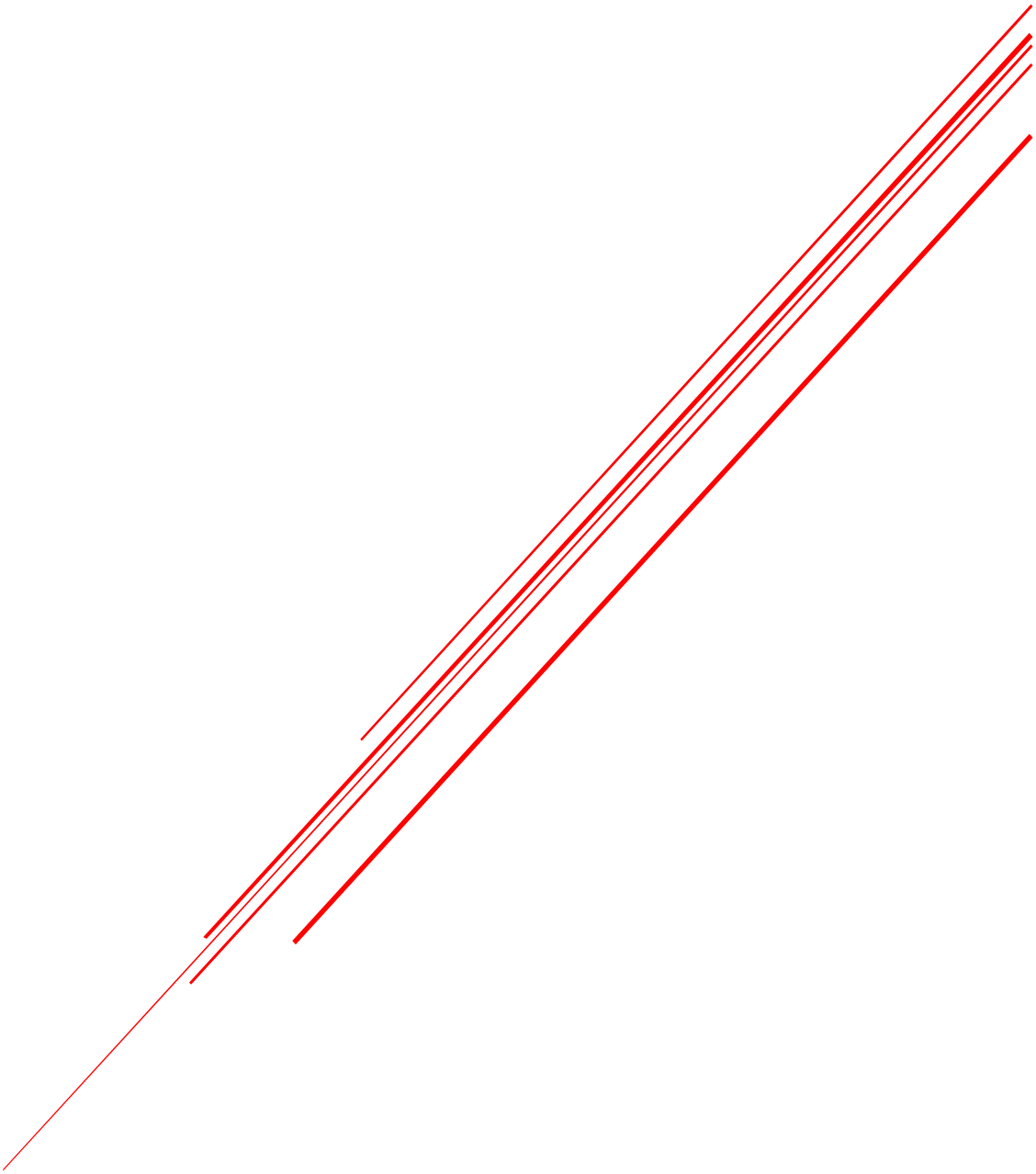
MOTION-CONTROLLED LED MATRIX

New Devices Lab project of A10



Nikita Kuprins & David Scholten

8 December 2024

Contents

1 Introduction

WILL BE ADDED FOR FINAL VERSION

2 Goal and approach

In this chapter, the process towards the final project will be discussed. The first paragraph will be a short explanation of the goal and then the approach to this goal will be described.

2.1 Goal

The goal of the project is mainly for users to be able to have fun with the games that are created in the project, but also to work on their hand-eye coordination. The final product allows users to be able to train their coordination on either or both of their hands and do this while trying to get their high score in games like *Snake* and *Pong*.

When the controller is attached to the screen, the movement of the blocks in *Cubes*, the snake in *Snake* and the pong bat in *Pong* follow gravity. When made safer for children, this could possibly help them understand gravity and movement, but is not the main goal of the product.

2.2 Approach

Firstly, paragraph 2.2.1 will be about the feasibility study. After that, the making of the minimal viable project will be described in paragraph 2.2.2. Finally, the steps towards the final version will be discussed in paragraph 2.2.3.

2.2.1 Feasibility study

In this paragraph, the different components and concepts that were tested before the making of the minimum viable product are discussed. These are the BNO055, 64x32 LED matrix and socket communication, respectively. In the subparagraphs, the boards that were used will also be mentioned.

2.2.1.1 BNO055

The initial idea was to create a game on a monitor or laptop and make it motion controlled with a hand gesture sensor which communicates through a Wi-Fi connection. The feasibility study started off with testing the BNO055 accelerometer and gyroscope functions to test its suitability for the hand gesture sensor. For this is was connected to the LOLIN D1 mini PRO.

With code that returned directions for certain angles, for example “LEFT” if the rotation over the x-axis was -15 or less degrees, the sensor turned out to work well and was not very complicated. It took just over one session to get this working, so the decision was made to continue with this sensor, but try to make the project more complicated in another way.

2.2.1.2 64x32 LED Matrix

It was then decided to make the game on a programmable 64x32 LED matrix instead of a monitor or laptop screen. To get the screen working, the LOLIN Wemos D1 mini, the Arduino MEGA ADK and the Raspberry Pi Pico. After having the same issue with all of these boards, where several parts of the screen were not lighting up correctly (see figure 1) and checking the wires and the code a lot of times, the realization came that the issue might not be in the boards, the wires and/or the code, but in the matrix itself.

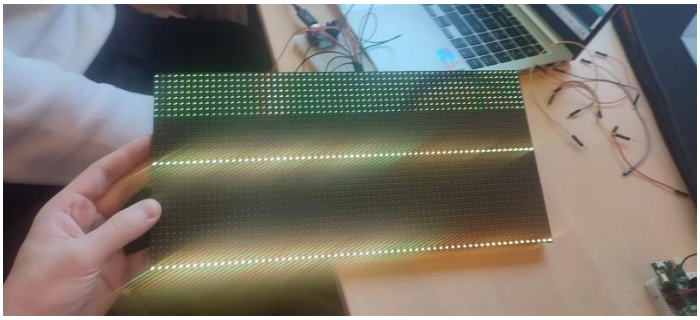


Figure 1: Matrix working incorrectly

After four practical sessions of struggling with the first matrix, a different matrix was tested and worked. This cost more time than necessary, but in the end there was a working matrix with both the Arduino MEGA ADK – which was initially intended to be in the project, but did not have Wi-Fi - and the Raspberry Pi Pico, which also did not have Wi-Fi, but this was easily fixable by using the Raspberry Pi Pico W. As programming this matrix seemed complicated, but not too complicated, the decision was made to continue with this project, combining it with the BNO055.

2.2.1.3 Socket communication

For the project, a persistent Wi-Fi connection between the matrix and the sensor, so the Raspberry Pi Pico and LOLIN D1 mini PRO, was desired. As the Raspberry Pi Pico itself does not have Wi-Fi, a switch was made to the Raspberry Pi Pico W, which does have Wi-Fi. For the persistent communication, socket communication is preferred over communication with HTTP requests, which was explained and used in one of the first practical sessions of the course.

It took some effort to get the socket communication working. Every time a connection was opened on a port, but not closed due to an error or another mistake in the code, the port would not be available anymore. This would result in the board not being able to find and connect to the NDL_24G network anymore, unless a ‘flash_nuke.uf2’ file – which completely resets the entire board to its initial settings – was put onto the board and all the files and libraries would be put back after this. This took a few minutes every time and when it had to be done multiple times per session, this was obviously quite annoying. After a few sessions, this also started to work.

2.2.2 Minimum viable product (MVP)

This paragraph will provide insights into the making and coding of the minimum viable product (which will be abbreviated to MVP from now on). Subparagraph 2.2.2.1 will be about the communication, subparagraph 2.2.2.2 about the games and finally, 2.2.2.3 will contain information on the combination of the two.

2.2.2.1 Communication

An essential part of the project is communication between multiple components. For this, the final version will use a Wi-Fi connection. However, when trying to test the code that was written for the games, it quickly became clear that having to make a connection with the BNO055 every time would take some time and sometimes the issue discussed in subparagraph 2.1.3 could occur. To make sure that the games could be tested without having to make a connection, the games were made to use buttons first (see figure 2).

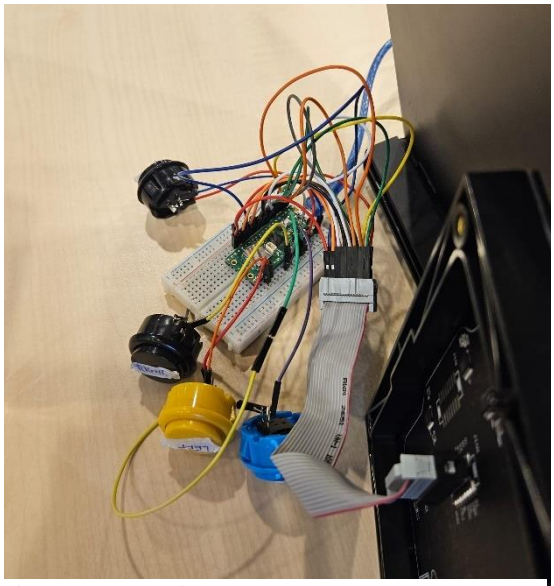


Figure 2: Board with buttons

2.2.2.2 Games

For the MVP, the games that were chosen to recreate are Snake and Pong. This choice was made as there was already some experience in making these games and these games usually already do not need many pixels, so they were logical to make on the matrix.

2.2.3 Final product

After making the MVP work, the next step was to make the games work with the BNO055 sensor and fix the issues that were previously encountered with the connection. After a few tries, the connection issue would only occur sometimes when the code was changed and reuploaded. In the other scenarios it always works, so it was decided to continue with other parts.

As the boards used were not supposed to be connected to laptops in the final product, one of the group members brought a power bank to provide electricity to them and a frame that could be used by the user to hold the screen and to connect the power bank to was constructed. The BNO055 was stuck on the power bank. To make it more interesting, the idea came up to be able to connect the controller to the screen, but also to be able to use it separately. This meant glue or tape were not a great option. Luckily, there were stickers with Velcro (Dutch: klittenband), which made it easy to connect and disconnect the controller to the screen. As the controller would be upside down when connected to the screen, the code for the sensor needed some changes, so it would still give the values needed for the games. After this was done, the software for the final

product had one final refactoring and is finished. **The hardware design might still change in the upcoming sessions.**

3 Electrical and mechanical design

This chapter will be about the hardware of the project, split into three paragraphs. Paragraph 3.1 will be about the electrical design, including wiring diagrams and parts that were used. After that, the mechanical design of the device will be discussed in paragraph 3.2 and finally, in paragraph 3.3, the combination of the two will be shown with a diagram.

3.1 Electrical design

The electrical part of the device exists of two smaller devices, namely the 64 x 32 LED Matrix connected to the Raspberry Pi Pico W and the BNO055 sensor connected to the LOLIN D1 mini PRO 2.0.0. Both will have their own subparagraph in this paragraph.

3.1.1 LED Matrix with Raspberry Pi Pico W

To provide enough processing power and Wi-Fi, it was chosen to use the Raspberry Pi Pico W for the LED matrix. In figure 3, it is shown how the wiring between the microcontroller and the LED matrix HUB75 IN-port is done.

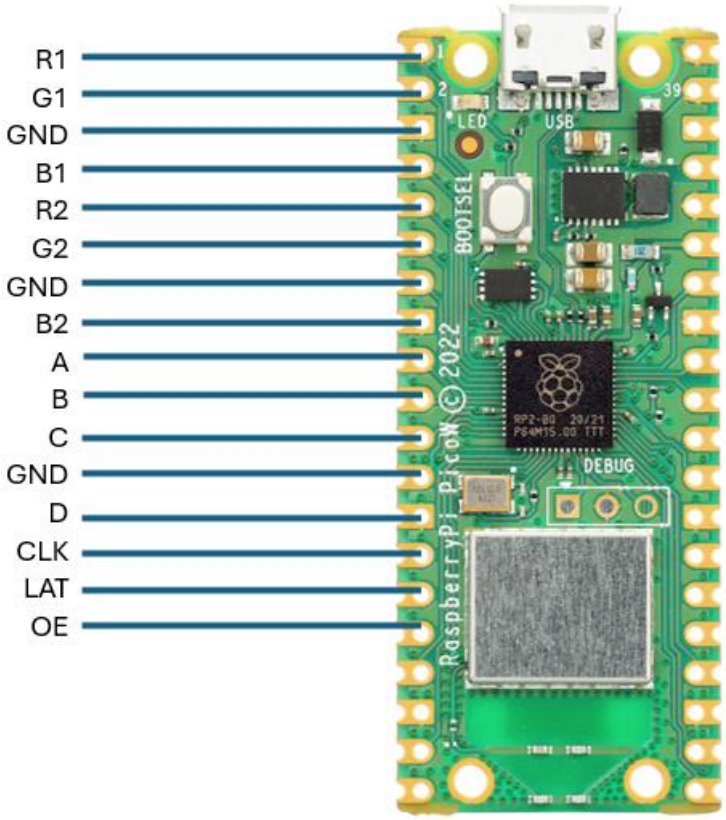


Figure 3: Wiring LED Matrix (Source picture of Raspberry Pi Pico W: Raspberry Pi Ltd, 2024)

3.1.2 BNO055 and LOLIN D1 mini PRO 2.0.0

As the microcontroller connected to the sensor would not need a lot of processing power and memory to be able to extend the project if wanted, the choice was made to

use a PRO version of an ESP8266 board, namely the LOLIN D1 mini PRO 2.0.0. This was then connected to the BNO055 sensor on a breadboard, following the wiring scheme in figure 4.

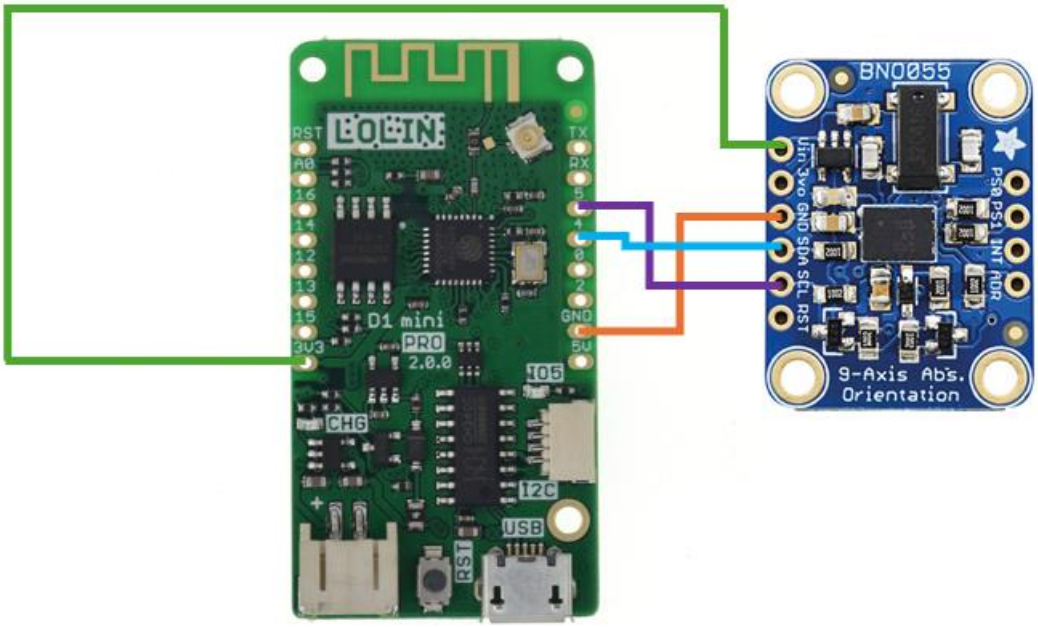


Figure 4: BNO055 wiring. (Source LOLIN Board: WEMOS, n.d.; source BNO055: reichelt elektronik GmbH, n.d.)

3.2 Mechanical design

WILL BE WRITTEN WHEN MECHANICAL DESIGN IS COMPLETELY DONE

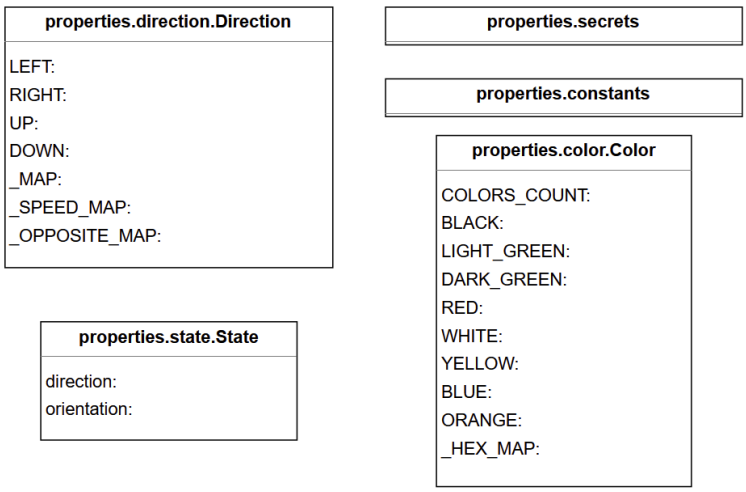
3.3 Total design

SAME AS 3.2

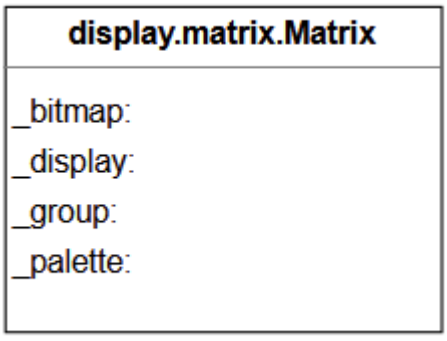
4 Software design

THIS IS A SKETCH, WE WILL WRITE THIS CHAPTER PROPERLY FOR THE FINAL VERSION

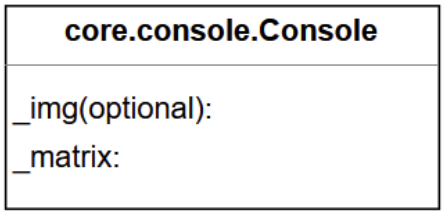
- Before diving into the flow we would like to show the properties that are used in the entire application



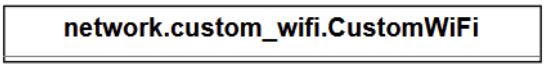
- We use asyncio library to achieve non-blocking flow of the program. The idea is to create two tasks: 1. for the controller(buttons or TCP) 2. for the console games. Then, we gather on these tasks and each of them uses asyncio to not block the other task.
- Everything starts from the code.py module that has 2 main functions. One is used for production with TCP communication to control the console, and the other uses buttons and is intended to use for debugging or development.
- In both cases we first intialise the matrix class

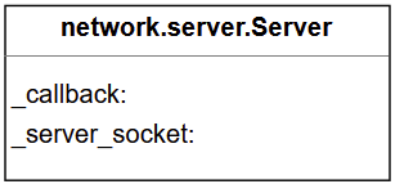


- Then, we intialise the Console and show the logo image. This logo image is needed only in the TCP part, because it takes a few seconds to setup the connection and connect to wifi and during this time we show the logo.

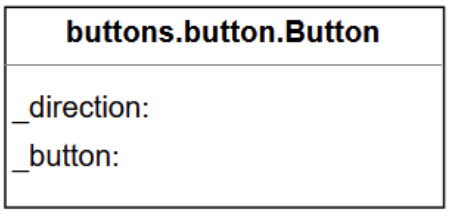


- Now, for the main function with TCP we connect to wifi. If it succeeds, then we create asyncio task for the console run() function, and another asyncio task setups a non-blocking server and in a loop accepts the connection and reads the data. For the incoming data we call the callback function.





For the main function with buttons we create 4 instances of Button in the buttons_controller with LEFT, RIGHT, DOWN, UP directions. Then, we create asyncio task for the console run() function, and another asyncio task that checks in a loop if some buttons were pressed and if so we call the callback function.

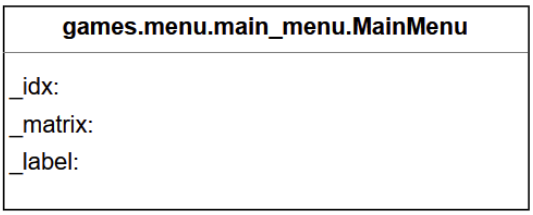


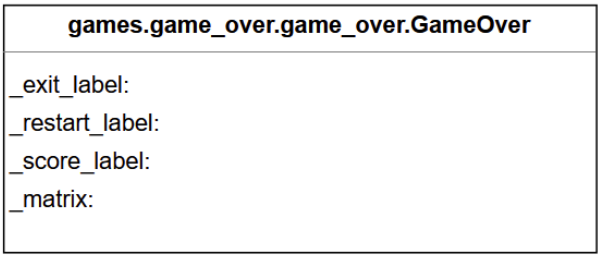
- So we have `await asyncio.gather(server_task, console_task)` or

```
await asyncio.gather(buttons_task, console_task)
```

- In our Console run() we choose what to do depending on the phase. If phase menu, then we show the menu and await for the user to select the game. If phase game running, then we await on the run of game. If await returns info that we lost,

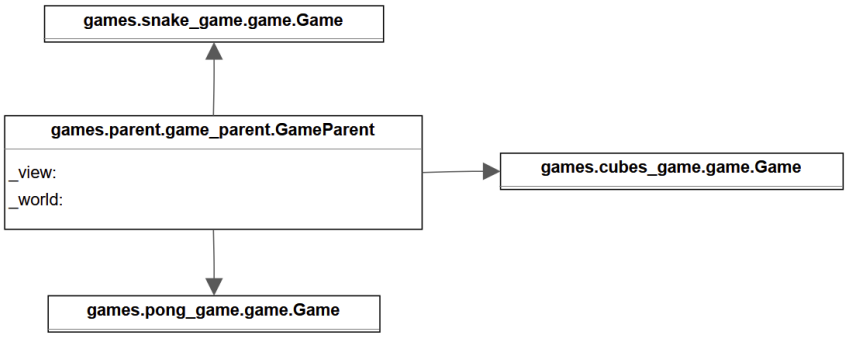
then we show the game over screen, otherwise we go to the menu and repeat again. In the game over screen, we can go to the menu or restart the game.

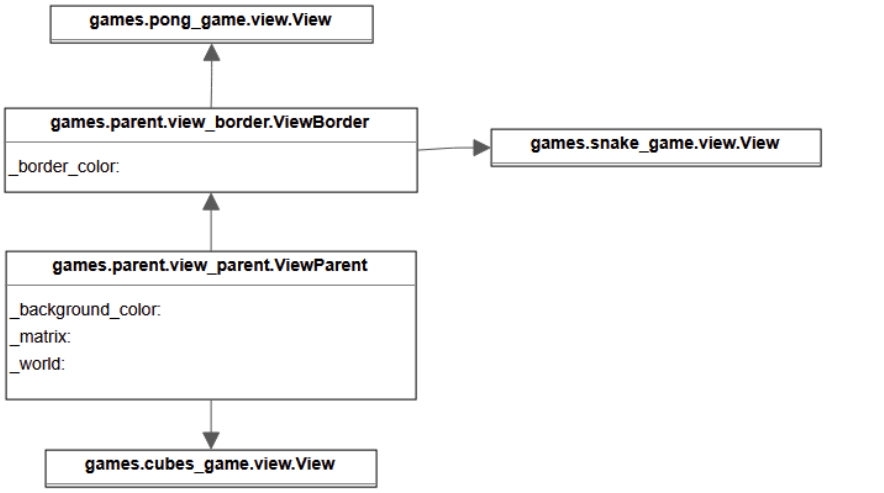


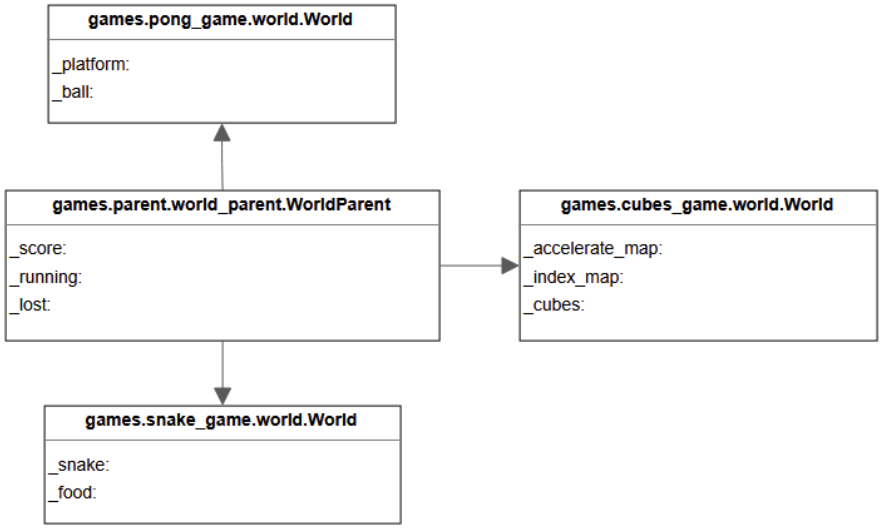


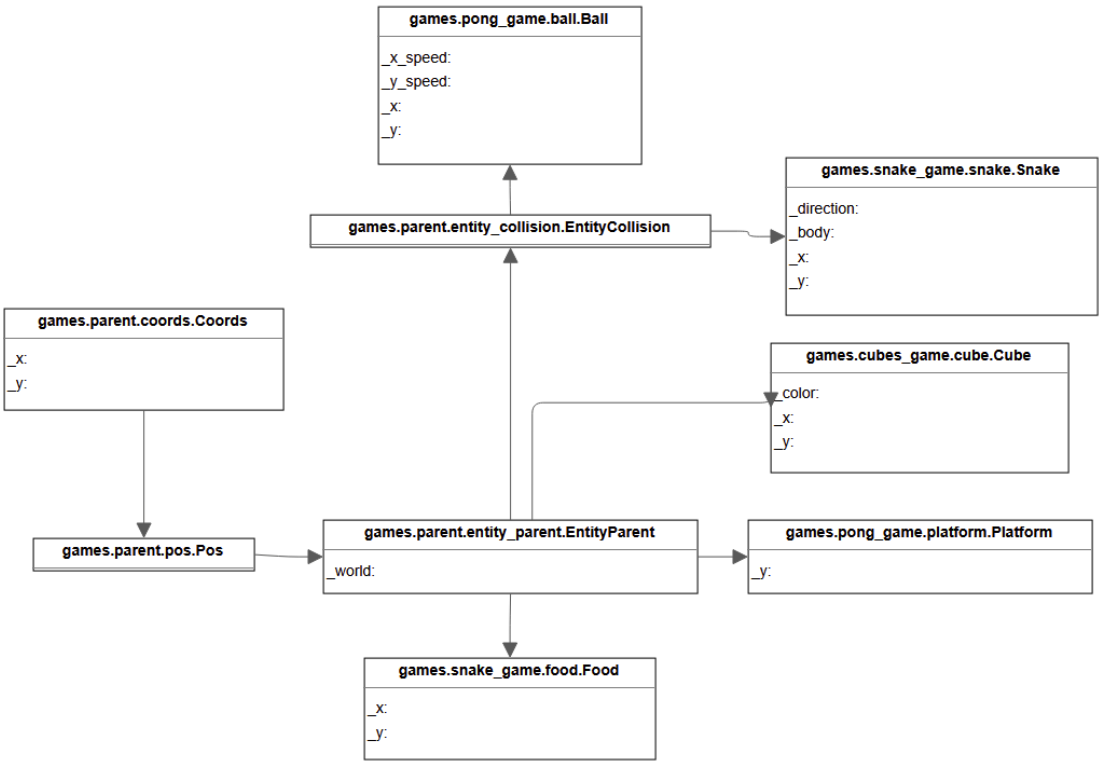
For each game we have a proper oop structure that allows to easily extend the project if needed.

- Game – the controller of the game
- World - the state of the game
- View – the view of the game
- Entity – the entity in the game




games.pong_game.view.View
games.parent.view_border.ViewBorder
_border_color:
games.snake_game.view.View
games.parent.view_parent.ViewParent
_background_color:
_matrix:
_world:
games.cubes_game.view.View


games.pong_game.world.World
_platform:
_ball:
games.parent.world_parent.WorldParent
_score:
_running:
_lost:
games.cubes_game.world.World
_accelerate_map:
_index_map:
_cubes:
games.snake_game.world.World
_snake:
_food:



Used examples:

https://learn.adafruit.com/rgb-led-matrices-matrix-panels-with-circuitpython/example-simple-two-line-text-scroller

https://learn.adafruit.com/adafruit-matrixportal-m4

Used libraries:

Asyncio

Raspberry Pi Pico W Adafruit CircuitPython library

5 Process Report

In this chapter we will reflect on our process (so far) and what we could have done better in this process. For this, we will first talk about the technical implementation, then the learning and implementation process will be discussed and the final paragraph will be about the sustainability of the process.

5.1 Technical implementation

Our current project does what it is supposed to do and we think it looks alright. We focused a lot on the code quality and the functionality of the games. The wiring is sometimes a bit messy, but with the amount and the placing of wires that is needed for the matrix, we think this was not really avoidable. We are happy with the code and how the code works. The games we made look nice (for the matrix we have) and work as intended. The sensor also sends the correct z and y value to the matrix, even if it is upside down like it is in the final version of the project. The Wi-Fi connection and the complexity of the code does create some delay between the input and the output. This is something that might have been better with either Bluetooth or a wired connection. For sustainability and time purposes and not knowing if this would actually improve the performance, we decided not to order the Raspberry Pi Pico with Bluetooth. If we were to do it again, we could have tried this in the feasibility study, but because of the broken screen, there was no time for this.

5.2 Learning and implementation process

We learned a lot from the project, especially that a project that seemed quite doable at the beginning of the course becomes quite hard if you get multiple setbacks. When we started out the feasibility study with the BNO055 sensor, we realized our original plan at the start of the course was not complicated enough. This was not an issue, as we knew part of the feasibility study was that you could change, simplify or – in our case – complicate the project if necessary. We made the decision to involve an LED matrix and make games for this, which can be controlled by the BNO055 sensor. Coding the games on a matrix took a lot more time and debugging than programming these games normally for a laptop, phone etc., also because it was our first time using CircuitPython. CircuitPython misses some libraries and functions that would have been useful, for example enumerations and NumPy, but we think we do now have a lot of good quality code, which is also extendable. For safety, we uploaded our code to GitLab often. We also chose to include more of a demonstration of how our code works with the blocks-falling animation and made a menu to switch between the games and this.

We are happy with the games we have, but focusing on making three games took a lot of time, which we could also have used to incorporate different sensors or functionalities and make one big game. This would have been a risk as we would not be able to test the

feasibility of all the elements and we might have ended up with an end project that would not work. However, it might have also increased the complexity of our project. Complexity was an issue in our project all the time. We wanted to start off with a simple project and, if possible, make it more complicated along the way. We learned that this is a good way to attack the problem, but might also take away a bit of ambition. Our project was complicated to make for us, but might not look that way and we think we might be able to do something more unique in a next project. We could have done better on the creativity, but we are happy we were able to recreate the idea we had in mind before starting on the MVP.

5.3 Sustainability

In this paragraph, we will explain our decisions made on sustainability in the process. We will first look at the hardware and mechanics choices in 5.3.1 and then discuss the sustainability in the software in paragraph 5.3.2. In the end we will also discuss possible improvements in 5.3.3.

5.3.1 Hardware and mechanics

From the start of the project, we tried to use as much elements as possible from the inventory list to avoid having unnecessary costs, shipping and materials. We did order one sensor, namely a gyroscope, which we did not end up using as is turned out the BNO055 already had a built-in gyroscope. This was obviously not a great decision in terms of sustainability. For making the frame around the matrix, we used materials from the Totem building boxes instead of trying using the 3D printer, which could have wasted quite a lot of plastic. To make the frame, we had to saw some pieces of plastic into smaller pieces, but the frame can be disassembled after the project and all materials could be reused. This would probably not have been the case if we 3D printed it.

Some components, like the breadboard and the Velcro, had a sticker side and are stuck onto other components with that side. As we removed the covers from the sticker side, these components can probably not be reused, or only by adding additional glue. However, to get to this point the stickers would have to be removed from the components they are stuck on to. We think this is not too hard with enough force and maybe a sharp object, but can cause some issues for reusability.

We also tried to choose boards that we thought have enough data and processing power for our project with the Raspberry Pi Pico W and the LOLIN D1 mini PRO instead of using a full Raspberry Pi and an ESP32 board, which have better processors, but will also probably use more energy.

5.3.2 Software

In the software we added a few things to help with energy consumption and hence sustainability. For example, when someone loses with Snake or Pong, they get sent to a

screen asking to play again, instead of just restarting the game which has a much higher complexity than showing a menu over and over. This way, we save energy when players stop playing or do not pay attention. The same goes for Cubes, where we make sure that the game stops if the cubes are stationary for ten seconds. This way there is lower power consumption.

In the controller, the BNO055 automatically will send signals less frequently and go into a low power mode (Mischianti, 2023) if no movement was detected for more than 5 seconds and once it notices there is movement again, it will start sending signals more frequently again. This could be changed by setting a standard mode in the code for the BNO055, but we decided to keep it this way as it also helps with sustainability and power usage.

5.3.3 Improvements

Although we tried to make our project sustainable, there are a few areas where we could have improved on it. Ordering a gyroscope even though we already had a sensor with a gyroscope function was a waste of money and materials. In hindsight, we did not use any of the other functions of the BNO055 sensor so we could have used this gyroscope instead of the BNO055.

Also, we are not completely sure whether the first LED matrix was already broken or that we have accidentally broken it in some way. We think we did not break it, but it is always good to be careful when connecting wires and applying pressure on things when you have to attach or remove something from it. We tried to be even more careful on the second screen, which still works.

Finally, if we were to do this again, we could do more research into which boards use the least amount of power and materials for the end goal of the project. For this project we guessed that the boards we used would be good enough for our project, but not too good, but did not do a lot of research into this. Maybe we could even have used some boards that use even less power, which would help with the sustainability.

6 User documentation

There are two possible ways to use the device. You can either use the controller as a real controller or connect the controller to the screen and use the screen both as a controller and as a screen. You can even switch between the two in the middle of the session. In this chapter, both ways will be explained.

6.1 Setup

1. Check if there are no loose wires. If there are any loose wires, do NOT plug in the device.
2. Check if both power banks are charged.
3. Connect the microcontroller on the breadboard which is stuck on a power bank to that power bank.
4. Connect the other microcontroller on the other breadboard to the other power bank.
5. Plug in the LED Matrix into a power socket.

6.2 Controller not connected to screen

The LED matrix should be turned on now. You should see a menu showing the name of a game and some arrows. The next steps are:

1. Hold the controller in your hand, with the USB cable towards yourself and hold your hand in a horizontal position.
2. Raise the top of the controller to go to the next game (up). Lower the top of the controller to go back (down).
3. When you have found the game you want to play, tilt your hand to the right to start the game.
4. Snake: try to eat the apple without hitting the walls or your snake in the process. To change directions, these steps can be taken:
    - Move up; raise the top of the controller.
    - Move down; lower the top of the controller.
    - Move right; tilt your hand towards the right.
    - Move left; tilt your hand towards the left.

    You can only make 90 degrees turns. The game is over when you hit your snake or the walls.

    Pong: make sure the ball does not hit the wall on the left side of the screen by moving the pong bat up and down. This can be done in the same way as in Snake. If the ball hits the left wall, the game is over.

Cubes: Move your hand in any way and see how the cubes fall. They should move towards the place you are pointing the controller to. The game stops if you stop moving.

5. When the game is over, you enter a menu. Follow the instructions here.
6. If you are done playing, unplug the microcontrollers and the LED Matrix to save power.

6.3 Controller connected to screen

Use the Velcro on the back of the LED matrix and power bank to connect the screen to the matrix. Test if they are connected properly. Hold the screen horizontally. The next steps will be almost the same as in paragraph 6.2, where the only difference is that you have to hold the handles of the screen now, and movement is based on gravity. So, if you want to move left, lower the left side of the screen, if you want to move down, lower the bottom of the screen and so on.

Bibliography

Mischianti, R. (2023, January 9). *BNO055: power modes, accelerometer and motion interrupt – 4*. Retrieved from Renzo Mischianti: https://mischianti.org/bno055-power-modes-accelerometer-and-motion-interrupt-4/

Raspberry Pi Ltd. (2024, October 15). Raspberry Pi Pico W Datasheet.

reichelt elektronik GmbH. (n.d.). *Ontwikkelaarsborden - Sensor met verschillende functies, BNO055*. Retrieved from reichelt: https://www.reichelt.com/nl/nl/shop/product/ontwikkelaarsborden_-_sensor_met_verschillende_functies_bno055-235479?country=nl&CCTYPE=private&LANGUAGE=nl

WEMOS. (n.d.). *D1 mini Pro*. Retrieved from WEMOS: https://www.wemos.cc/en/latest/d1/d1_mini_pro.html