|  |
| --- |
| Title: |
| Snake-like Robot |

| candidate(s): | | | | |
| --- | --- | --- | --- | --- |
| Håkon Bjerkgaard Waldum, Ruben Svedal Jørundland, Marcus Olai Grindvik | | | | |
| Date: | Course code: | Course title: | | Restriction: |
|  | IP304814 | Introduksjon til Mekatronikk | |  |
| Study programe: | | | pages/Appendix: | Library no.: |
| Automatiseringsteknikk | | | / |  |

|  |
| --- |
| Supervisor(s): |
|  |

|  |
| --- |
| Summary: |
| [The summary here will be identical to the SUMMARY section of the report] |

This report is submitted by students for evaluation at NTNU Ålesund.

Table of contents

[summary 4](#_Toc23235958)

[terminology 4](#_Toc23235959)

[1 introduction 5](#_Toc23235960)

[1.1 Goals of the project 5](#_Toc23235961)

[1.2 Project planning 5](#_Toc23235962)

[1.2.1 Planned timetable 6](#_Toc23235963)

[2 Background and theoretical basis 7](#_Toc23235964)

[2.1 Reason for project idea 7](#_Toc23235965)

[2.2 Usage in the real world 7](#_Toc23235966)

[2.2.1 Search-and-rescue 7](#_Toc23235967)

[2.2.2 USAGE 2 7](#_Toc23235968)

[2.3 Power Consumption 7](#_Toc23235969)

[2.4 Algorithm for searching 8](#_Toc23235970)

[3 METHODS 9](#_Toc23235971)

[3.1 Parts and Assembly 9](#_Toc23235972)

[3.1.1 Parts included in the snake-like robot 9](#_Toc23235973)

[3.1.2 3D Models 9](#_Toc23235974)

[3.1.3 Assembly 13](#_Toc23235975)

[3.2 Applied Theories 14](#_Toc23235976)

[3.3 Test Setup and Plan 14](#_Toc23235977)

[3.4 Computer Analysis Programs 14](#_Toc23235978)

[3.5 Communication 14](#_Toc23235979)

[3.5.1 Communication between ESP32 and PC 14](#_Toc23235980)

[4 results 15](#_Toc23235981)

[4.1 Theoretical Analysis 15](#_Toc23235982)

[4.2 Experimental Results 15](#_Toc23235983)

[4.3 Design Alternatives 15](#_Toc23235984)

[4.3.1 Alternative 1 15](#_Toc23235985)

[4.3.2 Alternative 2 15](#_Toc23235986)

[4.3.3 Alternative 3 15](#_Toc23235987)

[4.4 Challenges and Problems 15](#_Toc23235988)

[4.4.1 Communication challenges 15](#_Toc23235989)

[5 discussion 15](#_Toc23235990)

[6 Conclusion 15](#_Toc23235991)

[7 REFERENCES 16](#_Toc23235992)

[Appendix 17](#_Toc23235993)

*[This report contains predefined styles that you can use for the most common sections. The following styles are defined:*

*Heading 1 Heading for level 1 (hotkey Alt-1)*

*Heading 2 Heading for level 2 (hotkey Alt-2)   
Heading 3 Heading at level 3 (hotkey Alt-3)   
Body Standard text in a paragraph. Use this for all "normal" text (Shortcut Alt-A)   
Definition used mainly in the section entitled "TERMINOLOGY"   
References used in the Reference section.   
AppendixList used in APPENDIX section.   
Comment in blue text. Remove all the text of this type of report.]*

*[NB! This template provides a suggested structure of the main report. The main structure shall be followed.. However, the report must be structured by creating sub-chapters under main shapters. To some extent uou are free to decide how many sub-chaøpters and levels you want. Try anyway to avoid too many levels – normally 3 levels are enough. The level is meant the number of under-the chapter, for example, chapter 4.3.4 is on level 3, while section 3.2 is on level 2]*

summary

*This report is the final project report of Best Practice Course – System engineering. The project of our team is about*

# terminology

RRT – Rapidly Exploring Random Tree

RRT\* - A Variant of the RRT-method

GUI – Graphical User Interface

ESP32 – A type of Microcontroller

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

# introduction

*[This is the first chapter in the scientific report. It should treat the background for the project, the contractor, the problem / problem history and / or task to be solved. Here you should also say something about the scope or boundaries of the project.*

*Finally, you should briefly describe what the report further includes, amongst other things, what can the reader expect to find in the report.*

*Comment: This is where you will provide an introduction or a kind of presentation of the whole assignment. And it is also where you are going to present the issue to be resolved and any refinements made.*

*If the task has been dealt a specification of requirements, the main features from the requirements should be outlined here, with reference to the full requirements.]*

The project is based on modular robots. The task is to craft a modular snake-like robot that can find an object in a maze with the help of a overhead-camera as well as a front-mounted camera.

This is a task given to us by Houxian Zhang in the course “Introduksjon til Mekatronikk” as a final project for the course.

The goal is that the snake-like robot will be able to find its way around a maze to find an object, by having pre-existing knowledge of the maze by getting a feed from the overhead camera. During a new search after successfully finding the object, it should still be able to find its way around the maze if the maze has been changed, and the position of the object is changed.

There is also a goal of remote monitoring/control GUI. We want to be able to remotely control it via WiFi, and remote monitor what happens with the robot.

In this report we will go through our entire process for the task, from the beginning with building a theory for how this all should be achieved, till the final steps of testing and seeing it through to the end.

## Goals of the project

The specific goals of the project are as follows;

* Make a modular maze
* Use overhead camera to take picture of the maze
* Feed maze through pathfinding algorithm
* Send this path to the snake so it can efficiently search through it
* Search for object in maze using front-facing camera on snake
* Find the wanted object

The goal is that one can build the modular maze however one wants and the snake should be able to find the object in the most efficient way possible.

## Project planning

The project planning consisted of separating the project into its base parts; hardware and software. From there subcategories were created for each of the categories.

Hardware was separated into;

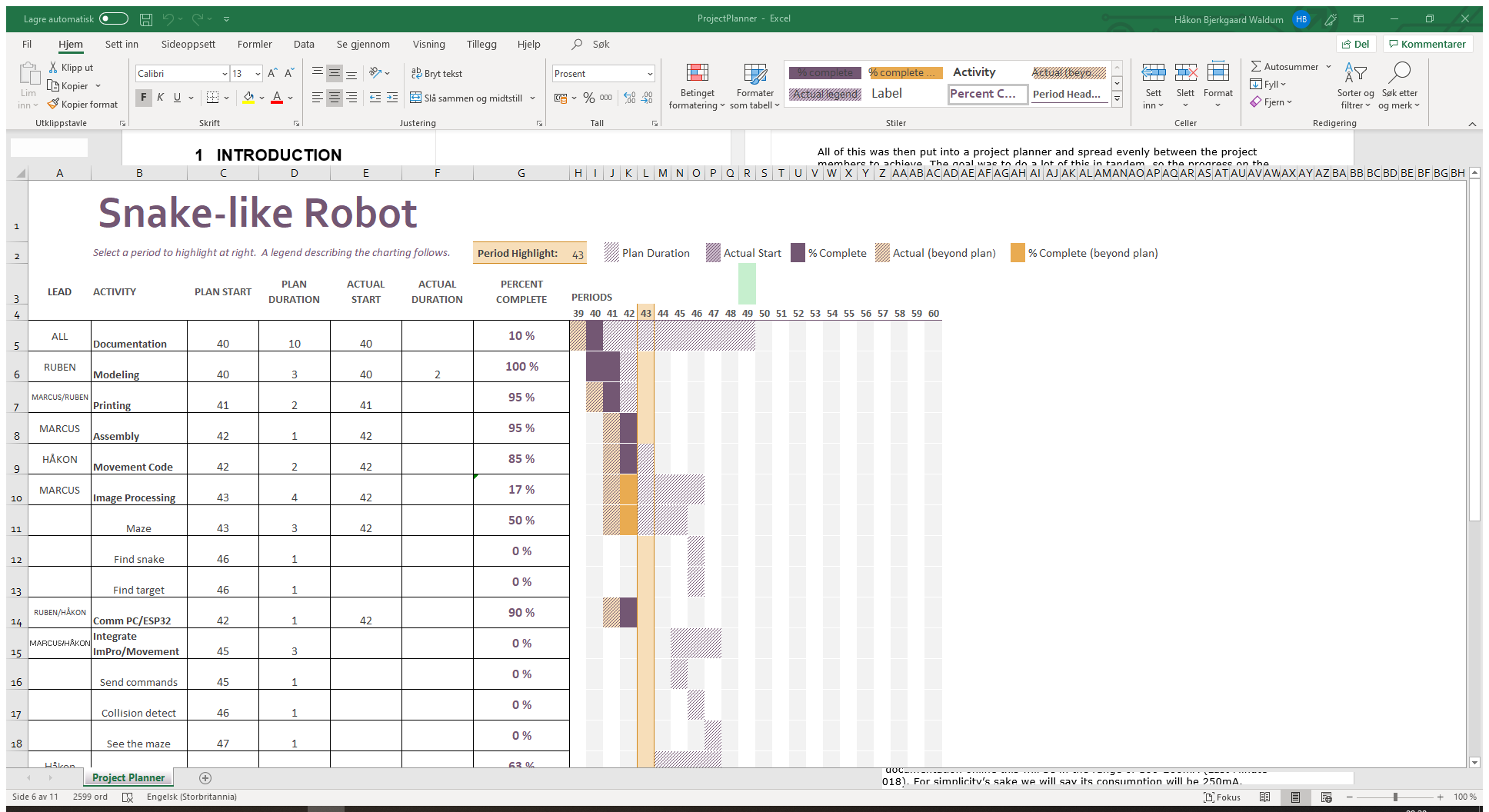
* Modelling
* Printing
* Circuitry
* Assembly
* Mazebuilding

Software was separated into:

* Image processing
  + Maze
  + Find snake
  + Find Target
* Communication ESP32/PC
* Pathfinding
* Remote GUI/Monitoring
* Giving commands to the snake

All of this was then put into a project planner and spread evenly between the project members to achieve. The goal was to do a lot of this in tandem, so the progress on the project would be efficient and swift.

### Planned timetable



Timetable for project

The timetable (found in the appendices) was created and the tasks spread evenly between the project members. Here there are goals with corresponding predicted times for starting and finishing each task. There is also information here about how long the tasks actually took, which is shown compared to what was predicted.

During the project this was regularly updated so each member could see how the others were doing with their task if the work was done separately instead of together at the university.

# Background and theoretical basis

## Reason for project idea

The reason that this is the project that was chosen was the challenges that would arise during the building and programming. Making the snake solve a maze is in itself not that big of a challenge, but incorporating a over-head camera to give the snake directions and finding the most efficient search-path is quite a challenge.

The combination of using the over-head camera with a front-facing camera presents quite the interesting problem; how does one synch the information from one camera with the other, and when the object in the maze is found, can we make the over-head camera be able to present this on just from the picture and position of the snake?

## Usage in the real world

This project has a lot of possible uses outside in the real world. Here some of them will be presented;

### Search-and-rescue

In a potential case of an earthquake hitting a city, one could have a drone fly over a location with a lot of rubble, to give a search-robot a general layout of the area, and guide it through the most efficient way of searching the area. Then by synching the robot and overhead drone, it could be possible to show on a video stream from the flying drone where there are possible locations for what the search-robot thinks are humans.

### USAGE 2

Fill text

## Power Consumption

When choosing the amount of batteries and their size, there are several things to take into account. The power consumption of the ESP32, as well as the 5 servos, the boost converter and the front facing camera will all factor into how long the battery time of this snake-like robot will be.

The ESP32 has several different modes it can run in to save on power consumption. For our purpose where we want access to its WiFi-capabilities we have to run it in what’s called “Active Mode”, which is the mode where its power consumption is biggest. According to documentation online this will be in the range of 160-260mA (Last Minute Engineers, 2018). For simplicity’s sake we will say its consumption will be 250mA.

The servos power consumption will all depend on how much torque it will need to apply to change its position. The producer of the servo has documented that the idle power consumption is 170mA, and at stall it will consume 1200mA (Tower Pro, 2019). From our experience we can see that it generally pulls around 200-300mA.

When it comes to what camera is used, it ended up being an ESP32-CAM, which has a OV2640 camera installed onto an ESP32. From what documentation is found about this unit, it seems that the maximum draw from it will be around 310mA (AI-Thinker, 2019). This is with flash on, as well as brightness set to maximum. So realistically it will draw less, but it’s better to calculate with the worst case scenario.

The boost converter has an efficiency of 90% according to the manufacturer (Texas Instruments, 2019).

We can therefore summarize the power consumption;

|  |  |  |  |
| --- | --- | --- | --- |
| **PART** | **POWER CONS.** | **AMOUNT** | **TOTAL CONS.** |
| ESP32 | 250mA | 1 | 250mA |
| ESP32-CAM OV2640 | 310mA | 1 | 310mA |
| MG995 | 300mA | 5 | 1500mA |
|  |  |  |  |
|  |  | TOTAL: | 2060mA |

The total draw at constant power consumption will be around 2060mA. When accounting for the boost converter we have to calculate this over to Watts, and see how much watt the boost converter will have to produce to reach this amount.

This is the general formula for calculating power. The voltage this all will run on is 5V.

With an efficiency of 0.9, the boost converter will have to produce;

Which then brings us back to;

So our current draw will be 2289mA.

Our snake-like robot has mounted four 2000mAh batteries. This means that our snake-like robot will in ideal situations run for just about 3.5 hours before running dry.

## Algorithm for searching

After getting a picture of the maze, the picture is run through several filters before being used in a RRT\*-algorithm, which will find a path through the maze.

The RRT\* method is a continuation of the regular RRT-method. It is based on random node-placement and checking for collision before expanding onwards and doing this on repeat until it has run all its iterations.

The big difference between RRT and RRT\* is that RRT\* will look at old nodes when continuing its path to check if it can rewire the path to be more efficient. It does this by giving each node a “cost”, which is based on its distance to its parent node. It therefore compares the cost to its current parent, to another node that is close to check if the cost can be reduced.

This all results in a more linear path to the goal, instead of the path which would come from the regular RRT-method, which almost always will be very full of twists and turns and will not be as efficient.

# METHODS

## Parts and Assembly

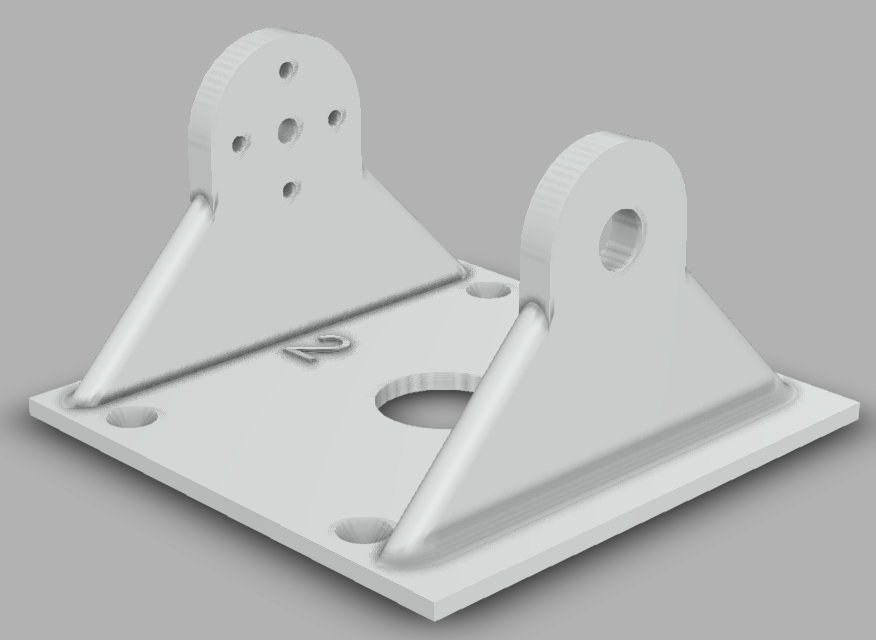
### Parts included in the snake-like robot

The snake-like robot has a lot of parts included in it to work properly. It consists of a total of 5 modules that are connected to make it able to move as we want. The modules are basically all the same, but some of them have some unique flavor for all the parts to be able to fit properly. The front and the back are also unique in that the front has to be able to fit the front-facing camera, and the back has to have a end-plate so the batteries do not fall out.

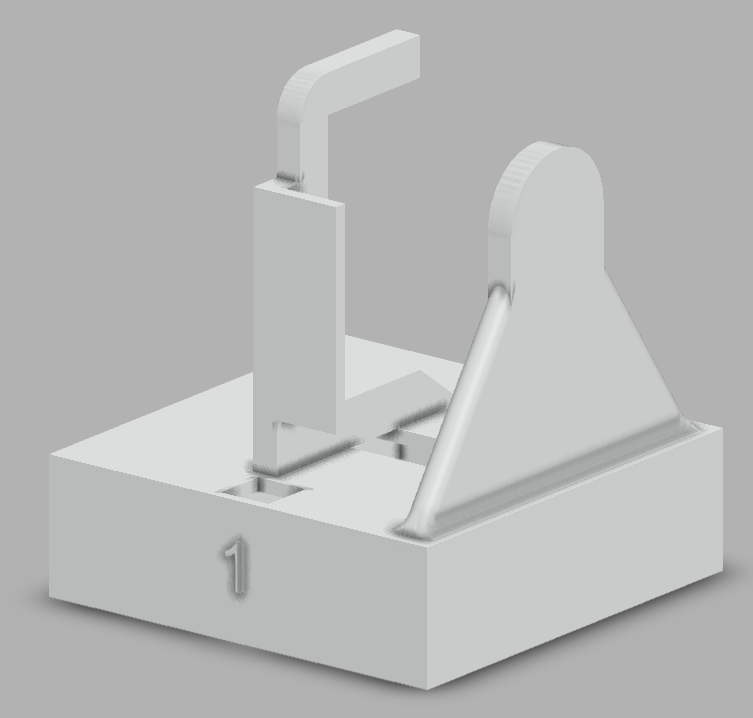
|  |  |  |
| --- | --- | --- |
| **PART** | **AMOUNT** | **FUNCTION** |
| ESP32 | 1 | Controlling the movement of the snake-like robot, as well as communicating with the PC which does all the image processing, pathfinding etc. |
| Boost Converter | 2 | Converts the nominal voltage from the batteries to 5V for the servos and the ESP32. |
| ESP32-CAM | 1 | ESP32 with a camera mounted on it, mounted in the front of the snake-like robot, used for finding the object in the maze. |
| Battery 2000mAh | 4 | Used to supply the robot with the power required. |
| LiPo USB Battery Charger Board | 1 | Used to slowly recharge the batteries between usage when connected through a USB. |

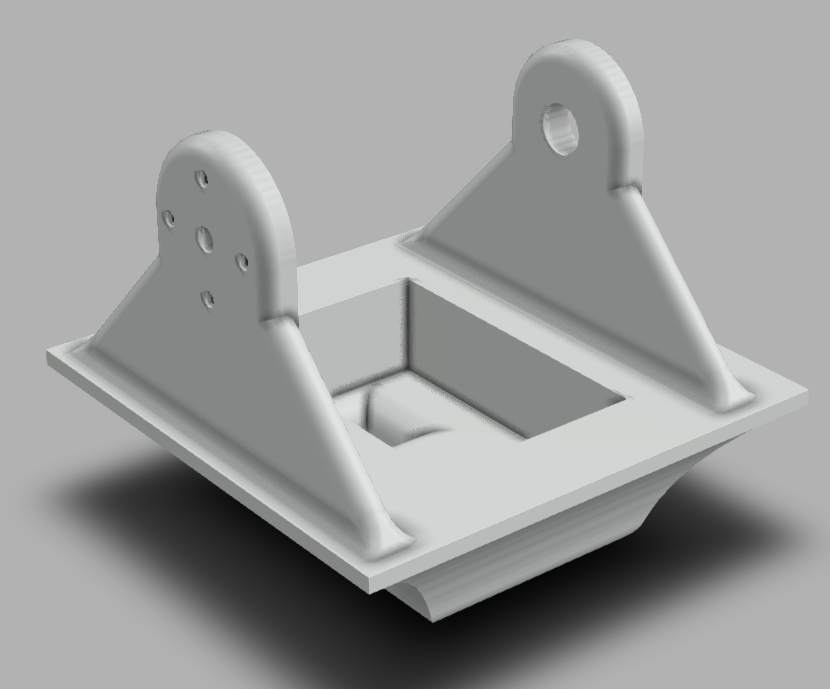
### 3D Models

All mechanical parts for the snake-like robot are modelled using Fusion 360, and are made so it can be expanded with relative ease. Under are shown the basic parts, with also some unique parts, and the complete assembly.

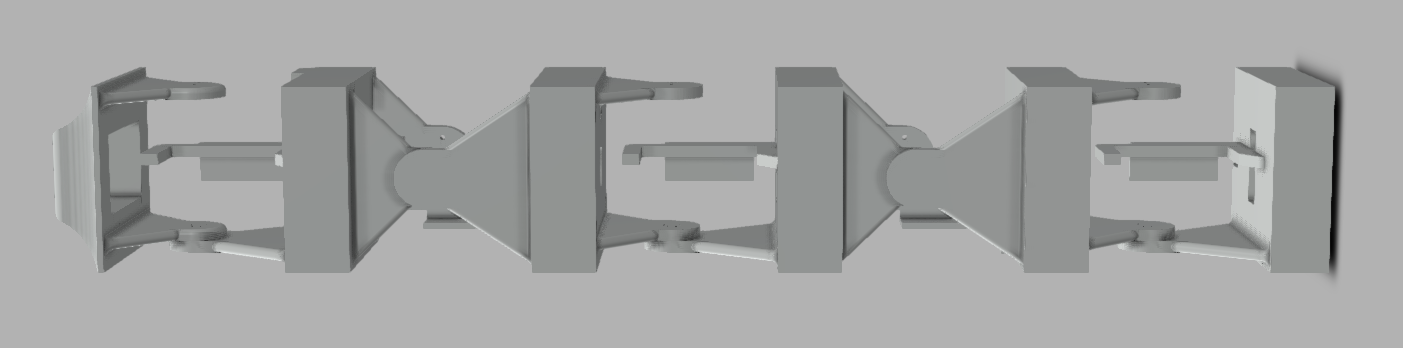


Over one of the parts are shown, this is a basic part that is connected to the other piece which will have the servo mounted on it.

This part is connected to the first part shown. This is the one that has the servo mounted on it, it also has a thick base which as room for batteries, boost-converters or similar parts.

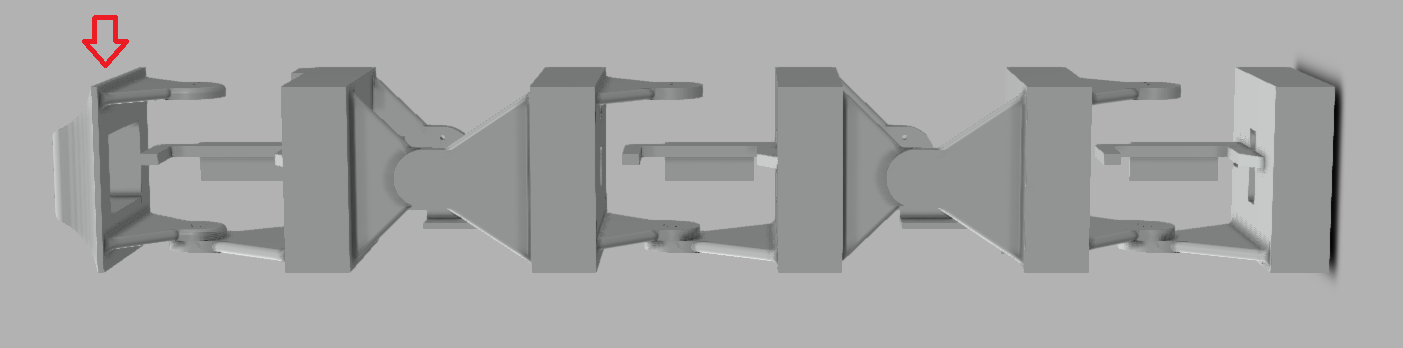


This is the front of the snake-like robot. This has space in it for the front-facing camera. This part will be painted in a distinct color, which makes it possible to find the front of the snake-like robot with ease.

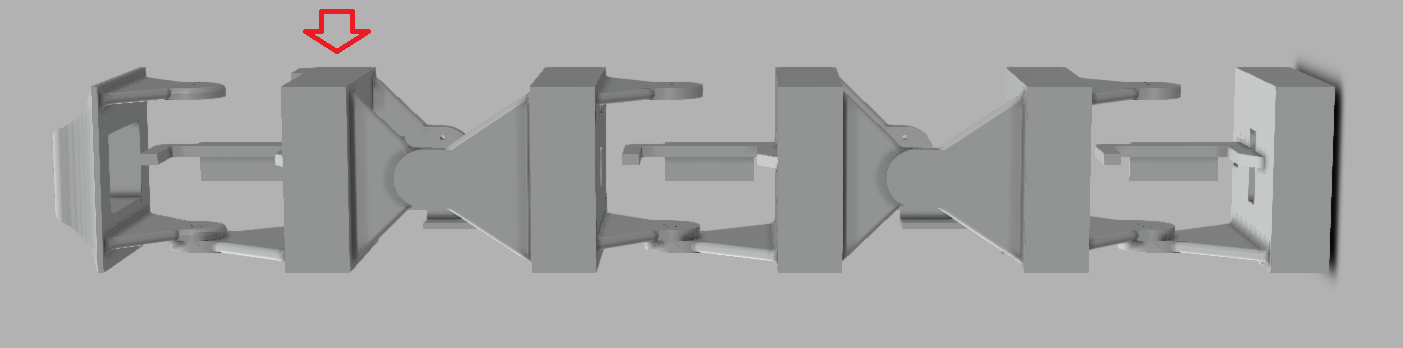


The snake-like robot assembled completely is shown above. As mentioned earlier, it consists of 5 modules, with a designated front, and a designated back. All the parts in the middle between the two modules mounted together are made for storing the electronic parts needed for the project.

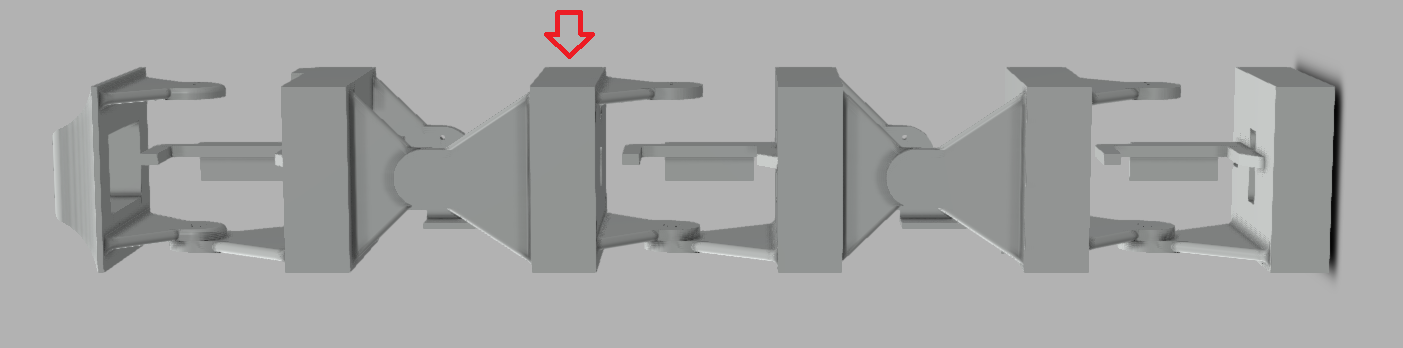
### Assembly



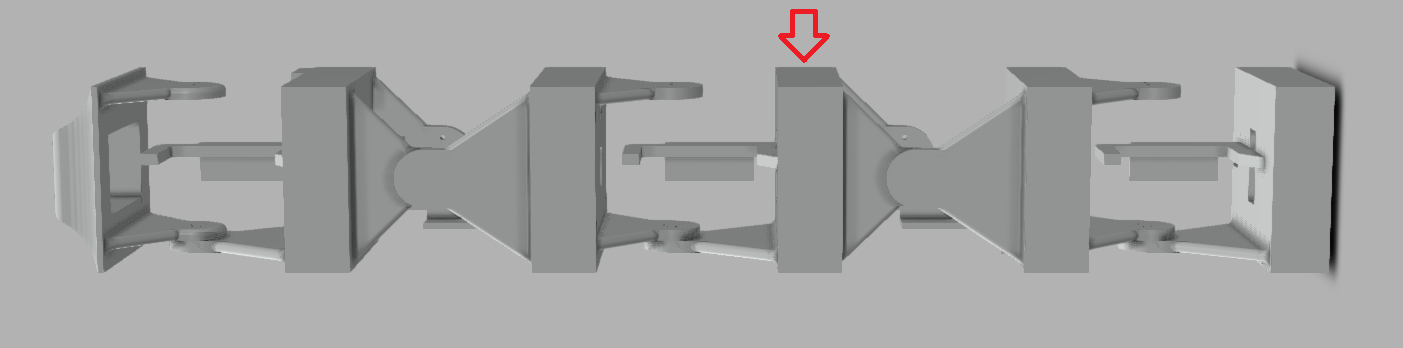
In the first piece there is only the ESP32-CAM which is mounted. This part is only connected to power, and sets up a WebServer on its own IP-adress, port 80. Which is accessed by Python to retrieve a video stream or a picture.



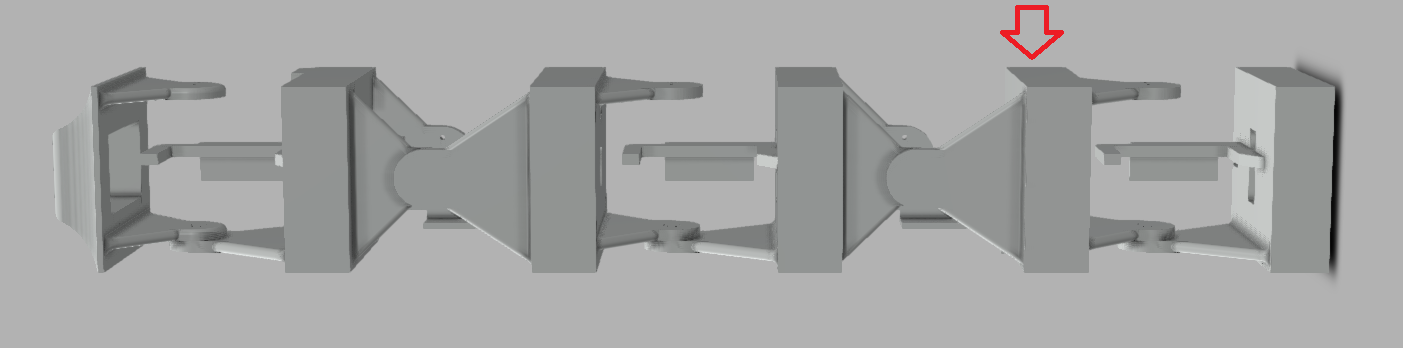
In the second piece there is mostly just circuitry, a connection point for ground and +5V. This is just a junction for the cables basically.



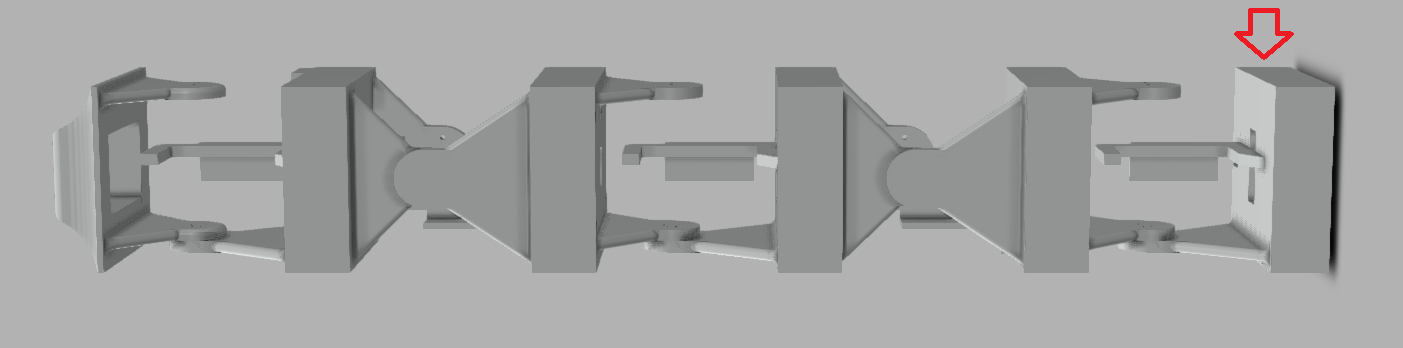
In this part the ESP32 is mounted, as well as a button which is mounted to be able to reprogram the ESP32.



‘In this part the whole power circuit is mounted, which consists of the two boost converters, external power port as well as a power button to be able to remove power from the servos and the ESP32.



This part consists of two batteries connected in parallel. These just supply the rest of the robot.



This part consists of two batteries, as well as the USB recharging board, which makes it possible to recharge the batteries slowly.

## Applied Theories

## Test Setup and Plan

## Computer Analysis Programs

## Communication

### Communication between ESP32 and PC

Our initial plan for communication between the ESP32 and the computer was to use WiFi and send information via TCP. This is because TCP sends a message back when receiving packets to give notice if the package does not arrive or there is any other problem (receiver not connected to the internet etc.). But during the testing phase we saw that sending just a 100x200 array could take up to 9s one way. This would take way too much time to be able to actively send pictures and get information back to the snake to tell it if it has found the object.

It was quickly decided to try out UDP to check the time for sending information via this protocol instead. The downside to using UDP is of course that it is “fire-and-forget”. It does not care if the receiver is not online, it send the packet and is quite happy with the result regardless of what happens with the packet.

During testing here it was found that sending the same array as earlier took less than 1ms. This is a drastic improvement and it was quickly decided that this is the protocol to use. We do not see it as a huge risk, as the PC and the snake will not be far from each other, and we will also make a checksum-kind of check to see that the package is received as it is expected.

As for how to send the pictures from the front-facing camera to the PC, after testing the first camera that was supplied (a VC0706 UART Camera) it was found that it did not want to communicate with anything. Therefore it was changed with a ESP32 with integrated camera. This camera creates a webserver which it “streams” its content to. This makes it quite easy to use Python to send requests to the server, which makes the camera take a snapshot.

# results

## Theoretical Analysis

## Experimental Results

## Design Alternatives

### Alternative 1

### Alternative 2

### Alternative 3

## Challenges and Problems

### Communication challenges

During the programming of the Arduino the program was sliced into separate modules to easily be able to test each module to see that everything worked as expected. During the merging of the programs there came some challenges that were not expected.

Specifically during the merging of the movement and the communication modules. When initializing the Arduino program during the testing here, everything went as expected. The module initialized and attached the servos to the right pins, and then connected to the WiFi and sent the test-package to a given IP and Port. But during the parsing of an incoming packet the ESP32 raised a Guru Meditation Error (the ESP32’s variation of a BSOD), saying “LoadProhibited” and gave a dump of information.

From what we could deduce from documentation found online this was because the application attempted to access a member of a structure, but the pointer to the structure was NULL (ESPRESSIF, 2019). After a lot of testing and debugging we found that if we changed the pins the servo attached to, everything went fine. After this worked, we realized that one of the pins we attached the servo to was pin 16, which is an RX-pin, which probably is what caused our problem.

# discussion

# Conclusion

*[Here you should present the main results of the work together with the experience you've gained in the process. Here you will summarize the most important chapter discussions, and arrive at a conclusion. Did you solve the problem as required or expected with the chosen methods? Was the result according to the mission stated bye the employer? What did we learn from this project, both scientifically, and not least in relation to the work process of a project? ]*

# REFERENCES

ESPRESSIF. (2019, 10 22). *API Guides: Fatal Errors*. Hentet fra ESP-IDF Programming Guide: https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/fatal-errors.html

ITEAD. (2019, 10 16). *Wiki: VC0706 UART Camera*. Hentet fra https://www.itead.cc/wiki/VC0706\_UART\_Camera\_%EF%BC%88Supports\_JPEG%EF%BC%89

Last Minute Engineers. (2018, December 23). *Insight Into ESP32 Sleep Modes & Their Power Consumption*. Retrieved from https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/

Texas Instruments. (2019, 10 16). *LOW INPUT VOLTAGE SYNCHRONOUS BOOST CONVERTER.* Hentet fra https://www.sparkfun.com/datasheets/Prototyping/tps61200.pdf

Tower Pro. (2019, 10 16). *Products: MG995*. Hentet fra Tower Pro Online Shop: https://www.towerpro.com.tw/product/mg995/

*[Authors, title of book or article, name of journal or publisher / publisher, or no date for the journal, year, place as referred to in the report. Course lectures can also be referred to, as with the title on the subject and the name of the presenter. Internet pages must also be included. Even oral discussion partners can be included in the reference list, when this is a source of important or detailed information used in the report*

*see example below]*

1. *Design of Propulsion and Electric Power Generation systems* – H.K. Woud and D.Stapersma
2. Matlab. Wavelet Toolbox. Users Guide. The Math Works Inc., 1997
3. Pugh, D T., Tides, Surges and Mean Sea-Level. John Wiley & Sons. New York, 1996
4. http://dieselmarine.com/

Appendix

*[Material and data prepared or collected in connection with the project, but not natural to include in the main part of the report. The reason cam be the level of details, volume or format.*

*Typical examples are: detailed calculations or analysis, set of design drawings, supporting information, computer code etc…. ]*

Appendix A Project Planner

Appendix B FEM analysis of cylinder liner

Appendix C Design drawings

Appendix D Material properties

Appendix E etc….etc…..