024	39.0	
60	0.557	0.474	5.024	38.8	Figure 45
120	0.557	0.474	5.024	40.0	
180	0.557	0.474	5.024	38.4	
240	0.557	0.474	5.023	38.2	

According to the results shown in Table 27, the buck converter can provide the stable output voltage without significant heat dissipation.

The power consumption on the battery is:

$$P_{\text{in}} = U_{\text{B}} * I_{\text{B}} = 7.4\text{V} * 0.474\text{A} = 3.51\text{W} \tag{36}$$

This is roughly 18% less than when using the voltage regulator.

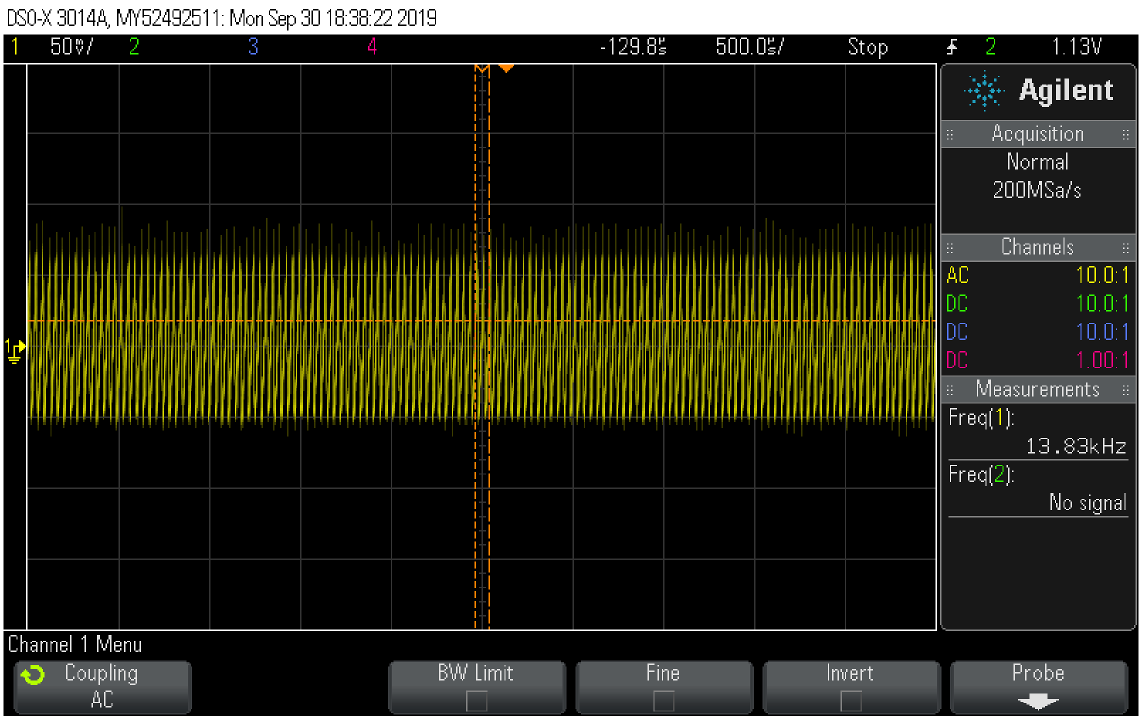


Figure 45. Signal on output of LM2596S at 557mA, after 1 minute of operation.

The noise level in this mode of operation is around 130mV.

Intermediate conclusion III

With a power saving of up to 18% compared to the LDO while maintaining the full output voltage, the buck converter seems to be the better choice for operation of the motors at 5V instead of (nominal) 6V.

However, the effects of the voltage reduction on the motors on the behaviour of the robot are hard to estimate without extensive tests in water, which would exceed the scope of this project. It was therefore decided to keep the motors running at 6V, and as above experiments have shown, the voltage regulator is still the better choice to do that.

A 1.2 Verification of chosen buck converter

As stated in the previous section, there are concerns about the high noise level on the output of the buck converter. It must be ensured that a sensor, with capacitors added between "GND" and "+5V" near its supply terminals, provides stable output when using the buck converter as the power source. This is the purpose of the following experimental set.

The experiments were conducted using the old pressure sensor MS5407-AM [66], supplied with 5V. The pressure sensor is supposed to be one of the most noise-sensitive devices and would most likely show false readings under bad power supply conditions. The sensor was interfaced using the Arduino Mini 05 microcontroller board via the SPI bus. The sensor readings were plotted into the Serial terminal of the Arduino IDE (plotter mode). The microcontroller board was also supplied with 5V. The 5V rail which powers sensor and Arduino took its voltage from one of three sources, in each of the experiments:

1. voltage regulator (LDO) 29151-5.0 (used on old µ-CAT);
2. USB power supply from the laptop;
3. buck converter LM2596S (used on new µ-CAT).

The results indicate that all three voltage sources yield stable readings of the pressure sensor. Had the supply voltage been too shaky or noisy, unstable sensor output values could have been expected.

The success of this test verifies that the chosen buck converter is adequate for use on the 5V rail.

A 1.3 Power consumptions

Power consumptions using the PCB of old and new µ-CAT were compared. The experimental results allow for conclusions on:

- added power consumption due to newly integrated devices;
- power consumption differences due to the use of the new power supply (buck converter on 5V rail);
- power consumption differences due to any other reasons, such as unstable supply voltages, heating up of PCB copper traces etc (verification of the new PCB).

The experiments were conducted using power supply Instek PSP-405 at a set input voltage of $U_{\mathrm{B}} = 7.4\mathrm{V}$. Resulting input current I_{B} was recorded and input power consumption P_{in} was calculated:

$$P_{\mathrm{in}} = U_{\mathrm{B}} * I_{\mathrm{B}} \tag{37}$$

In experiments where the camera or the motors are used, the resulting current varies, and therefore at least 24 repeated measurements were conducted. The power was then calculated using the average value of the measured currents.

As Table 28 shows, a total of 12 experiments was conducted. The first six were carried out on the new PCB, the last six on the old PCB. In each experiment, another combination of devices was used. For a more direct comparison between the PCBs, the Raspberry Pi was additionally powered with the old PCB during experiments 8 and 10, using the onboard 5V regulator 29151-5.0.

The results of the following experimental sets are directly comparable, because they employ the same devices:

- 1 and 7;
- 4 and 10;
- 5 and 11;
- 6 and 12.

Table 29 lists the obtained results of the experiments.

Table 28. Experimental sets, PCB under test and involved devices. Colour-codes indicate which sets are directly comparable.

	New PCB						Old PCB					
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Tone detectors powered	●	●	●	●	●	●	●	●	●	●	●	●
Motors running	●	●	●				●	●	●			
Bluetooth module powered		●	●									
Messaging LEDs powered		●	●									
Raspberry Pi powered		●	●	●		●		●	●	●		●
Camera powered		●	●	●		●		●	●	●		●
Camera in use			●	●					●	●		

Note that when the Raspberry Pi is powered, its WLAN connection is also established. This means that the measured Raspberry Pi's power consumption always includes WLAN.

Table 29. Power consumption experimental results and calculations.

Experimental set	**Input current I_B [A]**	**Input power P_{in} [W]**
1	0.352	2.605
2	0.606	4.487
3	0.766	5.668
4	0.354	2.617
5	0.089	0.659
6	0.188	1.391
7	0.464	3.434
8	0.565	4.185
9	0.789	5.839
10	0.416	3.080
11	0.112	0.829
12	0.330	2.400

Current consumption of the Bluetooth module was measured separately to be 45mA, and it amounts to 0.333W.

Current consumption of the messaging LEDs was measured separately to be 120mA, and it amounts to 0.888W.

Current consumption of the tone detectors was measured separately to be 15mA, and it amounts to 0.111W.

A 2 Components and values

This appendix section lists some electronic parts used on the new PCB. Wherever applicable, additional information on electrical characteristics is given.

A 2.1 LEDs

Resistor values and power consumption are calculated according to the equations shown in Section 4.3. Chosen resistor values according to the availability on Farnell[1].

Table 30. RGB status LED parameters.

Eagle name	**Colour**	**Supply voltage U_{CC} [V]**	**Forward voltage U_{LED} [V]**	**Real forward current I [mA]**	**Calculated resistor value R [Ω]**	**Chosen resistor value R [Ω]**	**Power consumption P [mW]**
LED5-G	green	3.3	3.2	10.0	10	10	33.0
LED5-R	red	3.3	2.0	10.0	130	130	33.0
LED5-B	blue	3.3	3.2	10.0	10	10	33.0

[1] Online: https://ee.farnell.com/

Table 31. Single-colour status LEDs parameters.

Eagle name	**Colour**	**Supply voltage U_{CC} [V]**	**Forward voltage U_{LED} [V]**	**Real forward current I [mA]**	**Calculated resistor value R [Ω]**	**Chosen resistor value R [Ω]**	**Power consumption P [mW]**
LED1	green	3.3	2.1	1.99	600	604	6.5
LED2	yellow	5.0	2.1	2.03	1450	1430	10.1
LED9	yellow	5.0	2.1	2.03	1450	1430	10.1
LED3	yellow	5.0	2.1	2.03	1450	1430	10.1
LED4	red	5.0	2.0	2.00	1500	1500	10.0
LED6	green	5.0	2.1	2.03	1450	1430	10.1
LED8	green	6.0	2.1	1.99	1950	1960	11.9
LED10	blue	3.3	2.65	2.01	325	324	6.6
LED11	blue	3.3	2.65	2.01	325	324	6.6

Table 32. Messaging LEDs parameters.

Eagle name	**Colour**	**Supply voltage U_{CC} [V]**	**Forward voltage U_{LED} [V]**	**Real forward current I [A]**	**Calculated resistor value R [Ω]**	**Chosen resistor value R [Ω]**	**Power consumption P [W]**
	white	6.0	5.5	0.128	3.57	3.90	0.77

A 3 Mechanics, solid modelling and manufacturing

A 3.1 Parts names and descriptions

Table 33. Parts codes, file names and descriptions.

Code	Name	Description
01-01	PLEXIGLASS_HULL	cylindrical Plexiglass tube
01-02	BATTERY_HOLDER	cylindrical tube containing the batteries
01-03	THREADED_ROD_WEIGHT_DISTRIBUTOR	M5 rod holding the weights
01-04	Weights_Divider	Plexiglass supports for PCB
01-05	PCB_raw	rough model of new PCB
01-06	THREADED_ROD_OUTSIDE	M3 rods closing the end caps
01-07	Battery	battery cell model
01-08	Washer_Outside	DIN 433; washer on M3 rods
01-09	HexNut_Outside	ISO 4035; nut on M3 rods
01-10	HexNut_WeightDistributor	ISO 4035; nut on M5 rods
02,03-01	SERVO_FS70MG	servo motor
02-02	Hull_Front	front end cap
02,03-03	MOTOR_SHAFT_CLUTCH	clutch on servo motor
02,03-04	SHAFT_SLEEVE	sleeve on fin shaft
02,03-05	Oil-less bush 8mm	bushing on fin shaft
02,03-06	Nitrile oil seal_5x16x6mm	seal on fin shaft
02-07	Camera	Raspberry Pi camera
02-08	Dome_Front	front dome
02-11	Camera-Cap_Body	main body of camera cap
02-13	Camera-Seal-Glass	seal under glass of camera cap
02-14	Camera-Glass	glass of camera cap
02-15	LED-PCB_raw	rough model of LED PCB
02-16	Pressuresensor_Hose-Connector	connecting tube for pressure sensor hose
02,03-17	Fin-Shaft	shaft of fin
02,03-18	Fin	fin
02,03-19	Oring_Hull	sealing ring of cyclindrical Plexiglass tube
02-20	Photodiode-Plug_raw	rough model of seal around photodiode
02,03-22	Fin-Shaft-Tube	tube surrounding shaft of fin
02,03-23	Weights-box	box for additional weights
03-01-01	USB-MainBody	main body of USB insert
03-01-03	USB-oring-main	sealing ring of USB insert

Table 33. Parts codes, file names and descriptions [continued].

Code	**Name**	**Description**
03-01-02	USB-Nut	nut on USB insert
03-01-04	USB-Cap	cap of USB insert
03-01-05	USB-Cap-Seal	seal under cap of USB insert
03-02	Hull_Back	back end cap
03-07	Bluetooth_raw	rough model of Bluetooth PCB
03-08	Dome_Back	back dome
03-09	BatteryHolder-Reducer_raw	rough model of reducer holding battery cap
03-20-01	BatteryCap_Body	body of battery cap
03-20-02	BatteryCap_Seal	seal under battery cap
03-20-03	BatteryCap_Spring	spring connecting to batteries
03-24	TR Fastenings Ltd-M5	ISO 4032; M5 nut holding reed switch
03-25	ReedSwitch	reed switch

Table 34 lists all SolidWorks design files and indicates the following properties for each:

- to which assembly the assembly or part belongs (colour-coding);
- the number of instances used;
- the code used in the file name;
- the file type (assembly or part);
- whether it is part of the physical or mechanical design (not an element related to electronics);
- whether it has been present on the original µ-CAT robot (in some form);
    - if so, whether its design file has been taken over from the previous design as-is;
    - if so, whether its design file has been updated;
- whether the part is completely newly created;
- in case it is updated or newly created, whether it is created by the author of this thesis or taken from an external source;
- how it is produced:
    - bought and either taken as-is or manually modified after purchase;
    - custom-made by one or more methods, as indicated.

Parts names and descriptions are given in Appendix A 3.1.

Table 34. Overview of SolidWorks assemblies and parts. A: assembly; P: part; ■: source: [67]; ▲: source: [68]; ▼: modified part from source: [69]; *: alternatively, the Bluetooth module HC-05 can be used after some manual rework.

Quantity	Code	Type	Is part of physical/mechanical design	Is used on old μ-CAT robot	SolidWorks design					Manufacturing method(s)						
					From existing μ-CAT		New	Source (of new part or update)		Off-the-shelf		Custom-made				
					As-is	Updated		Self	External	As-is	With manual rework	3D-printing	Laser-cutting	Lathe-turning	Silicone-casting	Other
1	0	A														
1	01	A														
1	01-01	P	●	●	●						●					
1	01-02	P		●		●		●			●					
2	01-03	P	●	●	●						●					
2	01-04	P	●	●			●	●					●			
1	01-05	P					●	●								●
2	01-06	P	●	●		●		●			●					
2	01-07	P		●			●	●		●						
8	01-08	P	●	●			●	●		●						
12	01-09	P	●	●			●	●		●						
20	01-10	P	●	●			●	●		●						
1	02	A														
1	03	A														
4	02,03-01	P		●		●		●			●					
1	02-02	P	●	●			●	●				●				
4	02,03-03	P		●	●						●					
4	02,03-04	P		●	●									●		
4	02,03-05	P		●	●					●						
4	02,03-06	P		●	●					●						
1	02-07	P					●		■	●						
1	02-08	P	●	●			●	●				●				
1	02-11	P	●				●	●				●				
1	02-13	P	●				●	●							●	
1	02-14	P	●				●	●					●			

Table 5. Overview of SolidWorks assemblies and parts [continued].

Quantity	Code	Type	Is part of physical/mechanical design	Is used on old μ-CAT robot	SolidWorks design					Manufacturing method(s)						
					From existing μ-CAT			Source (of new part or update)		Off-the-shelf		Custom-made				
					As-is	Updated	New	Self	External	As-is	With manual rework	3D-printing	Laser-cutting	Lathe-turning	Silicone-casting	Other
1	02-15	P					●	●								●
1	02-16	P		●			●	●			●					
4	02,03-17	P	●	●	●						●					
4	02,03-18	P	●	●	●										●	
2	02,03-19	P	●	●			●	●		●						
2	02-20	P		●			●	●							●	
4	02,03-22	P	●	●			●	●			●					
2	02,03-23	P	●				●	●				●				
1	03-01	A														
1	03-01-01	P		●			●	●		●						
1	03-01-02	P		●			●	●		●						
1	03-01-03	P		●			●	●		●						
1	03-01-04	P		●			●	●		●						
1	03-01-05	P		●			●	●		●						
1	03-02	P	●				●	●				●				
1	03-07*	P					●	●								●
1	03-08	P	●	●			●	●				●				
1	03-09	P		●			●	●		●						
1	03-20	A														
1	03-20-01	P		●			●	●		●						
1	03-20-02	P		●			●	●		●						
1	03-20-03	P		●			●	●		●						
1	03-24	P	●	●			●		▲	●						
1	03-25	P		●			●		▼	●						

A 3.2 Manufacturing

A 3.2.1 Equipment and procedure for SLA 3D-printing

Equipment and procedure are explained in the following.

Equipment:

▪ 3D printer:	formlabs Form 3 (early 2019 model)
▪ Cleaning:	formlabs bathing station with isopropyl alcohol
▪ Curing:	formlabs curing station
▪ Printing material:	formlabs Durable resin
▪ Software:	formlabs PreForm 3.4.2 and newer

Procedure:

4. Export an STL file from SolidWorks.
5. Arrange part(s) in formlabs PreForm software, using rafts and 0.5mm touch points.
6. 3D-print part(s).
7. Clean parts in alcohol bath for 20 minutes.
8. Harden part(s) in curing station at 60°C for one hour.
9. Let part(s) cool down.
10. Remove rafts.

A 3.2.2 Equipment and procedure for FDM 3D-printing

The front and back dome are printed using the other in-lab printer. Equipment and procedure are described in the following.

Equipment:

▪ 3D printer:	3D Systems BFB Touch 3D
▪ Printing material:	PLA (yellow)
▪ Software:	Axon 2

Procedure:

1. Export an STL file from SolidWorks.

2. In Axon 2, arrange it on the printing platform and choose settings: material, density (quality) and raft support options.
3. Generate the printer file (G code) and save it to flash drive.
4. Insert flash drive into printer and start the print.
5. After the print is finished, remove the object from the platform and remove rafts (if applicable).

A 3.2.3 Procedure for silicone-casting

Seals around photodiodes and the gasket inside the camera cap are silicone-cast. The procedure is described in the following list. Note that mixing time and ratio depend on the type of silicone used.

1. Design mould in SolidWorks.
2. 3D-print mould.
3. Cover the mould with “Ease Release” agent[1] (optional).
4. Mix two silicone components in 1:1 ratio.
5. Pour a thin stream into mould from distance, to minimize formation of bubbles.
6. Knock mould on flat surface to get bubbles out.
7. Cover with film and place weights on it to achieve a flat smooth surface.
8. Let it cure for approx. 30 minutes (depends on size).
9. Remove weights, film and pull silicone out of the mould.
10. Cut off flashes.

A 3.3 Assembling

A high-level exploded view of the robot is shown in Figure 46. The numbers indicate the assemblies or parts and refer to the order of steps when assembling the robot:

1. Slide the PCB over the core assembly and into the provided grooves of the back end cap.

[1] “Ease Release” is a wax-based liquid release agent and can be used to facilitate the removal of the silicone body from the mould.

2. Route the wires of components installed in the back end cap assembly onto the PCB and connect them.
3. Slide the Plexiglass cylinder over the PCB and over the back end cap.
4. Bring the front end cap assembly close to the Plexiglass cylinder, route all wires of the installed components to the PC and connect them, then slide the front end cap into the Plexiglass cylinder. Close the hex nuts on the outside rods.
5. Place front and end dome over the front and end cap, respectively. Each dome requires two bolts to be fastened.

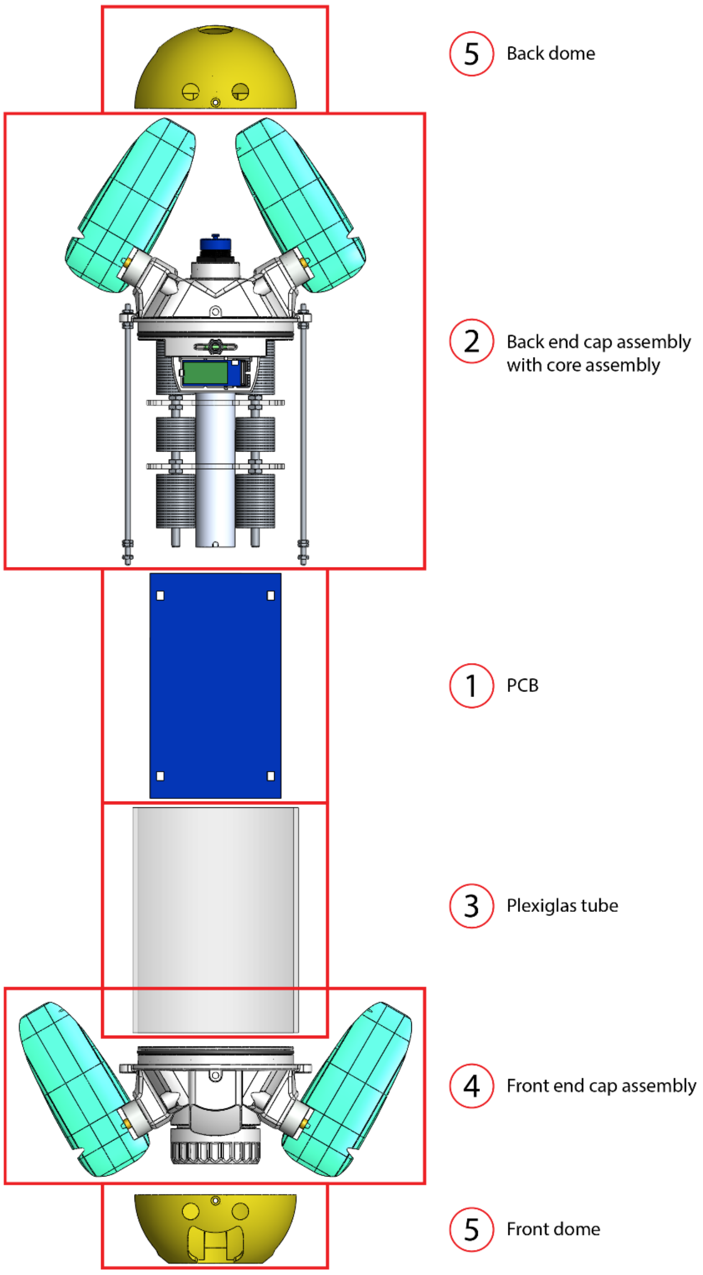


Figure 46. Assembling the robot.

A 4 Software

A 4.1 Arduino interfacing

Table 35. Pins of ATmega328P MLF32. ▲: Restricted to denoted type.

Physical pin number (MLF32)	**Arduino pin ID**	**Signal name**	**Type**
1	3	FSK_TX	Digital Output
2	4	RIGHTBEACON	Digital Input
3		GND	Power
4		+5V	Power
5		GND	Power
6		+5V	Power
7			Crystal
8			Crystal
9	5	LEFTBEACON	Digital Input
10	6	FSK_RX	Digital Input
11	7	CS1	Digital Output
12	8	CS4	Digital Output
13	9	CS3	Digital Output
14	10	CS2	Digital Output
15	11	SPI_MOSI	Digital Output
16	12	SPI_MISO	Digital Input
17	13	SPI_SCK_MEGA	Digital Output
18		+5V	Power
19	A6	BATTERY_MEASURE	Analogue Input▲
20		AREF	Power
21		GND	Power
22	A7	ADC7 (unused)	Analogue Input▲
23	A0	BATTERY_LOW_LED	Digital Output
24	A1	FSK_ENABLE	Digital Output
25	A2	ARD2RPI_HV	Digital Output
26	A3	MOTOR_EN	Digital Output
27	A4	I2C_SDA_HV	Bus
28	A5	I2C_SCL_HV	Bus
29		ARDU_RST	Reset
30	0	ARDU_RX	Digital Input
31	1	ARDU_TX	Digital Output
32	2	IMUINT_HV	Digital Input

Table 35 shows the Arduino microcontroller’s pins, along with their Arduino identifier (if any), the corresponding signal name and their usage on µ-CAT.

A 4.2 Raspberry Pi interfacing

Table 36. Pins in use on Raspberry Pi Zero W.[1]

Physical pin number (PCB header)	**BCM pin ID**	**WiringPi[2] pin ID**	**Signal name**	**Type**
1			RPI_3V3	Power
2			+5V	Power
3	2	8	RPI_SDA	Bus
4			+5V	Power
5	3	9	RPI_SCL	Bus
6			GND	Power
7	4	7	ARD2RPI_LV	Digital Input
8	14	15	RPI_TX	Digital Output
9			GND	Power
10	15	16	RPI_RX	Digital Input
11	17	0	ARDU_RST_LV	Digital Output
12	18	1	LED_R	PWM Output
13	27	2	BT_CMD	Digital Output
14			GND	Power
20			GND	Power
25			GND	Power
30			GND	Power
31	6	22	MUX_EN	Digital Output
32	12	26	LED_G	PWM Output
33	13	23	LED_B	PWM Output
34			GND	Power
35	19	24	MUX_A	Digital Output
37	26	25	MUX_B	Digital Output
38	20	28	BT_RST	Digital Output
39			GND	Power
40	21	29	BT_STATE	Digital Input

Table 36 shows the Raspberry Pi Zero W PCB header pins, along with their identifiers (if any), the corresponding signal name and their usage on µ-CAT.

[1] https://pinout.xyz/

[2] http://wiringpi.com/

A 4.3 Multiplexer interfacing

Table 37. Multiplexer routing table.

Address bit B	Address bit A	X (Arduino RX) routed to	Y (Arduino TX) routed to	Device
0	0	X0	Y0	Raspberry Pi
0	1	X1	Y1	Bluetooth
1	0	X2	Y2	FTDI
1	1	X3	Y3	(unused)

A 4.4 Test suite

A 4.4.1 Relevant files

Accompanying Section 5.2, the following codes are a reference for the test engineer to compile required files.

The "Makefile" presented in Figure 47 can be used as shown here. It includes the paths to Arduino libraries and the tag for the target architecture to compile for.

```
ARDUINO_DIR = /usr/share/arduino
ARDMK_DIR = /usr/share/arduino

ARDUINO_LIBS = SPI Wire I2Cdev MPU6050 EEPROM i2cdetect
USER_LIB_PATH = /home/pi/Arduino/libraries
BOARD_TAG = mini328
MONITOR_PORT = /dev/ttyS0

include /usr/share/arduino/Arduino.mk
```

Figure 47. Content of "Makefile".

The text presented in Figure 48 is part of the content of the file "testcases.csv" which is used by the main user interface to compile the list of available test cases. This file must be manually updated by the programming engineer, and its structure must follow the one shown here.

```
category|code|name|displayname|description|unitsUnderTest|instructions
other|o01|RGB-LED|RGB-LED|Testing RGB LED|RPi PWM, RGB LED|Guided test.
```

Figure 48. Headline and example data set in the file "testcases.csv".

Figure 49 shows – as an example – the content of the parameters file for one test case. The programming engineer must follow the structure used in the existing parameter files when creating a new parameter file for another test case. Possible options for the data type are (so far):

`int`	integer
`flt`	float
`hex`	integer in hexadecimal representation
`str`	string

```
canUseDefault,paramName,paramValue,paramType,paramUnit,inInofile
1,R_H,47000,int,Ohms,1
1,R_L,5600,int,Ohms,1
1,Analog reference,1.1,flt,V,1
```

Figure 49. Content of file “params.csv” for test case “a04”.

A 4.4.2 Main Python code

```
#!/usr/bin/python3

#########################################
## 30.04.2020
## Kilian Ochs
## Test suite v1.0
##
## A user-interactive script which manages and runs test cases.
## Test cases are specified in the csv file "testcases.csv", as specified
## by the variable "path_to_file_testcases" below.
##
## This code runs on the RaspberryPi.
##


import csv, os, time, subprocess
from datetime import datetime

## globally used variables:

  ## parameters:
path_to_dir_resources = "./resources"
path_to_file_testcases = path_to_dir_resources+"/testcases.csv"
name_of_file_params = "params.csv"
name_of_file_report = "report.txt"

  ## internal:
testCases = []
numTestCases = 0

## example csv line:
#  arduino,a01,UART_RPi,UART 1,Bidirectional test for UART interface between Arduino and
RaspberryPi

class TestCase:
  category = "" # arduino or other
  code = ""     # e.g., a01
  name = ""     # e.g., UART_RPi
  displayname = "" # e.g., UART 1
  description = "" # e.g., Bidirectional test for UART interface between Arduino and RaspberryPi
  instructions = ""
  devices = ""
  path = "" # path to test case directory
  num = 0   # global list number
  params = []  # parameters for this test case; each row will hold: canUseDefault, paramName,
paramValue, paramType (as strings)
  execTime = None

  def
__init__(self,_num,_category,_code,_name,_displayname,_description,_devices,_instructions):
    global path_to_dir_resources
    self.num = _num
    self.category = _category
    self.code = _code
    self.name = _name
    self.displayname = _displayname
    self.description = _description
    self.devices = _devices
    self.instructions = _instructions
    self.path = path_to_dir_resources + \
      "/" + self.category + \
      "/" + self.code + "_" + self.name + "/"

  def toString(self):
    print("%2d  |  Category        : %s" % (self.num,self.category))
    print("    |  Name            : "+self.displayname)
    print("    |  Description     : "+self.description)
    print("    |  Units involved  : "+self.devices)
    print("-----------------------------------------------------------------------------------------
-------------------------------------------------------------------------")


''' Reads csv file and prints the list to the terminal '''
def setup():
  print("Setting up test cases.")
  global path_to_file_testcases
  global numTestCases
  # Read available test cases into objects:
  with open(path_to_file_testcases, 'r') as csvfile:
    reader = csv.reader(csvfile, delimiter = '|')
    rowNum = 0
    for row in reader:
      if rowNum > 0:

testCases.append(TestCase(numTestCases+1,row[0],row[1],row[2],row[3],row[4],row[5],row[6]))
        numTestCases += 1
      rowNum += 1
  print("Parsed "+str(numTestCases)+ " test cases.")

''' Prints list of test cases imported from csv '''
def printTestCases():
  global testCases
  print("\n---------------------------------------------------------------------------------------
-------------------------------------------------------------------------")
  for tc in testCases:
    tc.toString()
  print("\n")

''' User-interactive menu '''
def loop():
  global numTestCases
  global path_to_dir_resources
  global name_of_file_params
  global testCases
  while True:
    # Present the list of test cases:
    printTestCases()
    # Wait for user input:
    try:
      instr = input("Open a test case by entering its integer.\n'0' to exit. ")
    except EOFError as error:
      print("\nCaught EOF error")
    except Exception as exception:
      print("\nCaught Exception")
    # Parse user input:
    choice = int(instr)
    if choice == 0:
      return
    if choice > numTestCases:
      print("\nUser input fault: No such test case!")
    else:

      # Display instructions:
      displayInstructions(choice)

      # Ask user if he really wants to run this test case:
      try:
        inStr = input("Type 'y' to start this test case, otherwise 'n'. ")
      except EOFError as error:
        print("\nCaught EOF error")
      except Exception as exception:
        print("\nCaught Exception")

      if inStr == "y":
        printHeadline(choice)

        # Check if there are parameters to be defined:
        path_to_file_params = path_to_dir_resources+\
          "/"+testCases[choice-1].category+\
          "/"+testCases[choice-1].code+"_"+testCases[choice-1].name+\
          "/"+name_of_file_params
        if os.path.isfile(path_to_file_params):
          testCases[choice-1].params = []
          try:
            with open(path_to_file_params, 'r', newline='\n') as csvfile:
              reader = csv.reader(csvfile, delimiter = ',')
              rowNum = 0
              numParams = 0
              for row in reader:
                if rowNum > 0:
                  testCases[choice-1].params.append([row[0],row[1],row[2],row[3],row[4],row[5]])
                  numParams += 1
                rowNum += 1
            print("\nParsed "+str(numParams)+ " parameters.")
            print("List of parameters:\n")
            for param in testCases[choice-1].params:
              print(param[1]+": "+param[2]+param[4])
              numParams += 1

            # If parameters are mandatory to be defined by user, ask for them and save in test
case object:
```

```
            allMandatory = True
            for param in testCases[choice-1].params:
              if param[0] == 0:
                try:
                  paramStr = input("\nSpecify "+param[1]+" value: ")
                except EOFError as error:
                  print("\nCaught EOF error")
                except Exception as exception:
                  print("\nCaught Exception")
                param[2] = paramStr
              else:
                allMandatory = False

            if not allMandatory:
            # If not, ask if user wants to define parameters:
              try:
                inStr = input("\nDo you want to change parameters with already defined values?
['y' or 'n']: ")
              except EOFError as error:
                print("\nCaught EOF error")
              except Exception as exception:
                print("\nCaught Exception")
              if inStr == "y":
              # If yes, present list of parameters and let user choose and input values:

                while True:
                  print("\nList of parameters:\n")
                  numParams = 0
                  for param in testCases[choice-1].params:
                    print(str(numParams+1)+"  "+param[1]+"  Current value: "+param[2]+param[4]+"
(Type "+param[3]+")")
                    numParams += 1
                  try:
                    inStr = input("\nSelect parameter integer or '0' to execute test case with
chosen values: ")
                  except EOFError as error:
                    print("\nCaught EOF error")
                  except Exception as exception:
                    print("\nCaught Exception")
                  inVal = int(inStr)
                  if inVal == 0:
                    print("User aborted.")
                    break
                  if inVal > numParams:
                    print("\nUser input fault: No such parameter!")
                  else:
                    try:
                      paramStr = input("Enter new parameter value: ")
                    except EOFError as error:
                      print("\nCaught EOF error")
                    except Exception as exception:
                      print("\nCaught Exception")
                    testCases[choice-1].params[inVal-1][2] = paramStr

            # Write values to params.csv file:
            numParams = 0
            print("\nThe following parameter values will be used:\n")
            for param in testCases[choice-1].params:
              print(param[1]+": "+param[2]+param[4])
            with open(path_to_file_params, 'w') as csvfile:
              writer = csv.writer(csvfile,delimiter = ',')
              row0 = ["canUseDefault", "paramName", "paramValue", "paramType", "paramUnit",
"inInofile"]
              writer.writerow(row0)
              for row in testCases[choice-1].params:
                writer.writerow(row)
                numParams += 1

            # Write new parameters into ino file:
            if testCases[choice-1].category == "arduino":
              #print("\nConfiguring ino file.")
              inoContent = ""
              origInoContent = ""
              paramBaseName = "param"

              try:
                with open(testCases[choice-1].path+testCases[choice-1].code+".ino",'r') as
inofile:
                  inoContent = inofile.read()
                  origInoContent = inoContent
              except IOError:
                print("\nError: Cannot access ino file for reading.")
                return -1
              startPos = 0
              for i in range(numParams):
                #print(testCases[choice-1].params[i][1]+" in ino: "+testCases[choice-
1].params[i][5])
                if testCases[choice-1].params[i][5] == '1': # replace only if this parameter is
present in ino file
                  startPos = inoContent.find(paramBaseName+str(i+1),startPos)
                  if startPos < 0:
                    print("\nError in ino file: Parameter \""+paramBaseName+str(i+1)+"\" not
found!")
                    return -1
                  startPos = startPos + len(paramBaseName)+len(str(i+1))+1
                  paramVal = inoContent[startPos:inoContent.find("\n",startPos)]
                  paramLen = len(paramVal)
                  newParamVal = testCases[choice-1].params[i][2]
                  if testCases[choice-1].params[i][3] == "str":
                    newParamVal = "\""+newParamVal+"\""
                  inoContent =
inoContent[:startPos]+newParamVal+inoContent[(startPos+paramLen):]

              if origInoContent != inoContent:
                try:
                  with open(testCases[choice-1].path+testCases[choice-1].code+".ino",'w') as
inofile:
                    #print("Writing parameter values to ino file.")
                    inofile.write(inoContent)
                except IOError:
                  print("\nError: Cannot access ino file for writing.")
                  return -1

          except IOError:
            print("\nError: Cannot access parameters file.")
            return -1

        else:
          print("No parameters.")

        # Create timestamp:
        testCases[choice-1].execTime = datetime.now()
        dt_string = testCases[choice-1].execTime.strftime("%d/%m/%Y %H:%M:%S")
        # Write timestamp into ino file:
        if testCases[choice-1].category == "arduino":
          inoContent = ""
          try:
            with open(testCases[choice-1].path+testCases[choice-1].code+".ino",'r') as inofile:
              inoContent = inofile.read()
          except IOError:
            print("\nError: Cannot access ino file for reading.")
            return -1

          startPos = inoContent.find("timestamp",0)+len("timestamp")+1
          paramVal = inoContent[startPos:inoContent.find("\n",startPos)]
          paramLen = len(paramVal)
          inoContent = 
inoContent[:startPos]+"\""+dt_string+"\""+inoContent[(startPos+paramLen):]
          try:
            with open(testCases[choice-1].path+testCases[choice-1].code+".ino",'w') as inofile:
              #print("Writing timestamp to ino file.")
              inofile.write(inoContent)
          except IOError:
            print("\nError: Cannot access ino file for writing.")
            return -1

        # Execute the chosen test case:
        print("\nTest case executes in 5 seconds.\n");
        time.sleep(5)
        resourcesFolderName = path_to_dir_resources[2:]
        '''
        process = subprocess.Popen(['python3',"./resources/arduino/a01_UART_RPi/a01.py", "a01", 
"UART_RPi", "date", "time", "params.csv"],shell=False)
        code = process.wait()
        print("Process exited with code: "+str(code))
        '''

        os.system("cd "+resourcesFolderName+\
          "&& cd "+testCases[choice-1].category+\
          "&& cd "+testCases[choice-1].code+"_"+testCases[choice-1].name+\
          "&& python3 "+testCases[choice-1].code+".py "+\
          testCases[choice-1].code+" "+\
          testCases[choice-1].name+" "+\
          dt_string+" "+\
          name_of_file_params)

        print("\n")

      else:
        print("User aborted.")

def displayInstructions(choice):
  global testCases
  print("\nUser instructions:")
  print("------------------")
  currentParagraph = ""
  contentLen = len(testCases[choice-1].instructions)
  iterator = iter(range(0,contentLen))
  for i in iterator:
    c = testCases[choice-1].instructions[i]
    if c == '\\':
      if i+1 < contentLen:
        if testCases[choice-1].instructions[i+1] == "n":
          print(currentParagraph)
          currentParagraph = ""
          next(iterator, None)
    else:
      currentParagraph += c
  print(currentParagraph)
  print("\n")

def printHeadline(choice):
  global testCases
  print("")
  print("**********************")
  print("*                    *")
  print("*",end='')
  lenOfCode = len(testCases[choice-1].code)
  lenOfName = len(testCases[choice-1].displayname)
  spaceCode = 20 - lenOfCode
  spaceCodeLeft = int(spaceCode / 2)
  spaceCodeRight = int(spaceCodeLeft + spaceCode % 2)
  spaceName = 20 - lenOfName
  spaceNameLeft = int(spaceName / 2)
  spaceNameRight = int(spaceNameLeft + spaceName % 2)
  for i in range (spaceCodeLeft):
    print(" ",end='')
  print(testCases[choice-1].code, end='')
  for i in range (spaceCodeRight):
    print(" ",end='')
  print("*")
  print("*",end='')
  for i in range (spaceNameLeft):
    print(" ",end='')
  print(testCases[choice-1].displayname, end='')
  for i in range (spaceNameRight):
    print(" ",end='')
  print("*")
  print("*                    *")
  print("**********************")
  print("")



print("u-CAT Test Suite v 1.0")
print("------------------------------------------------\n")
setup()
loop()
print("\nGood-bye!")
```

Figure 50. Python script of the main user interface.