BATTLESHIP

Feil Charel

JUNE 24, 2022
BTS-IOT
Lycée des Arts et Métiers
Project CREPO1 2021/2022
1, rue de l'école,
Bissen L-7768

Table of contents

1 Description of the task

The task given for this project was to produce a “connected boardgame” in one semester, we were given a free choice on what game we would produce or even create a new one. However, one game can only be covered by a maximum of 2 students.

The goal of this project is to produce the boardgame in a way that the current state of the game is sent over the internet, more precisely via the MQTT protocol. Furthermore, the state has to be visible on a webpage. That means a movement made by the player on the board has to be visible on the website in close to real time. Another requirement is that we use a microcontroller in our boardgame and that the whole thing must be battery driven, that means we can’t exceed a certain power consumption limit to be able to finish a game.

How we implement these requirements is up to us. However, we can’t exceed our given budget of 150€, which can be challenging but is manageable. Another big point is that the game can easily be maintained, that means not gluing everything together and forgetting about it.

These are the minimum requirements, but we were also given a list of bonus requirements which we don’t have to implement but can.

The first bonus requirement is that the configuration of the microcontroller is saved in the SPIFFS and can be modified on a locally run webserver. This would allow us to make it truly portable and not only work with one wireless network and one MQTT broker.

Another bonus requirement is the possibility that the boardgame can be remotely controlled. With this requirement we were given 2 choices, either show on the board where the other player would set his piece or create a mechanism to physically move the piece to another position.

The ability to play against the “computer” or another person on the internet, these are two of the bonus requirements, where LEDs indicate the move of the opponent.

2 Introduction

The beginning of the project was pretty overwhelming, because I wasn't set on my idea of making Battleship, but as my ideas grew, my confidence in finishing this project also grew.

In the next few pages, I will explain how I designed, built and the logic behind the code of my iteration of the battleship boardgame.

The first thing I did after finding an idea I was happy with was to look for a solution to recognise where the ships would be and how to show where you would shoot or have been hit.

In the beginning, I sprung to designing the case and to expand my idea into a working project.

All of the photos that I took during the project are in a folder called "Fotos" in the ZIP folder.

2.1 How it works

The Battleship boardgame can send and receive positional data from the website and update itself accordingly. The board is turned on by a switch, which you can see through the back and the game is solely controlled by a joystick with an X, Y-axis and a button.

When the board is started, the micro controller tries to establish a connection with the Wi-Fi network (currently hard coded as the school's Wi-Fi), if successful the LEDs on the board flash green. After that the board tries to establish a connection with the MQTT broker and the LEDs flash blue when successful.

Then you can see a green "S" being drawn with the LEDs on the upper half of the board. This means the board is ready to start a game, to start you need to press down on the Joystick, located on the bottom part in the bottom right corner.

This activates the reed-switch matrix, which checks for ships on the board itself. Due to problems with this matrix, I implemented that you need to confirm all of your positions with the joystick to eliminate flukes. The ships will be illuminated by blue LEDs.

The other player on the website can place ships and rotate them to their wish, if the player is happy with this ship's position he can click the "SET" button and confirm this position and place the next ship. When all ships are set, the player has to press the "SET" button again to confirm all positions and send them to the MQTT broker.

When all positions are set on both sides, the game can begin.

When it's the turn of the board player, they can select the position that they want to shoot at. This gets shown on the upper half of the board by a blue dot (I will call it the cursor from here on). The cursor is controlled by the joystick's X and Y-axis. To confirm and shoot at that given position the player needs to press down on the joystick.

The player on the website can shoot the position of his liking by simply pressing on the position that he wants to shoot at on the screen.

The shots of the website player are shown on the lower half of the board by green or red LEDs (green = missed, red = hit) and on the website, where the text of the button will change to “HIT” or “MISS” accordingly.
The players always take turns to give them an equal chance of winning.

If the board player has won, the board will show a green 😊 on the top half of the board, otherwise the board will show a red ☹ on the top half of the board. On both occasions, the bottom half will still be illuminated, so that the players can review how close it was.

3 Main part

3.1 Description of the task

The basic task was to create a connected boardgame, of which the current playing state can be accessed on a webpage.

3.2 Step-by-Step description

3.2.1 Beginning ideas

My idea in the beginning was to make the popular “Battleship” game connected and a little more modern, regarding most sets that you can buy are with plastic pins. My idea was to replace these with LEDs and instead of telling the other player: “I’m shooting there XY”, the player can choose where to shoot with a joystick and press it to confirm his selection.

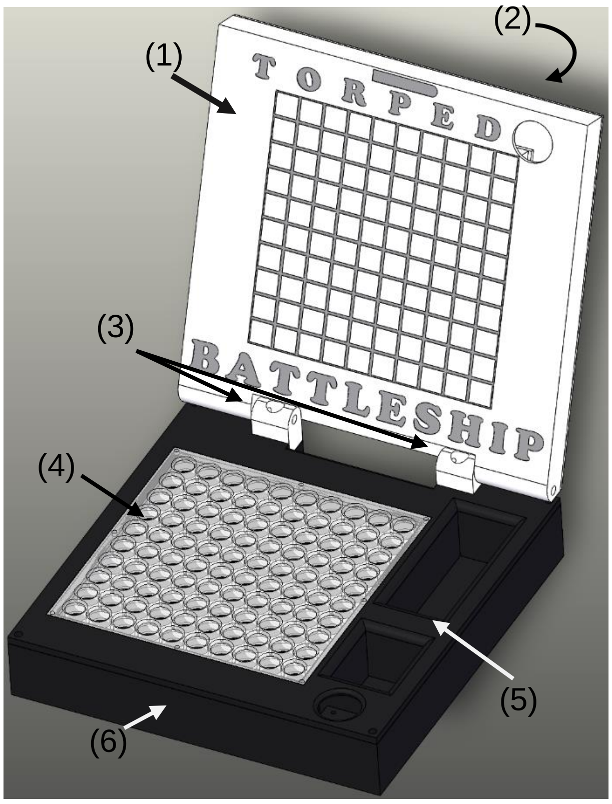
In my first thought, I wanted to make 2 boards, so that 2 players can play directly against each other. However, this would be a little too much regarding that it is still going to be a prototype, the time frame and budget of our project. I decided to make 1 board and have the other player play on the website or that you can play against the computer.
Those features also turn out to be difficult.

The main inspiration for this project was that I hadn’t found a battleship boardgame which is connected to a website. I had ambitions to finish the project and have a working project and website. However, I encountered a lot more problems than anticipated, on which I will further go into later.

3.2.2 Designing and 3D-printing

3.2.2.1 Designing

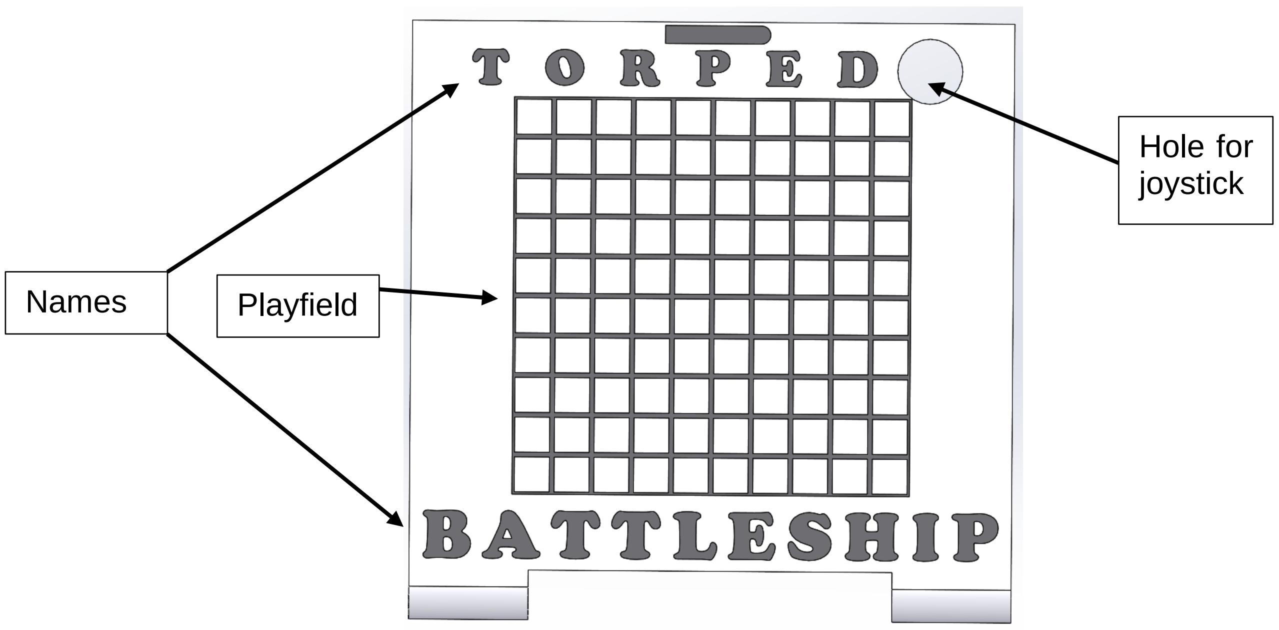
One of the first things I did when my idea was settled, was to start designing the necessary parts for my project. This was one of the first things I could do, because I didn't have any parts and needed a way to visualize the project.

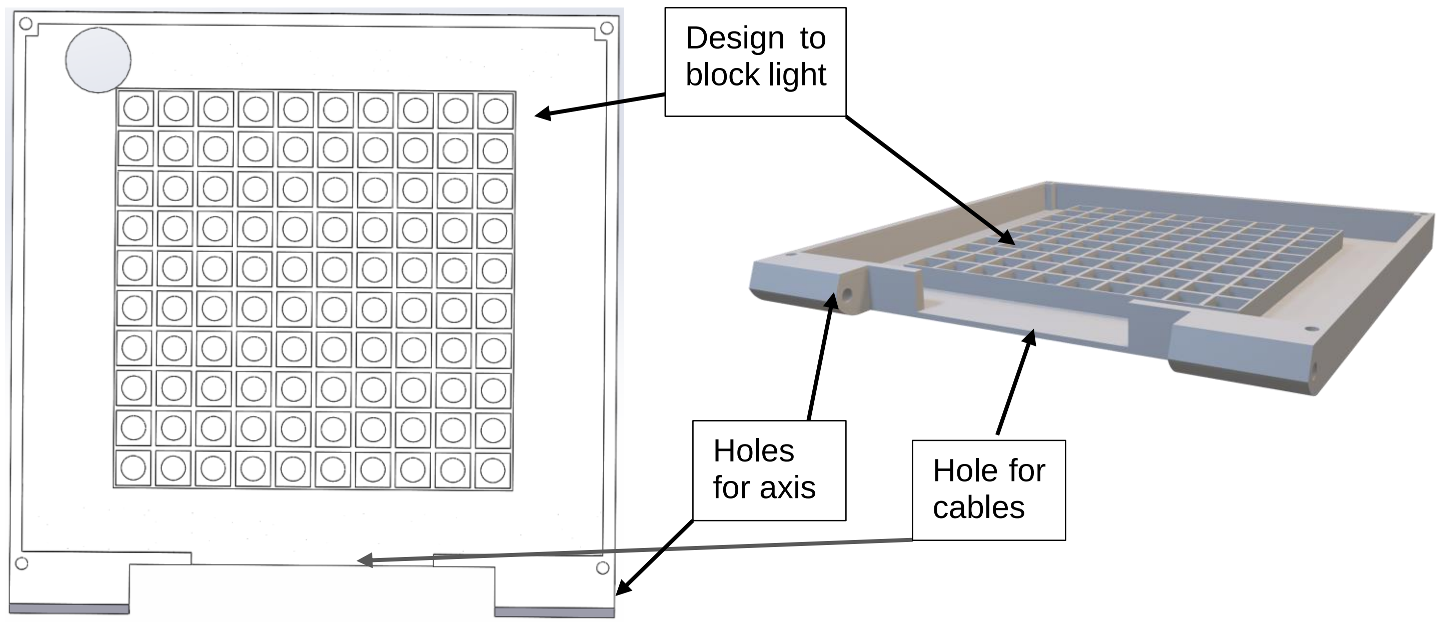


(1) Top part
(2) Top cover
(3) Hinges
(4) Bottom-Holes
(5) Bottom-Plate
(6) Bottom

3.2.2.1.1 Top Part

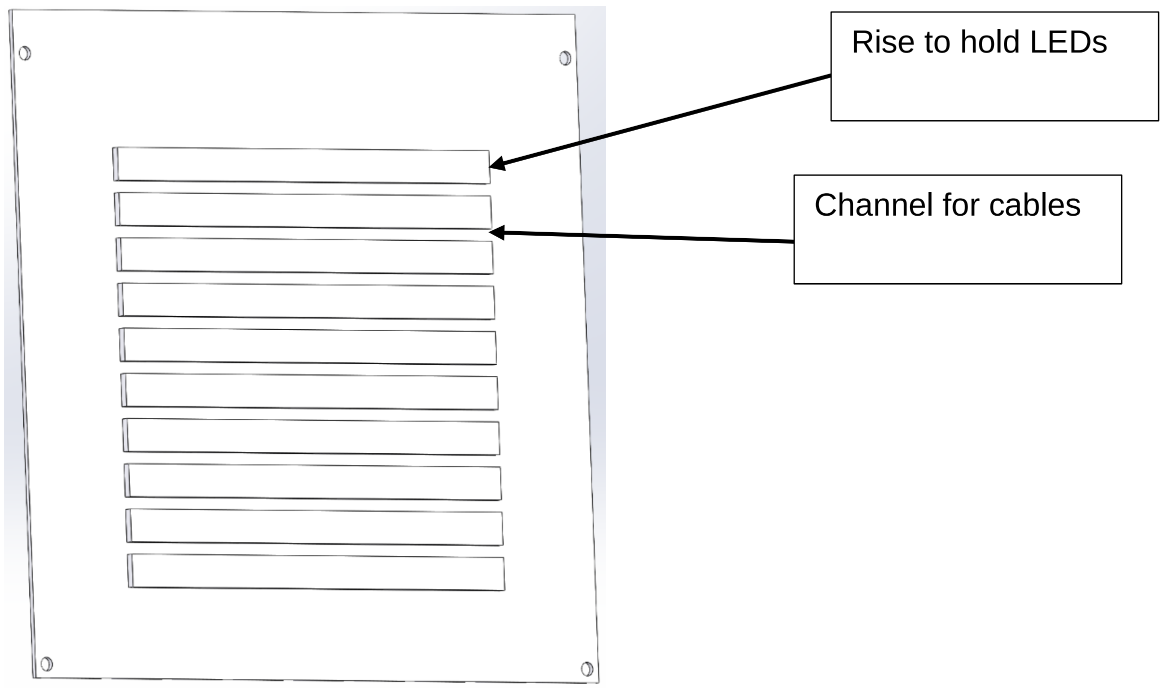
The top part is, as the name suggests, the biggest piece of the top half of the board. It has the name of the game written on it, “Battleship” and additionally the fictional name of the product “Torpedo”. The name “Torpedo” was suggested to me by Luka Theisen, because I needed a way to hide the joystick when folded. This part also was designed to block most of the light of the LEDs behind it and only let through a small circle. The circle is printed very thin and the square around it is much thicker. The squares seen on the front represent the playing field. In the bottom half of the part you can see there is an opening for the cables to run through and a hole to put an axis through, to connect to the hinges.





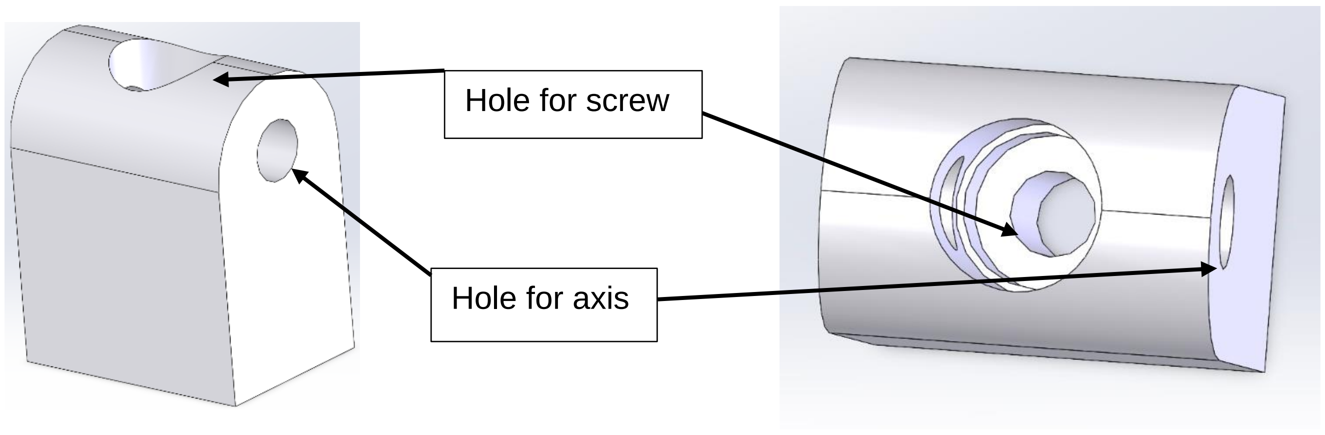
3.2.2.1.2 Top Cover

The top cover is as the name suggests, the cover for the top, which holds the LEDs in place, with little channels for the cables and gets screwed on the top part. This is also the part that prevents the whole top from falling back.



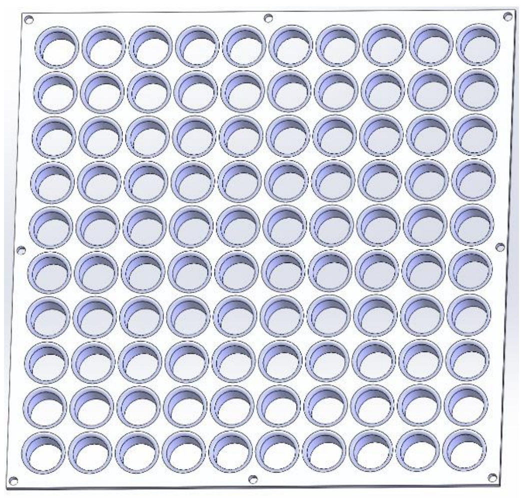
3.2.2.1.3 Hinges

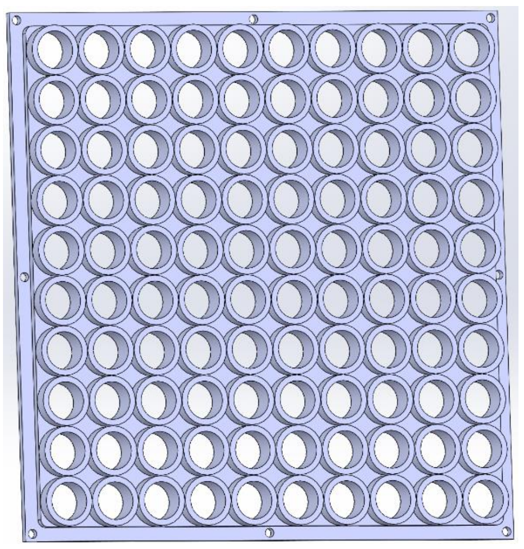
The hinges are their own part to simplify 3D printing the whole board. They get screwed in place to the bottom part and hold the axis through them.



3.2.2.1.4 Bottom Holes

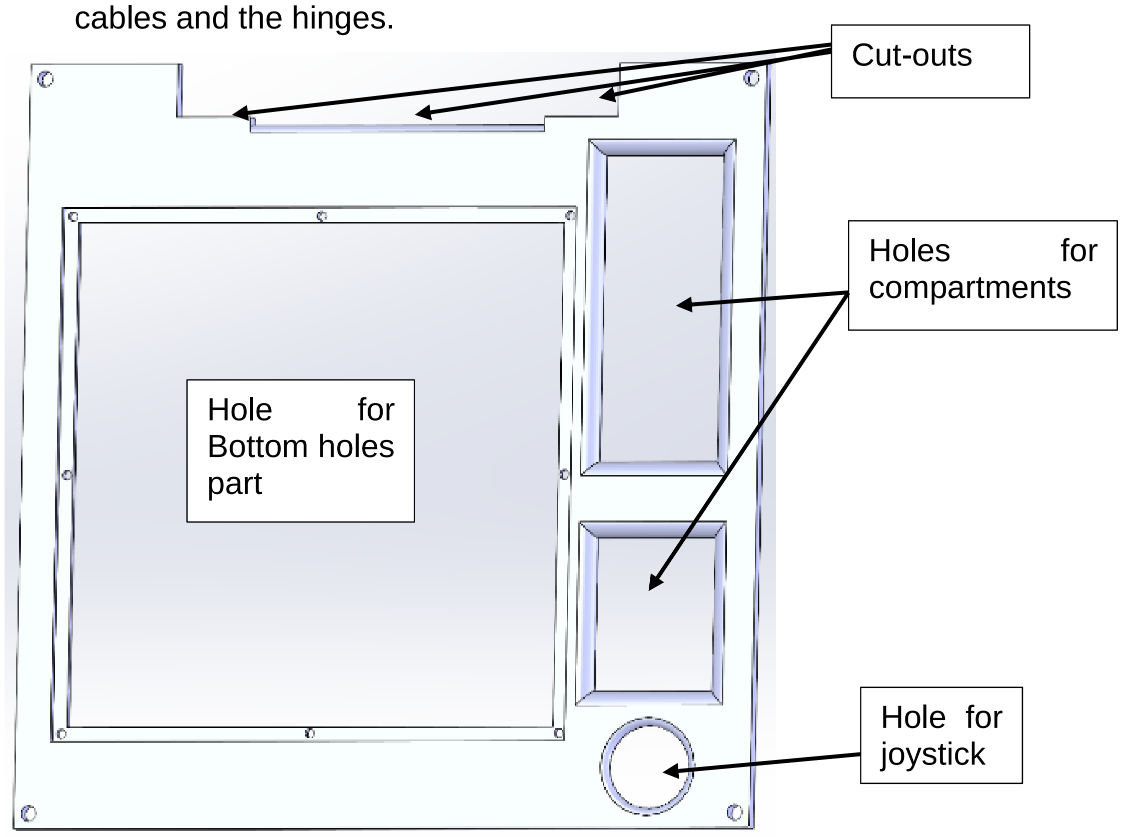
In the first iteration of my design, this part and the bottom plate were one. However, throughout the project I realized that the visibility through a black plate like that would not be sufficient, that's why I decided to create this part and print it in transparent filament later on. This part gets screwed on to the bottom plate and holds the ships securely.





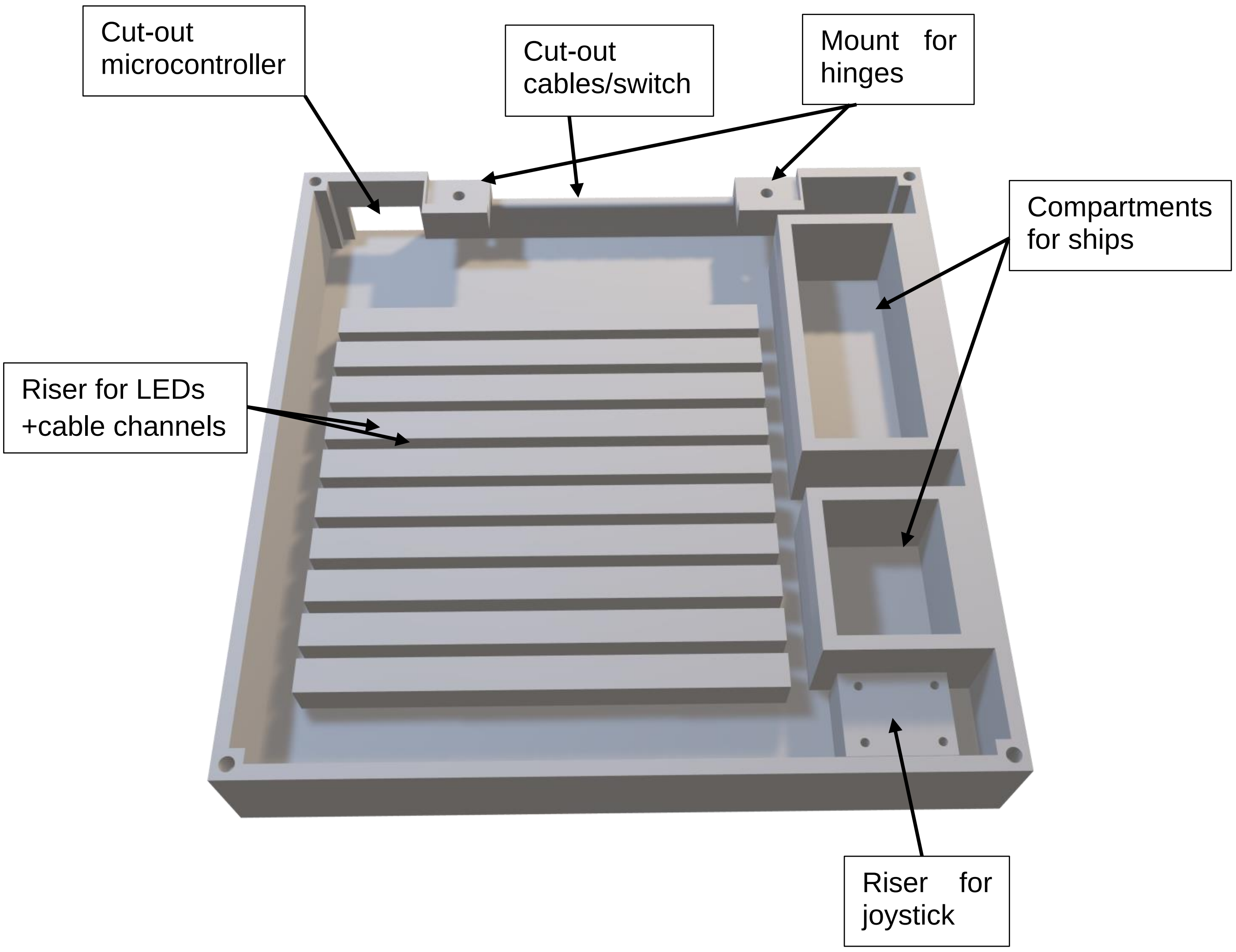
3.2.2.1.5 Bottom plate

The bottom plate has many openings, which all have their function. Firstly, there is the place to screw the Bottom holes part on and secure it. Then there are 2 openings for the compartments, which hold the ships. There is also an opening for the joystick to stick through, this is rounded to ensure more moveability with the joystick and that the user won't be hurt while playing. Furthermore, there are cut-outs to account for the cables and the hinges.



3.2.2.1.6 Bottom

The bottom part is the heart of the design. This holds the micro-controller, battery, joystick, LEDs, and the ships. There are again risers for the LEDs, which also create channels for the cables of the LEDs to go through. The compartments where the ships are stored are on the right and there is also a riser for the Joystick. In this part there are also the mounts where the Hinges attach and a cut-out for the cables, switch and microcontroller. The cut-out for the microcontroller has been made to be able to charge the battery without removing it.

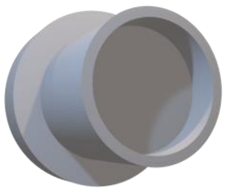


3.2.2.1.7 Ships

All of the ships have slots for the magnets to be placed in.

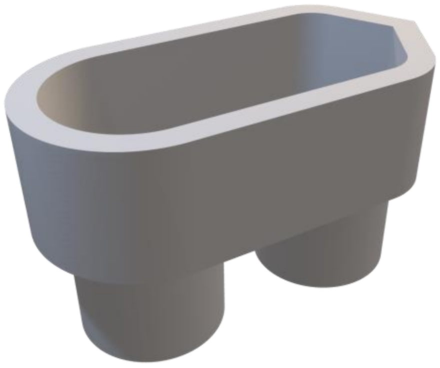
3.2.2.1.7.1 Boat

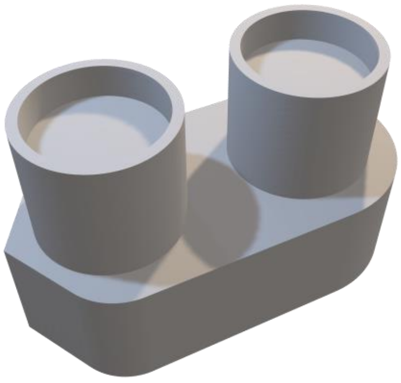
A single place ship, which is in this case only a glorified circle.



3.2.2.1.7.2 Destroyer

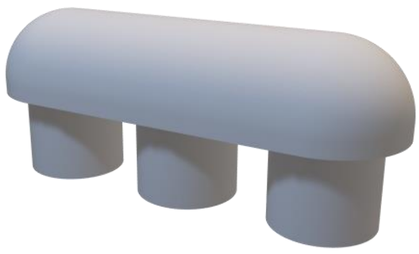
This is a 2-place ship and has a cut-out on the top to allow for more light to shine through from the bottom.

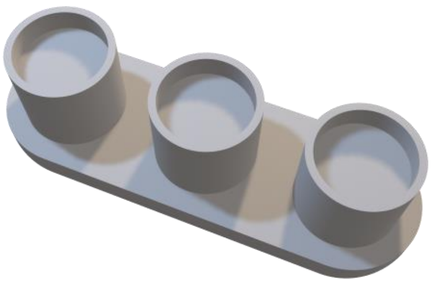




3.2.2.1.7.3 Submarine

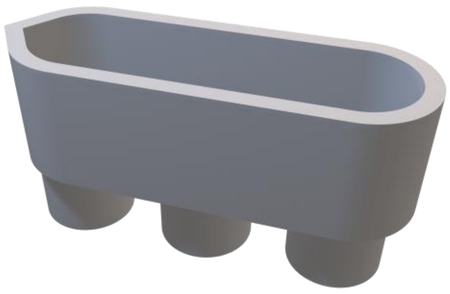
This is a 3-place ship and rounded off, to give more on an impression of a submarine.

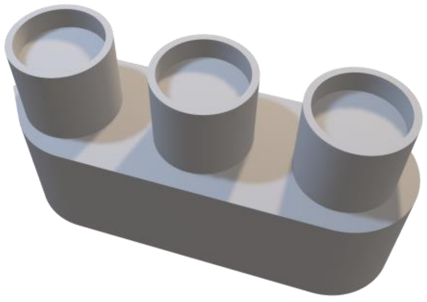




3.2.2.1.7.4 Cruiser

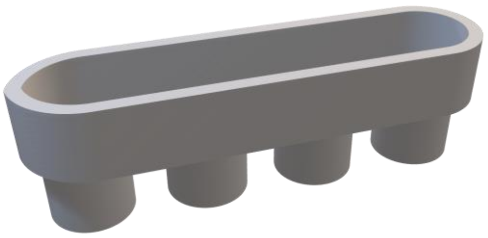
This is also a 3-place ship, but with a cut-out for the light again.

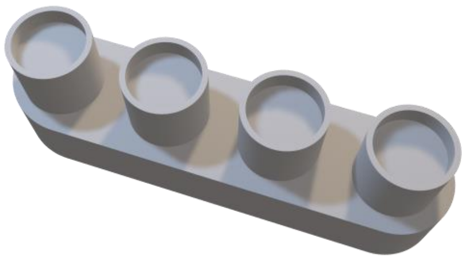




3.2.2.1.7.5 Battleship

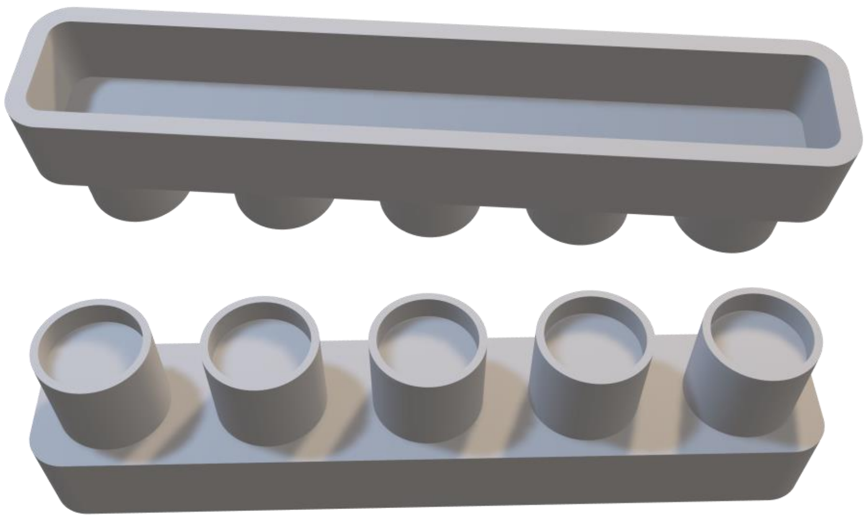
This is a 4-place ship with a cut-out for the light.





3.2.2.1.7.6 AirCarrier

This is a 5-place ship with a cut-out for light.



3.2.2.2 3D Printing

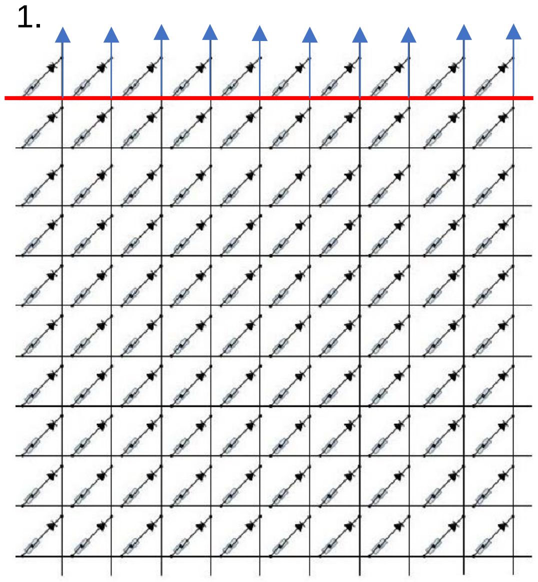
I have been 3D printing over a few days, and I have finished all the parts and ships. Most of the parts were easy to print, except for the bottom holes, which I had to retry a few times due to bad bed adhesion and under extrusion.

3.2.3 Reed Switch Matrix

The reed switches are explained in this chapter.

I use a 10x10 matrix to get the positions of the ships with the reed switches. A matrix is basically a grid, which in this case reduces the number of pins needed for my playfield to work. Due to this matrix I needed 20 pins instead of 100, which is a big improvement.

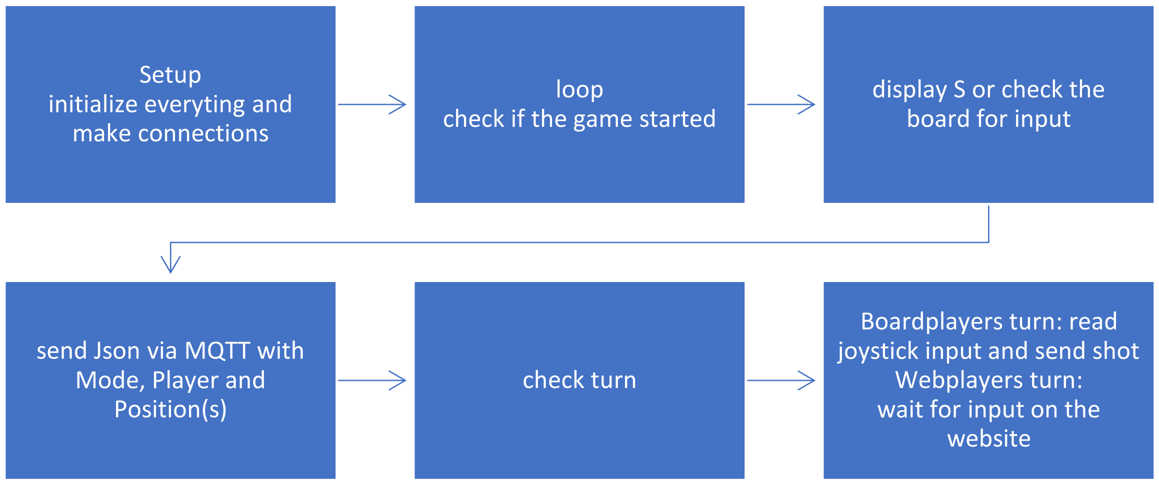
To read the matrix, the top side is being read and the right side is being powered 1 by 1.



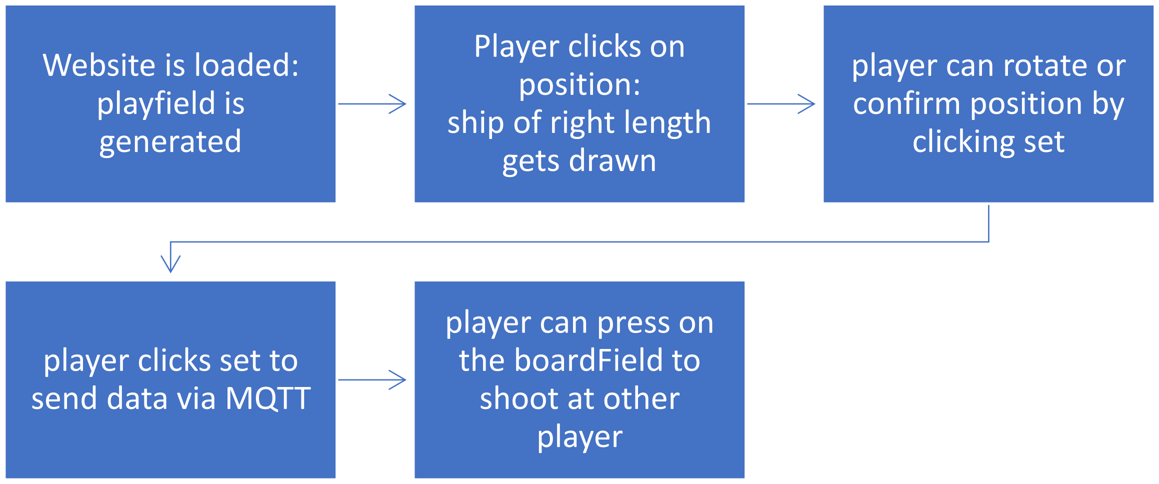
While 1 row is being powered, all of the columns are being read to get every input switch from that row. As you can see on the picture, there are diodes in between the switches and the reading rail, that's to prevent falsified inputs due to closed switches that send power through the wrong one.

3.2.4 Coding /explanation of code

You can find my code in the ZIP folder. It is quite large, I commented all of the main functions and the functionalities. I tried to select the variable names to be comprehensive of their function in the code. This schematic show the basic functionality of the Arduino code.


Setup
initialize everyting and make connections
loop
check if the game started
display S or check the board for input
send Json via MQTT with Mode, Player and Position(s)
check turn
Boardplayers turn: read joystick input and send shot
Webplayers turn:
wait for input on the website

However, this wasn't the only code written, the other half is on the website. The website and the MQTT broker are running on a Linux server (a VM on the school pc). For the website I had to revise my HTML and CSS knowledge and learn a little bit of JavaScript, because the connection of the Website to the MQTT broker is made via JavaScript.
This also allowed me to make the code a lot smaller for this large number of buttons that are displayed on my page. This schematic explains the functionality of the website.


Website is loaded: playfield is generated
Player clicks on position: ship of right length gets drawn
player can rotate or confirm position by clicking set
player clicks set to send data via MQTT
player can press on the boardField to shoot at other player

3.2.4.1 Testing

The first test was to see how a matrix and MQTT works, I tested this by programming a small TicTacToe game with Yves Fischer, where we connected our ESP32s to one MQTT broker and sent each other the last move so that we could play against each other. This updated a little graphic in the Serial output of the microcontrollers. The program is also in the ZIP folder.
I also did a lot of testing with the LEDs how they would control, because addressable LEDs were also completely new to me. I had always tested my program when I added small things, which cost me a lot of time.

3.2.4.2 Problems

I encountered a lot of problems during the coding and had many setbacks. I had to start over with the website 2 times, because the ideas I had would have been much lengthier than necessary and with an already very big Arduino code, I wanted to keep the rest at least shorter. The fact that I was overwhelmed with JavaScript at first did not help as this is the easiest way to connect the website to the MQTT broker according to many sources on the Internet. I had also asked for help from peers, which helped a lot to get the website mostly functioning correctly. The problem with the website at the moment is that after the first received message, the page disconnects itself from the MQTT broker and it doesn't reconnect, even if the function is called to be connected again.

The ESP32 also gave me a few problems, for example, the joysticks axis has to be read out by an analogue pin and in the first attempt this worked very well, but when I added Wi-Fi and MQTT connectivity, it didn't recognize one of the axis anymore. I later found out that I had this axis connected to a pin under ADC2 and the axis that worked under ADC1. I moved both axis to ADC1 pins, which fixed the issues.
The board worked fine with the joystick input and displayed the right places, but sometimes after a few turns, some of the pixels just changed colour when they weren't supposed to and I haven't found this bug, but I couldn't reproduce it since.

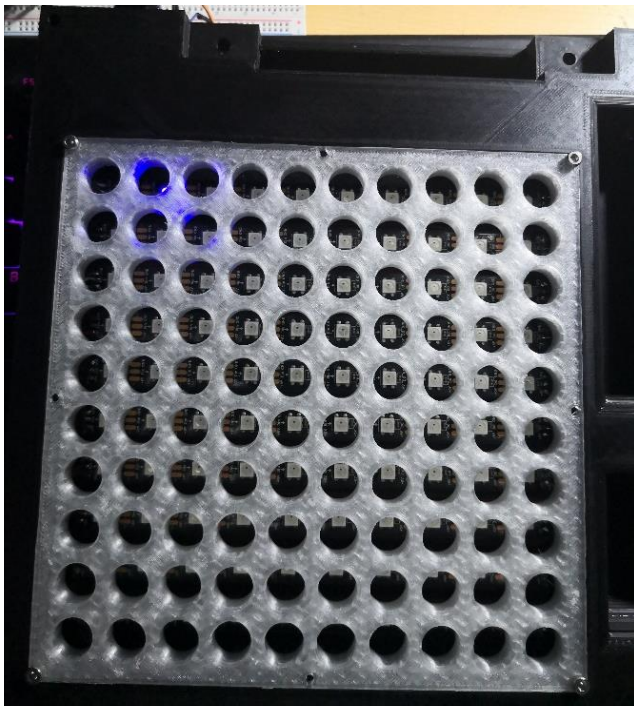
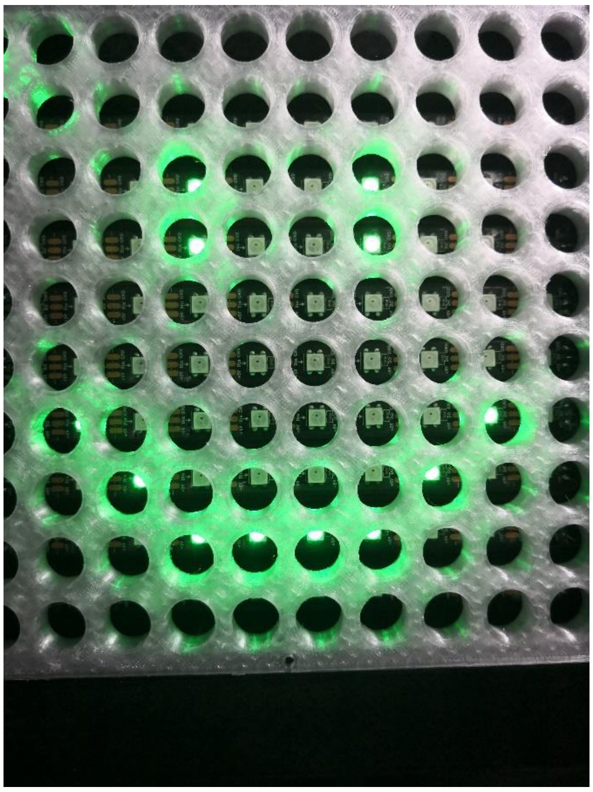
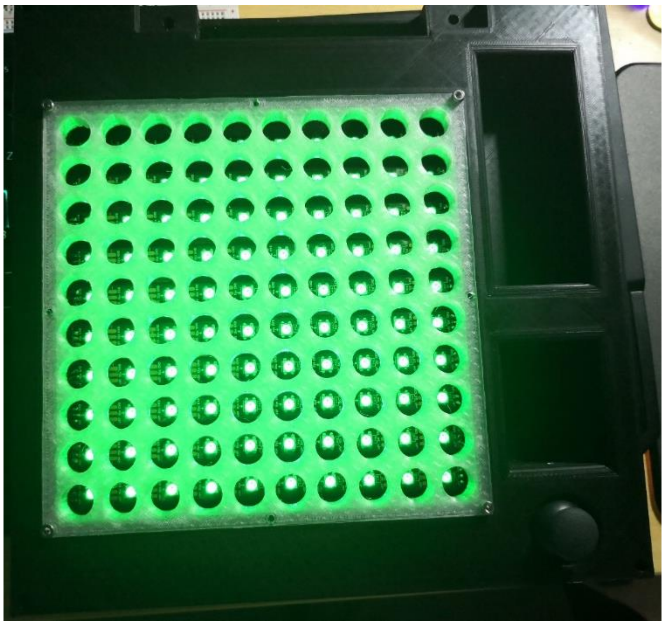
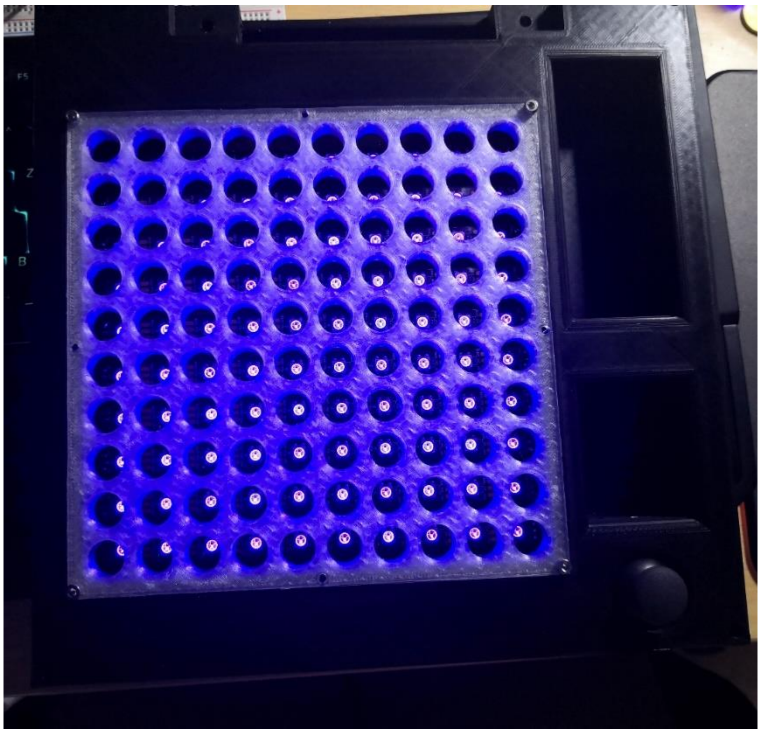
The reed switches became a real problem to the end. The matrix is a lot bigger than the one I had tested the theory of it. I had to put 2 shift registers in serial of SIPO and PISO, so that I had output and input at the same time (I explain the matrix in this chapter). The output was right, but the input received was not the one I was feeding it. This could have many reasons, for example, my code, the fact that the reed switches are so close together, some of them might be broken or a combination of all of the above. I did not have time to fix this, this is why the matrix is laying on top of the board game instead inside of the board. You can find pictures of the matrix working and not working in the "Fotos" folder.

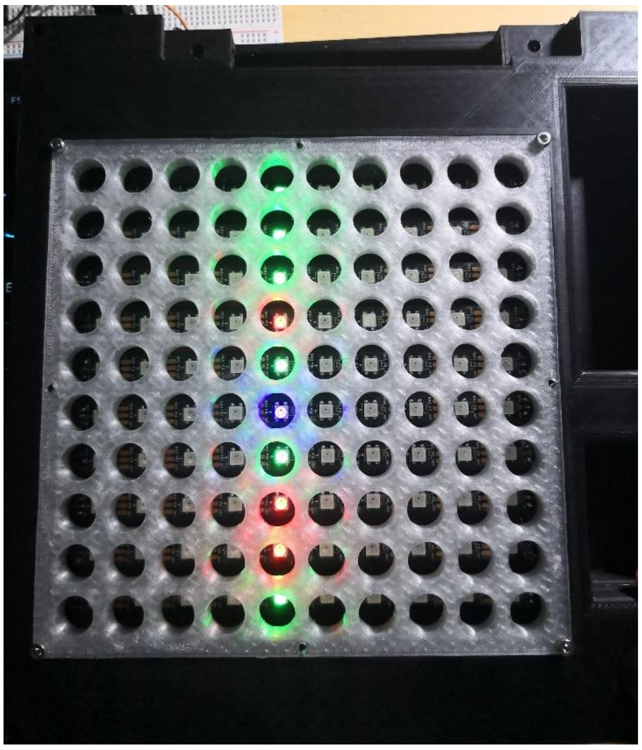
3.2.5 Assembly

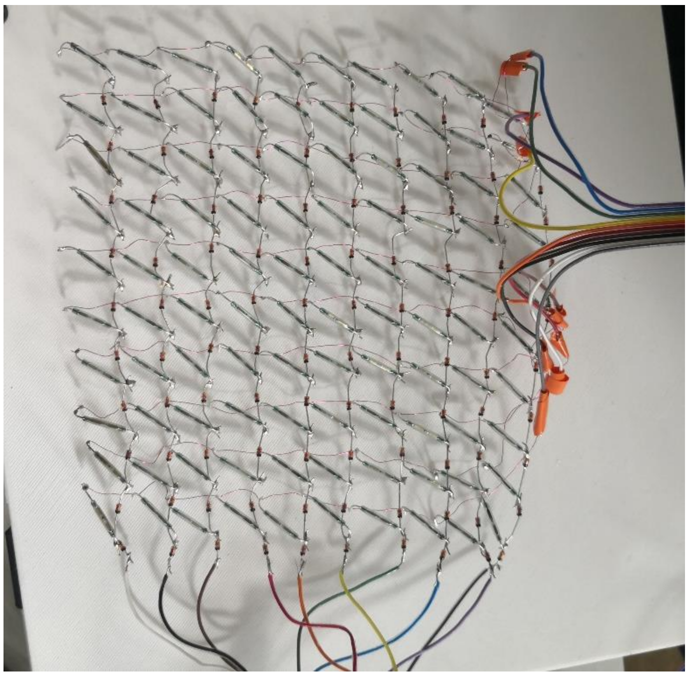
The first real assembly was done when the bottom part had been printed, then I could attach the LEDs to their according spots and soldering the first PCB. When the top cover had been printed, I could attach the rest of the LEDs to the top and make the connection between the 2 parts of the strips.
After that I started soldering the reed switch matrix. In my first attempt, I had no real way to keep them in the right place while soldering, so the dimensions didn't fit the

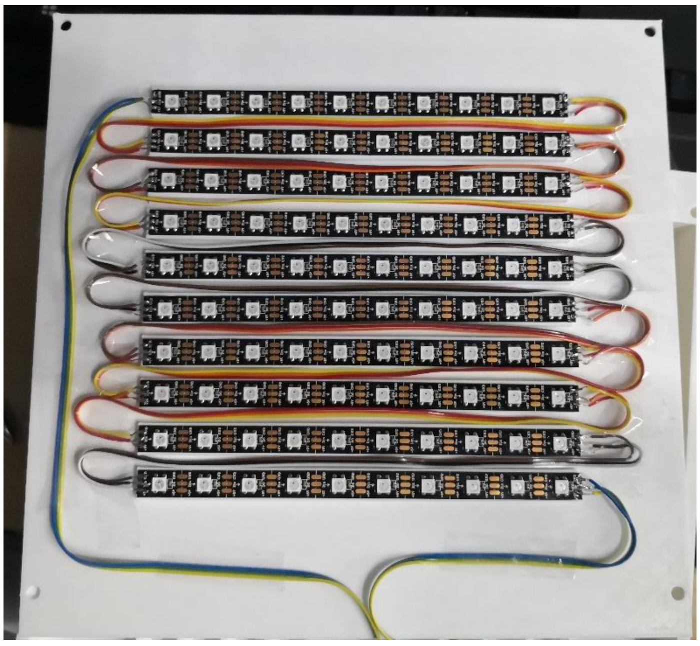
board. That’s why I changed my method to attaching them to a sheet of paper and soldering them through that. You can find a picture of this in the Photos chapter.

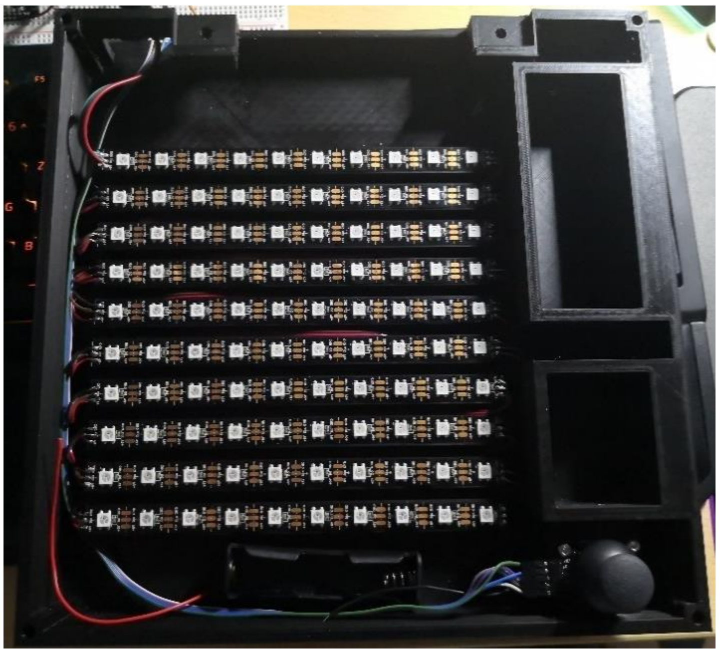
3.2.6 Photos of tests and assembly









3.3 Detailed List of parts

3.3.1 3d Printed parts

These parts were already explained in a previous chapter, but for completion, they have been added here.

3.3.1.1 Top part

3.3.1.2 Top cover

3.3.1.3 Hinges

3.3.1.4 Bottom holes

3.3.1.5 Bottom plate

3.3.1.6 Bottom part

3.3.2 Electronics

3.3.2.1 ESP32

The ESP32 is the heart of the project and handles the board part of the game. This model was preferred to an Arduino, because it is cheaper and has Wi-Fi, which is necessary to have for this project,

3.3.2.2 Joystick

This is used to control everything on the board.

3.3.2.3 Matrix

The playfield is a 10x10 field and the ESP32 does not have enough pins to be able to handle all of it. That’s why these ICs are used.

3.3.2.3.1 74HC595 SIPO

This IC is used to create more outputs for the ESP32.

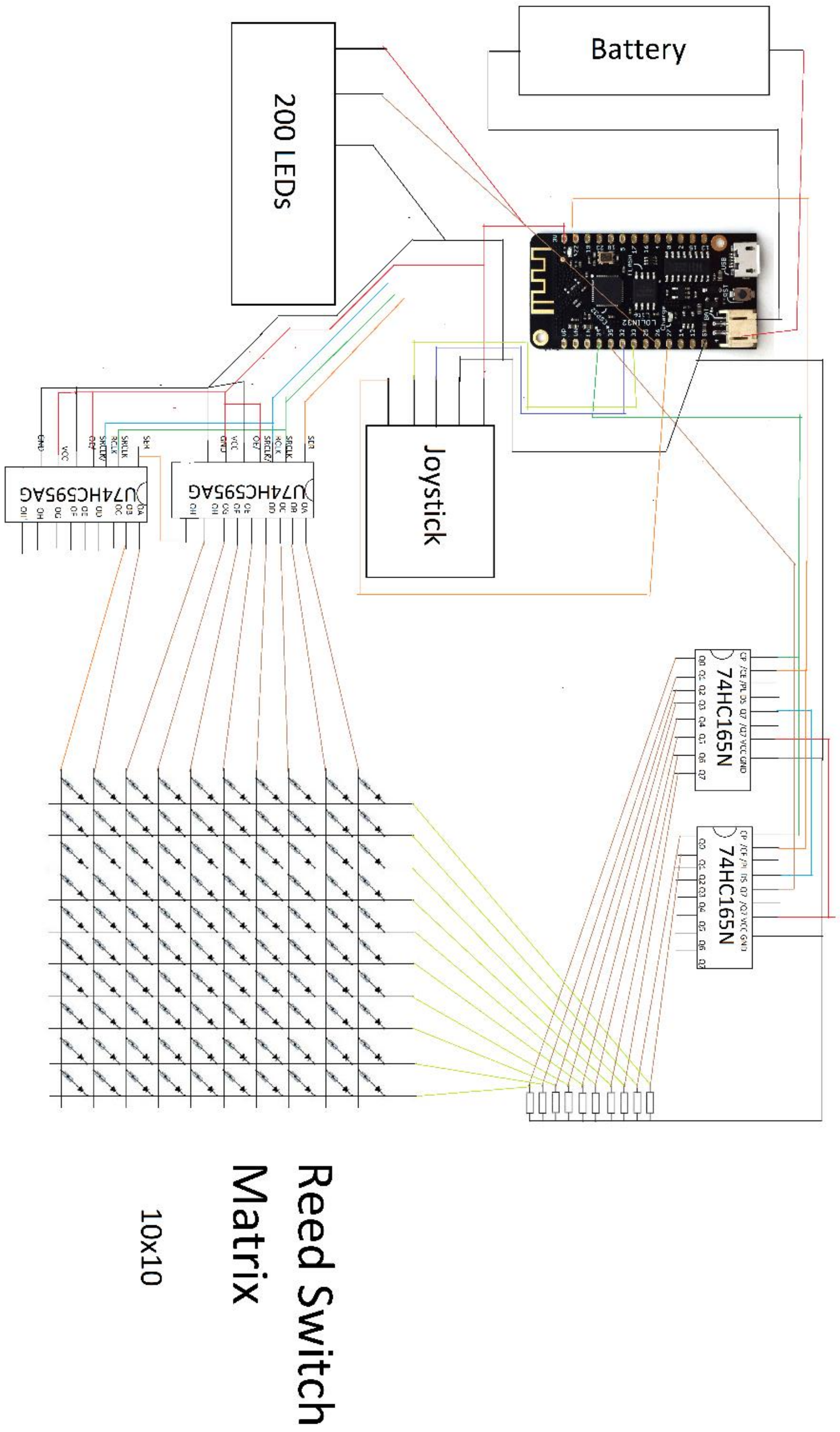
3.3.2.3.2 74HC165 PISO

This IC is used to create more inputs for the ESP32.

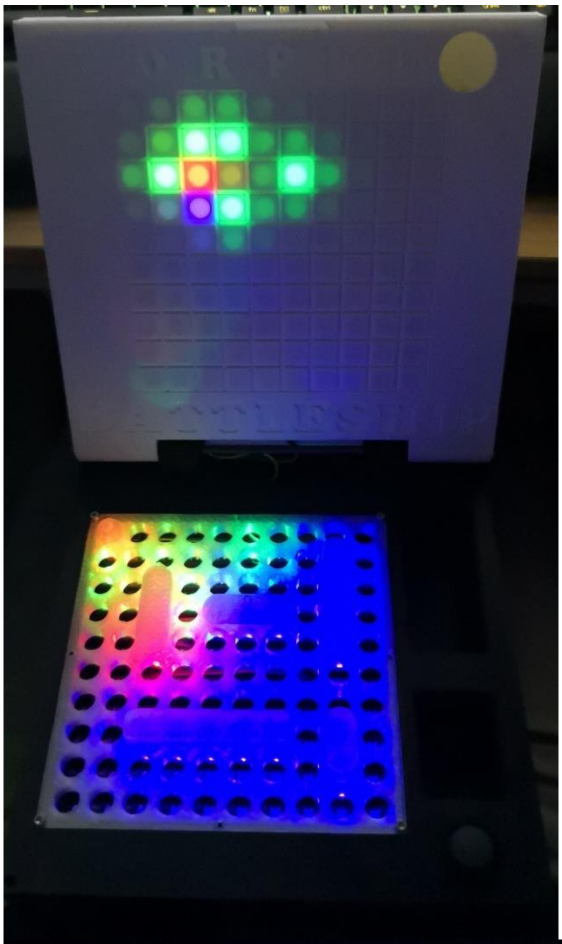
3.3.2.3.3 Reed switches

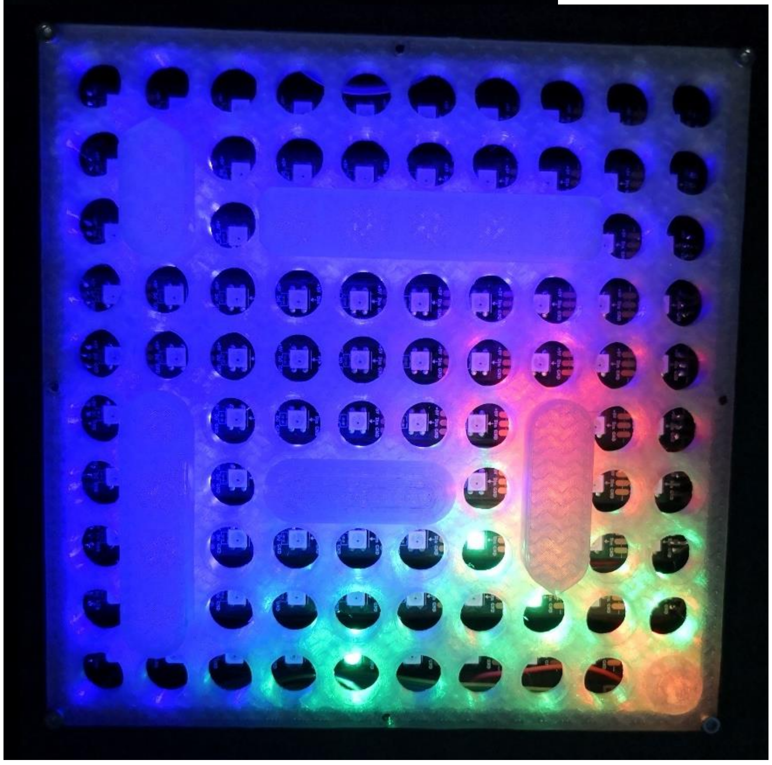
Reed switches are in a sealed glass housing and get activated by a magnetic field. These are used to detect the magnets in the ships.

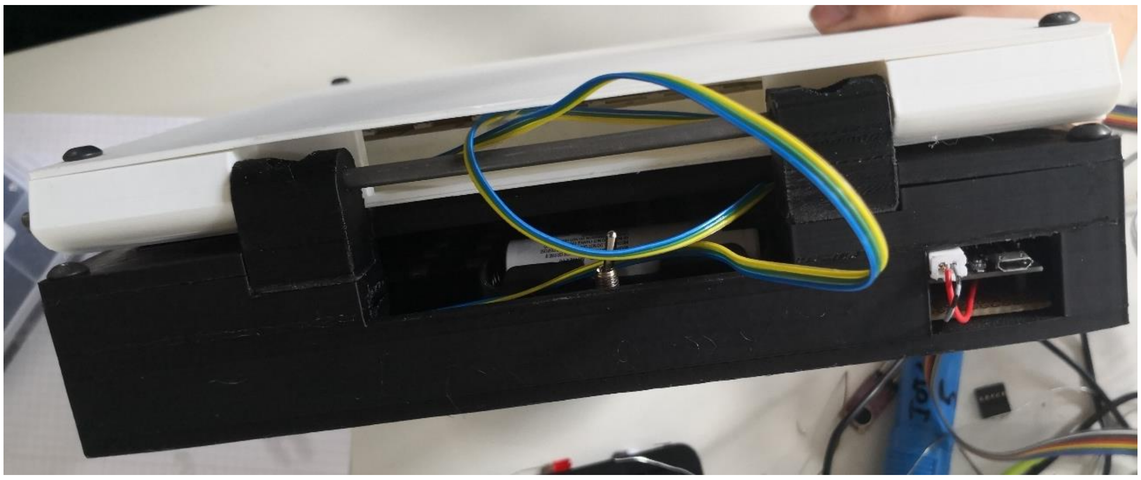
3.4 Electronic circuit schema

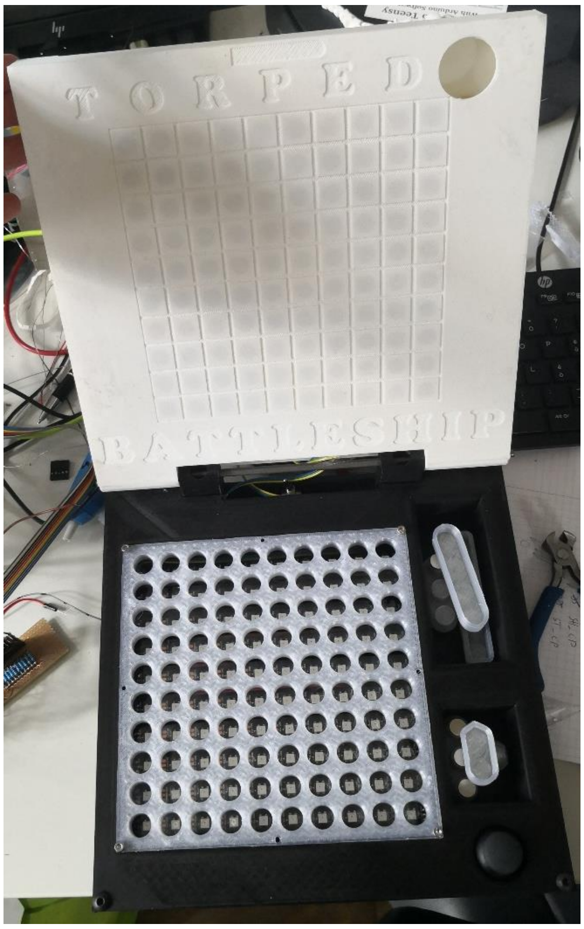


3.5 Pictures of the final product








TORPEDO
BATTLESHIP

4 Executive summary

My boardgame is able to send and receive data via MQTT and change the LEDs to the according colour. It is able to be controlled by the joystick.

However, the Reed switch matrix is not functional due to last minute complications and is outside of the box currently.

The website is functional, except for the fact that it disconnects from the MQTT after the first received message and I couldn’t find the error.

5 Acknowledgements

Special thanks to Yves Fischer, Luka Theisen and Luana Do Vale for helping me all around the project.

6 Declaration of autonomy

I certify herby that I have written the thesis and report independently without the help of other aids than those explicitly stated.

Name:	Charel Feil
Place and Date:	Bissen, Luxembourg 24.06.2022
Declaration:	I certify herby that I have written the thesis and report independently without the help of other aids than those explicitly stated.
Signature:	[illegible]