ORLÉANS

UNIVERSITE D'ORLEANS

Notice

Project Tetris : TETRIS³

Tutor of the project :

Mr. WEBER Rodolphe (Polytech)

Contents

Contents **1**

1 Abstract **3**

2 Introduction **4**

2.1 Context 4

2.2 The project's goals 5

3 The controller **8**

3.1 The design 8

3.1.1 The modelling 8

3.1.2 The wire management 9

3.1.3 The light effect 10

3.2 The electronic part 11

3.2.1 The LED strip 12

3.2.2 The battery 13

3.2.3 The MPU6050 15

3.2.3.1 The accelerometer 15

3.2.3.2 The gyroscope 17

3.2.4 The Bluetooth 19

4 The arcade terminal **20**

4.1 The Tetris game 21

4.2 The controls 23

5 Conclusion **25**

5.1 General conclusion 25

5.2 Personal conclusion 25

List of Figures **26**

A Appendix of the controller **28**

A.1 Assembly instructions 28

A.2 Drawing of base_1 29

A.3 Drawing of base_2 30

A.4 Drawing of poteau1 31

A.5 Drawing of poteau3 32

A.6 Drawing of poteaux24 33

A.7 Drawing of toit1 34

A.8 Drawing of toit2 35

A.9 Drawing of toit3 36

A.10 Detailed schematic of the electronic part 37

B Appendix of the arcade terminal **38**

B.1 Command to launch the game on the Raspberry 38

B.1.1 To connect to the HC-05 38

B.1.2 To run the game 38

B.2 Github of the project 38

1.Abstract

The main subject of this project is to create a new way to play a Tetris game. Indeed, currently everybody who plays a Tetris game uses a joystick or a keyboard with arrow to interact with the game.

So, the goal of this project is to be able to play Tetris while using a remote control that would be inclined or moved (to the right or left) according to how we want to interact the Tetris block.

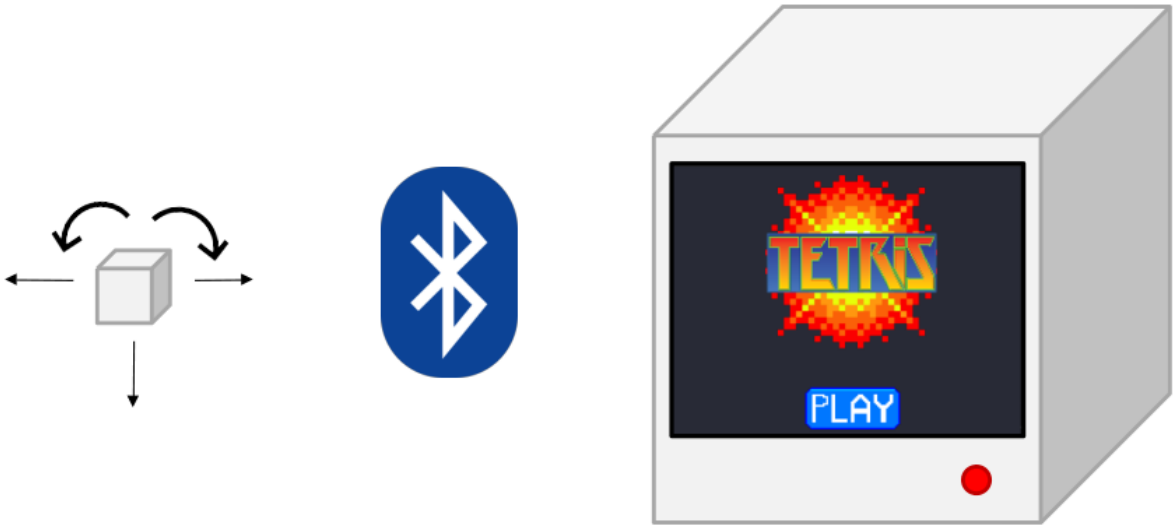


Figure 1.1: Diagram of the project

2.Introduction

2.1 Context

Nowadays, the world of gaming is constantly evolving. More and more games are improved to become more like the real life (haptic devices, photorealism, ...).
Nevertheless, the old games like Pac Man, Space Invaders or Tetris attract always the generation which has grow up.
To make new gamers want to play the old games, some people create a new version of these games. The gameplay will never change a lot but sometimes the look or maybe a new mode.
It is with this idea that this project has been designed.

Indeed, with Tetris^3, the game doesn't change but it's a new way to play to this famous game.
More technically, this new way is resumed by the control of Tetris with a little cube connected with Bluetooth to a Raspberry linked to a screen and play a Tetris game.
The little cube is like a wiimote for the Nintendo console. To move the Tetris block on the screen on the right, the little cube has to be moved on the right too (it's the same way with all the other movements).

2.2 The project's goals

In order to define the project and to see more clearly the final objective, a global diagram has been established :

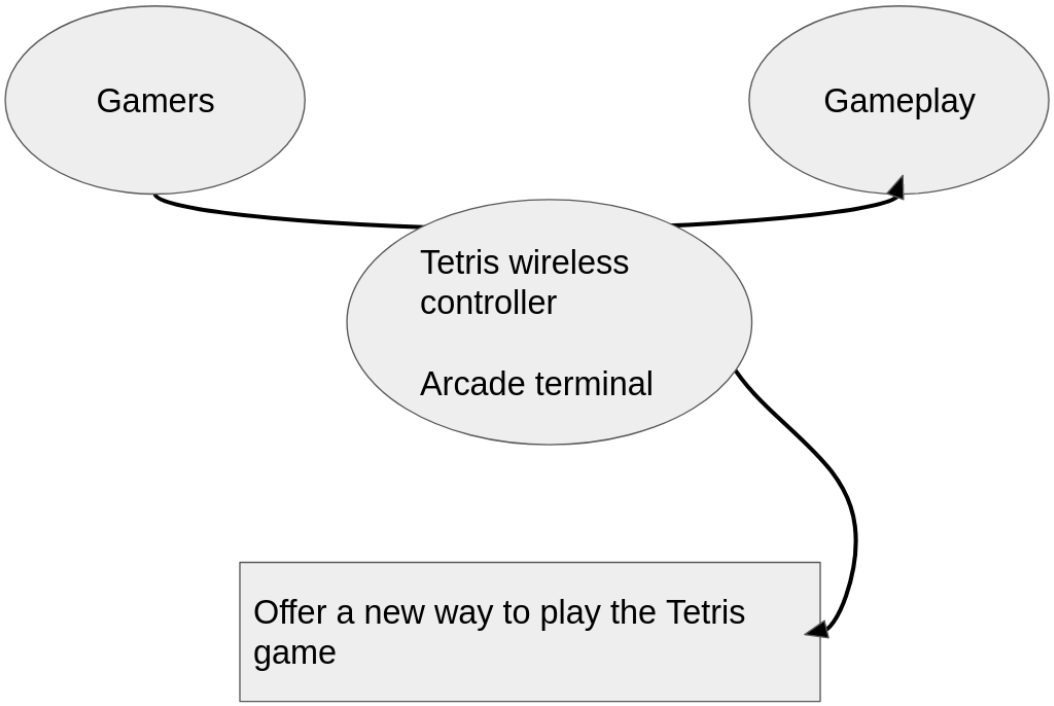


Figure 2.1: Header beast diagram

After defined the bases of the project, the use functions have to be listed with their constraints :

Id	**Functions**	**Details**	**Criteria**	**Solutions**
FP1	User can play Tetris			
FP1.1		A user-friendly interface	Usable by a 3 years old children	

FP2	Having a cube that allows the Tetris to be controlled remotely by movements			
FP2.1		Wireless control cube	Lenght : 2 m	Bluetooth
FP2.2		Rechargeable	Time of use : 1h Recharge time : 30min	
FP2.3		Easy to use	Usable by a 3 years old children	
FP2.4		Be fall resistant	Height : 2m	
FP2.5		Visually attractive		LED and mirrors
FP3	Play Tetris on an arcade terminal			
FP3.1		Good quality		Screen : 17 inches
FP3.2		Interact with the game	Lighting effect	

FC1	Be intuitive		Simple control	
FC2	Python program must be flexible		Possibility to put all of the instructions we want	
FC3	Arcade machine works with main power		230V	
FC4	Good quality of video an sounds		Screen and speakers	
FC5	Be ergonomic		Weight $<$ 500g	

The global operation of the project looks like the following diagram :

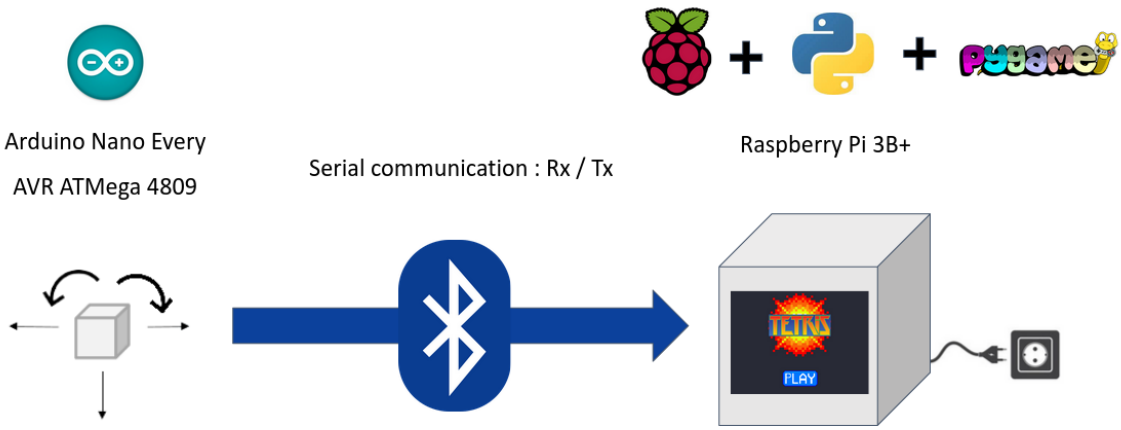


Figure 2.2: Diagram of the global operation

To resume, the goal of the project is to realise a wireless controller which permit to control a Tetris on an arcade terminal with the movement of the gamer.

So, in the first instance, the creation and the programming of the controller will be exposed.

After that, the design of the arcade terminal will be explain.

3.The controller

3.1 The design

3.1.1 The modelling

So, the appearance of the controller is a cube of 10x10x10cm (indeed, the 7x7x7cm wasn't enough to put all the light system and the electronic part inside the cube).

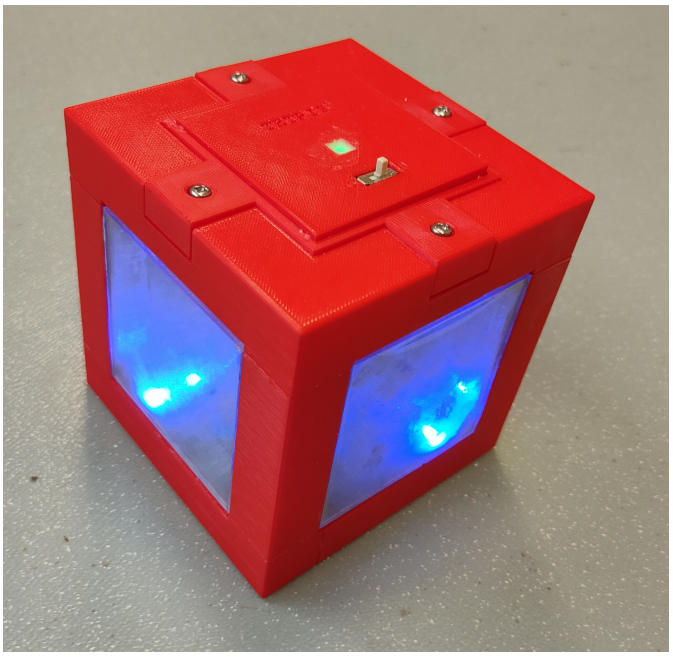


Figure 3.1: Photo of the controller

The cube was printed with 3D printer and is composed by PLA (173,09g). It is divided in nine parts (two for the base of the cube, four for the poles and three for the roof).

3.1.2 The wire management

All of these pieces were made with SolidWorks (CAO software) :

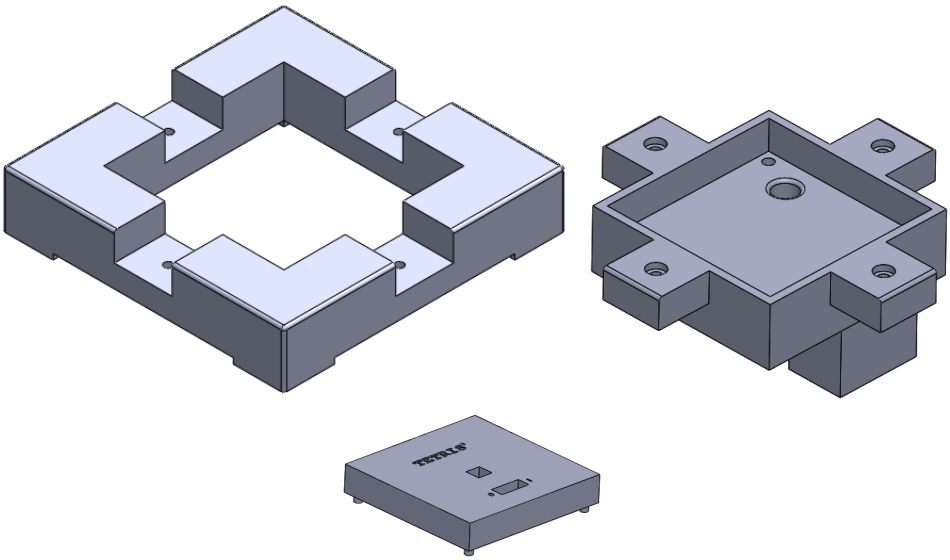


Figure 3.2: Model of the tops of the controller

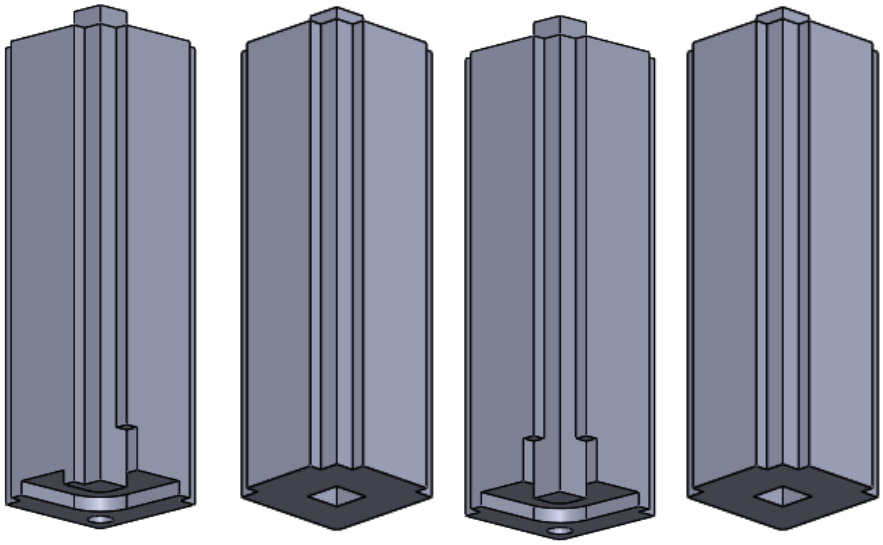


Figure 3.3: Model of the posts of the controller

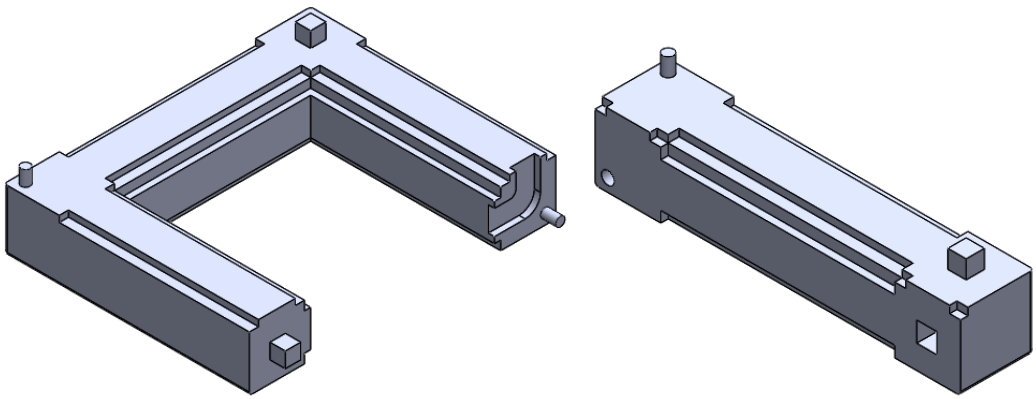


Figure 3.4: Model of the bases of the controller

The assembly instructions and the drawings of all the pieces of the controller are detailed on the appendix A.

3.1.3 The light effect

The light effect on each face is an infinite effect.

The necessary equipment to create this infinite effect is one mirror, one half reflecting glass and obviously some LEDs. If these LEDs are positioned between the mirror and the half reflect, all the light will be reflected on the mirror but that will not be the same case for the half reflect glass. It let pass only the half light and reflect the other half and it's this phenomenon which create the infinite effect.

3.2 The electronic part

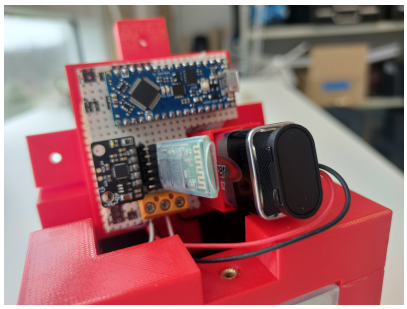


Figure 3.5: Photo of the electronic part

The electronic part inside the cube is divided in four parts. All of these parts are controlled by an Arduino Nano Every which is composed by an AVR ATMega 4809 (the microcontroller). This type of board was chosen for its simplicity of programming.

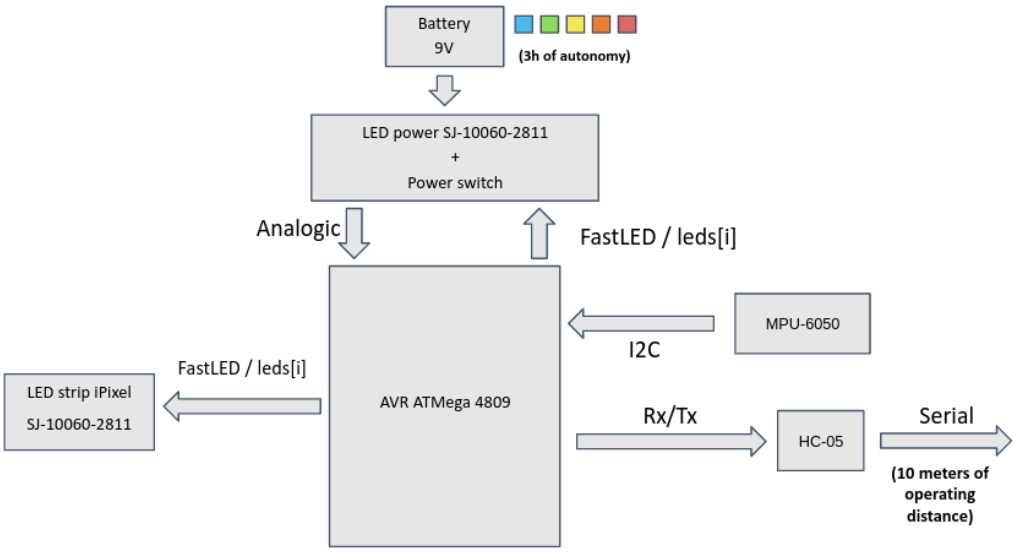


Figure 3.6: Schematic of the electronic part

A more detailed schematic of the electronic part can be found in appendix A.10.

3.2.1 The LED strip

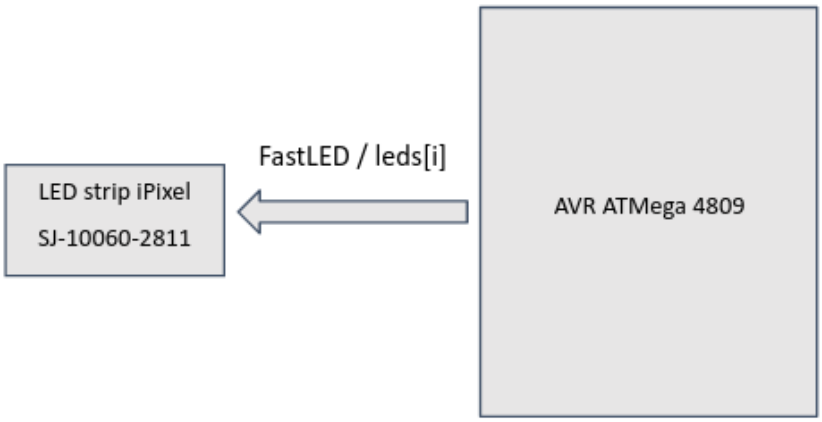


Figure 3.7: Schematic of the LED strip part

In this part, there is the LED strip (composed by 60 RGB LEDs), one resistance of 330ohm and the Arduino. The LED strip has 3 pins, one for the 5V, one for the ground and a last one to program the LED strip. Indeed, this LED strip is addressable so it's possible to choose which LED will turn on and with which color by using FastLED (it's an Arduino librairy which works with this LED strip). To not damage the LED strip, one resistant must be connect between the pin of the Arduino and the pin which permit to program the LED strip.

To create a movement of rotation with the LED on every face of the cube, it's necessarily to make a loop between 12 LEDs and at each "i" loop, the LED "i-1" is turned off and the LED "i" is turned on. When the loop was made 12 times, the patern is restarted.

3.2.2 The battery

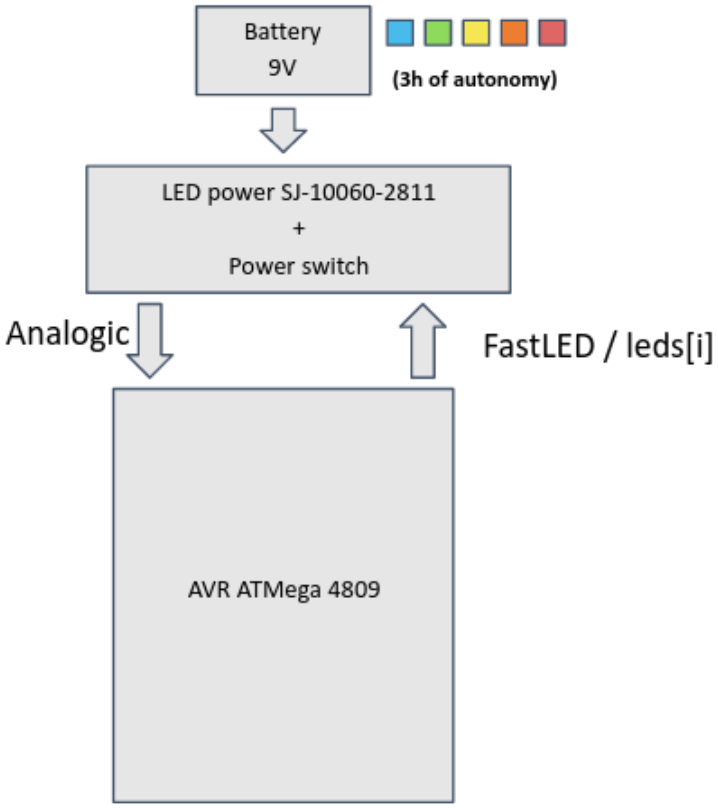


Figure 3.8: Schematic of the battery part

The controller (so the Arduino) is powered by a 9V battery. To find out how much battery is left before the Arduino is under powered, it's useful to create a tension divider bridge and connect it to one analogic pin to the Arduino. Indeed, the analogic pins of the Arduino Nano Every can support only 5V but no more.

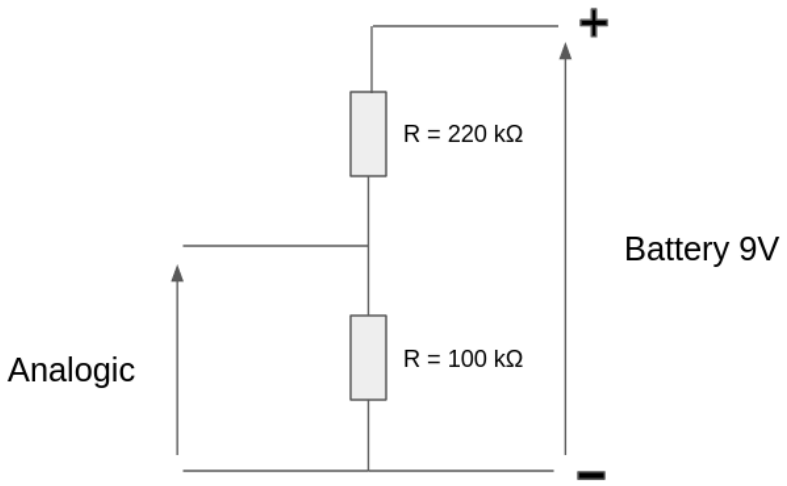


Figure 3.9: Schematic of the tension dividing bridge

After that, with a program which calculates the pourcentage of the battery on the Arduino and with the use of one new RGB LED, the user will know how much time the controller will continue to be powered. This LED will be blue if the pourcentage of the battery is between 100% and 80%, green between 80% and 60%, yellow between 60% and 40%, orange between 40% and 20%, and finally red under 20%.

3.2.3 The MPU6050

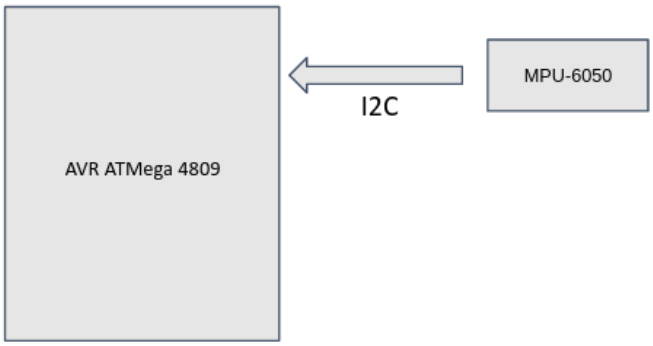


Figure 3.10: Schematic of the MPU6050 part

The MPU6050 is a component with an accelerometer 3 axis and a gyroscope 3 axis.

The accelerometer will be reserved for the translation movements of the tetris block and the gyroscope for the rotation.

But, without adjustment, the accelerometer and the gyroscope can't be used.

3.2.3.1 The accelerometer

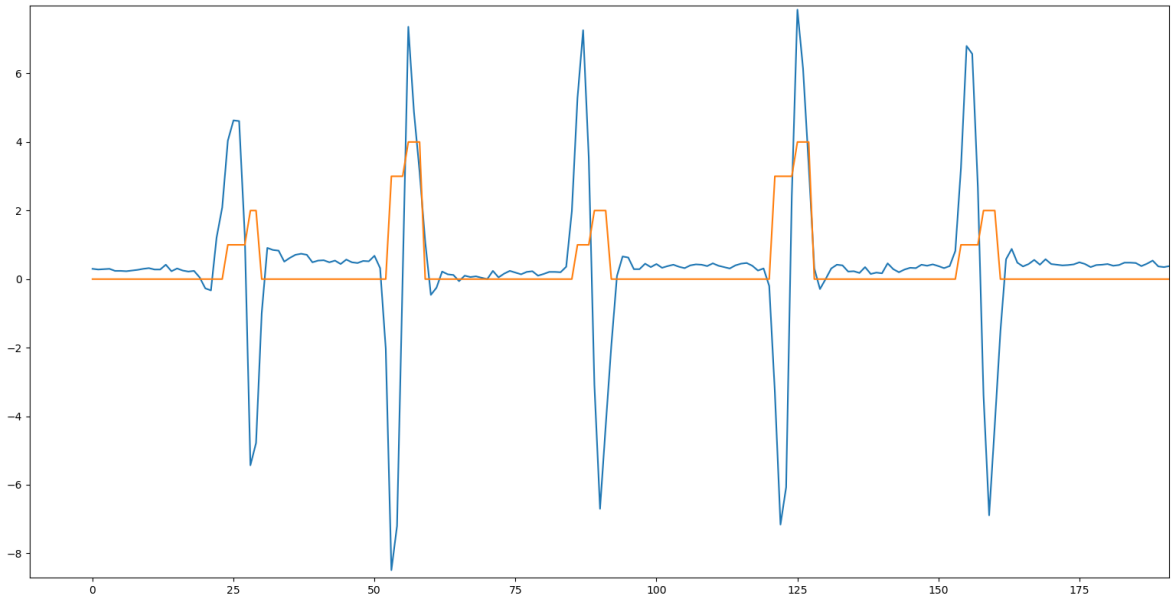


Figure 3.11: Graphic of the evolution of acceleration on Y axis

In fact, for example with the accelerometer with the Y axis on the figure 3.9, the blue curve represents the variation of the value of the Y axis sensor for the acceleration during 175 seconds. It's a test made with a Python program to observe how the acceleration evolve during a movement. When the sensor is moved on the direction of positive Y there is a rise in value and immediately after a fall until reaching the opposite value and the returning to a value close to zero. This phenomenon is due to the compensation of the acceleration.

Indeed, when something accelerate and stop very quickly, an opposite acceleration, due to the compensation, is detected. That create a problem if we want to distinguish a movement (go to right for example) and its opposite (go to left).

To detect the right movement and to prevent the acceleration compensation, the technique adopted is the creation of a state machine to analyse every possibility of our signal and assign the appropriate move when it's relevant.

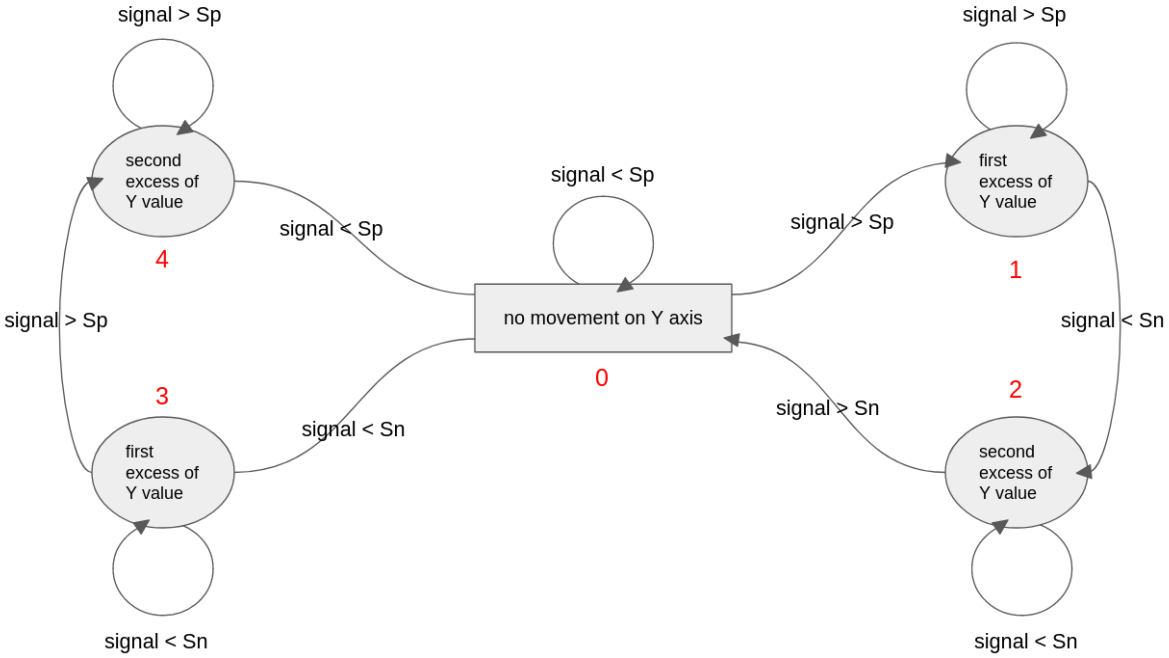


Figure 3.12: Schematic of the state machine for the accelerometer

The numbers in red represent all the different states in which the system can be. Sp and Sn are two thresholds which are been chosen after some tests with the different values of the accelerometer, Sp is the positive threshold and Sn is the negative one.

To explain this state machine, as long as the value of the accelerometer doesn't surpass Sp or Sn, there isn't movement detected. As soon as the threshold is exceeded by the

value, the instruction of the movement is sent at the state 1 or 3 (for example, with the X axis, when Sp is exceeded the program sends "right", and "left" if it's Sn). As long as the value is superior (for Sp) or lower (for Sn), there isn't movement sends.

The state 2 and 4 are made to avoid the acceleration compensation. Indeed, for positive value, while it doesn't surpass Sn, the program stays in the state 2 and send nothing. That is the same with negative value, while it doesn't be lower than Sp the program stays in the state 4. The result is observable with the orange curve one the Figure 3.9.

In this project, this state machine will be mainly used for the "left" and "right" movements (on the X axis) and for the "down" movement (on the Z axis).

3.2.3.2 The gyroscope

Another problem appears with the gyroscope.

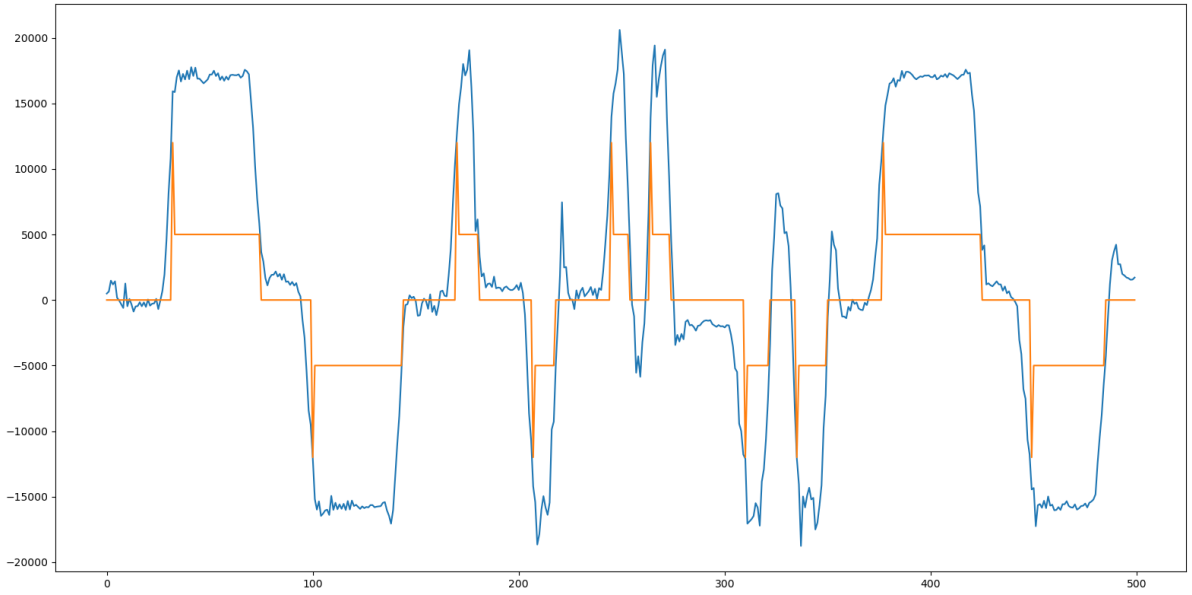


Figure 3.13: Graphic of the evolution of gyroscope on X axis

Indeed, when the controller is turned in one direction (left or right), everytime the program on the Arduino makes a loop and as long as the position of the controller is not changed, an instruction of movement is sent (blue curve on Figure 3.11). To have a playable game, the gyroscope must send the movement only after a change of the position of the cube.

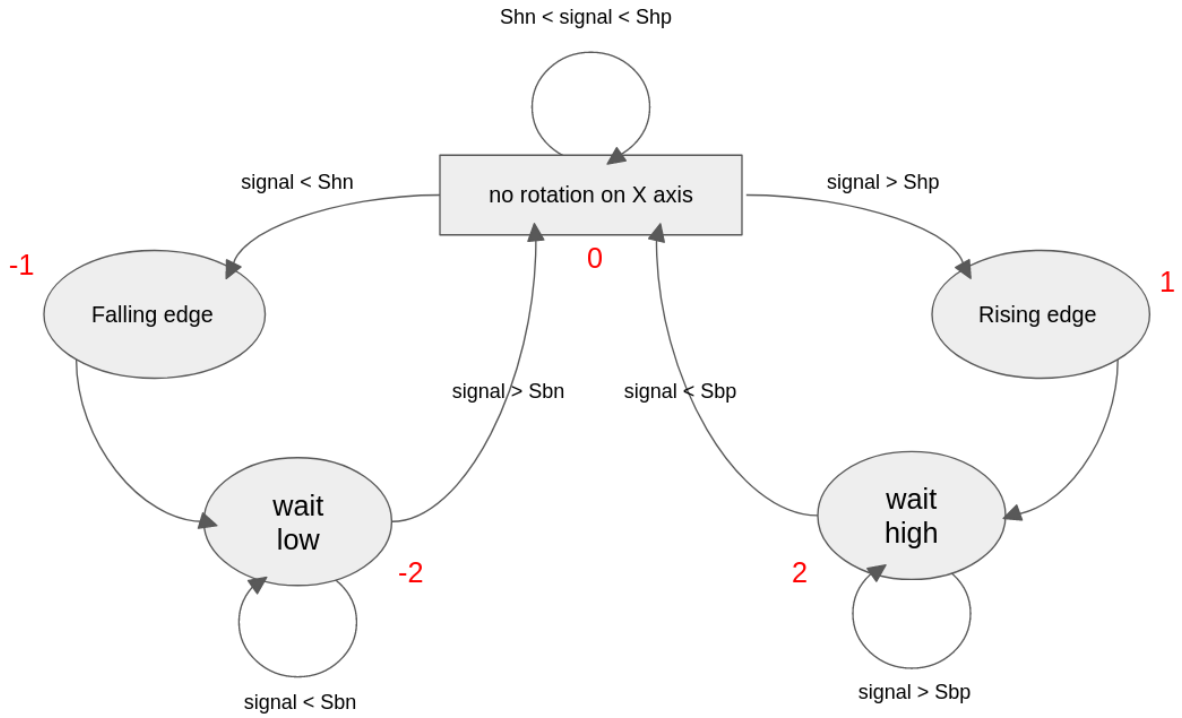


Figure 3.14: Schematic of the state machine for the gyroscope

In this state machine, four thresholds have been introduced, two for the positive values and two for the negative values (one for a low threshold and one for a high threshold).

For example with the positive part, as long as the value of the gyroscope doesn't surpass Shp (the positive high threshold), nothing will be send and the program stay on state 0. At the moment of the value exceeds Shp the program pass on state 1, the instruction of the movement of rotation is sent (here "rotright" for the right rotation) and directly after the program changes on state 2. While the value is higher than Sbp, nothing changes and the program stays on state 2. As soon as the signal is lower than Sbp, the program changes on state 0 only and nothing is sent.

The effect of this state machine is observable one the figure 3.11 with the orange curve. In the graphic, the Shp is equal to 12000, Shn is equal to -12000, Sbp is equal to 5000 and Sbn is equal to -5000. These different values are chosen to have a good playing experience without interfere with the acceleration information as far as possible.

3.2.4 The Bluetooth

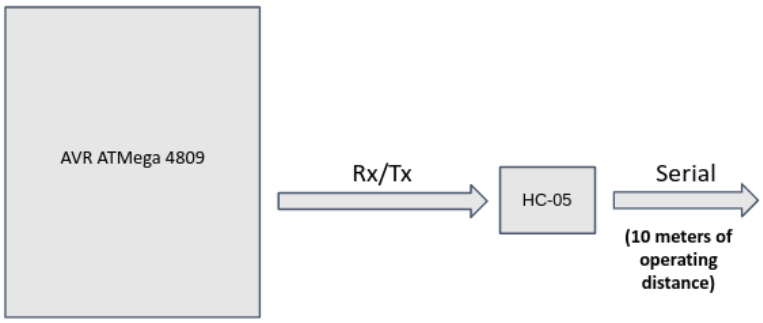


Figure 3.15: Schematic of the blutooth part

The bluetooth deserves to link the controller (here the Arduino) to send the instruction of movement with the arcade terminal (here the Raspberry). Every information is sent with a serial communication. In this project, the information sent by the Arduino are "left", "right", "down", "rotright" and "rotleft".

The bluetooth is made with a HC-05 which is linked to the Rx/Tx pins of the Arduino. It's a good bluetooth module in the case of this project because it's easy to use, the consumption of the battery isn't so high (80mAh), the sending data time is closed to the real time and the operating is higher than 10 meters.

4.The arcade terminal

This part deals with the creation of the arcade terminal. The arcade terminal is composed by a screen and a Raspberry Pi 3b+.

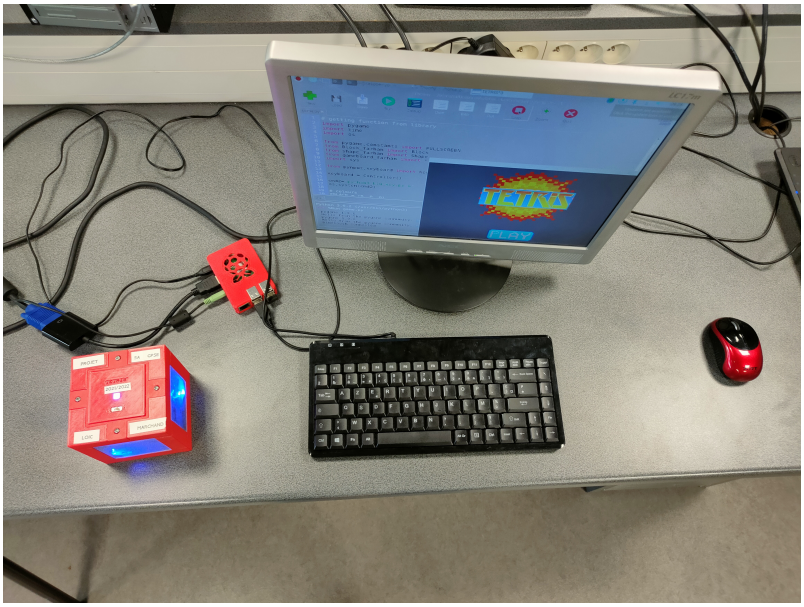


Figure 4.1: Components of the arcade terminal

As we can see on the figure 4.1, a keyboard and a computer mouse are still necessary to navigate and use two commands to play Tetris with the controller (the commands are on the Appendix B.1).

4.1 The Tetris game

The Tetris game is a Python program inspired by the Tetris program of farhan on GitHub.

The game is divided in 3 parts.

There is the opening screen :

Figure 4.2: Opening screen of the game

After press the ENTER key, the game is on and a second screen replaces the first one. It's the screen where the user plays the game :

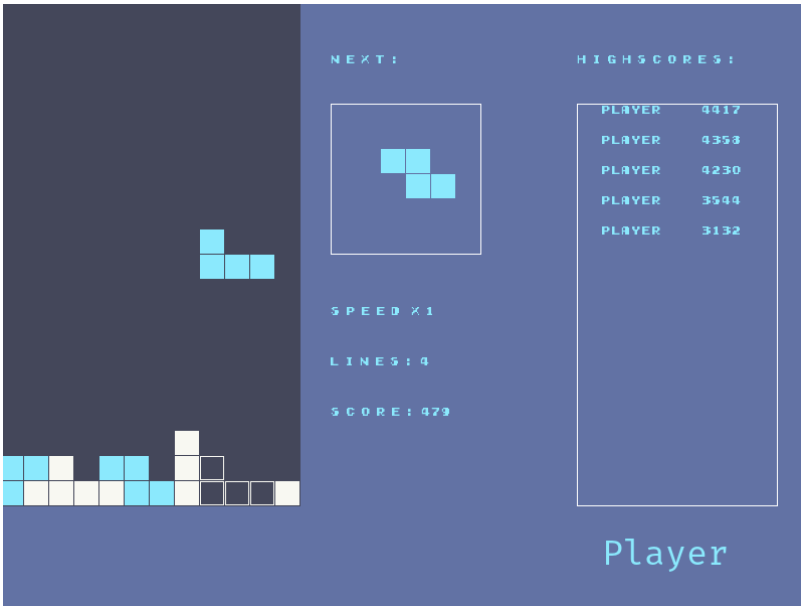


Figure 4.3: Screen of the playable game

When the game is over, an new screen appears :

Figure 4.4: Game over screen

To start a new game, the ENTER key has to be pressed again.

This Python program has been chosen because of its simplicity of programming. In fact, all the game is developed with Pygame library. This library deserves to program some games (not only Tetris) on Python and it's a very intuitive library. Thanks to this library, it's easy to change the music, the sound and the color of the game (blocks, interface), the opening and the game over screen, the speed of the game and the score.

The game launch and the bluetooth connection between the controller and the raspberry aren't automatic, so a keyboard is indispensable to write the appropriate line of code on the terminal before playing.

4.2 The controls

After customize the aspect of the game, the last thing to do is to link the information send by the Arduino with the commands of the game.

When the Arduino sends an information, the Raspberry receives the following data :

```
left => b'left\r\n'
right => b'right\r\n'
down => b'down\r\n'
rotleft => b'rotleft\r\n'
rotright => b'rotright\r\n'
```

Then, it's necessary to choose which letter will be associated with which movement.

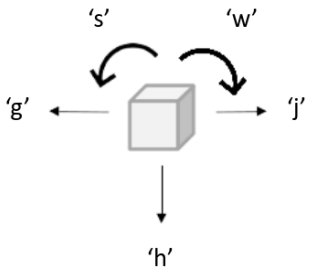


Figure 4.5: Letters associate with movement

After chosen the letters, to interact on the game with the controller, a second Python program must be written. This second program will imitate the press of one key on the keyboard. For example, when the user go right with the controller, the raspberry will associate the data it receives with the press of the 'j' key of the keyboard. It will run during the game in the background and it will begin in the same time the game is launched.

The Pygame library has always some event which recognize what letter is pressed on the keyboard. With this functionality, the project could be changed to be adapted to other old agames (PacMan for example because this game has some simple movements too).

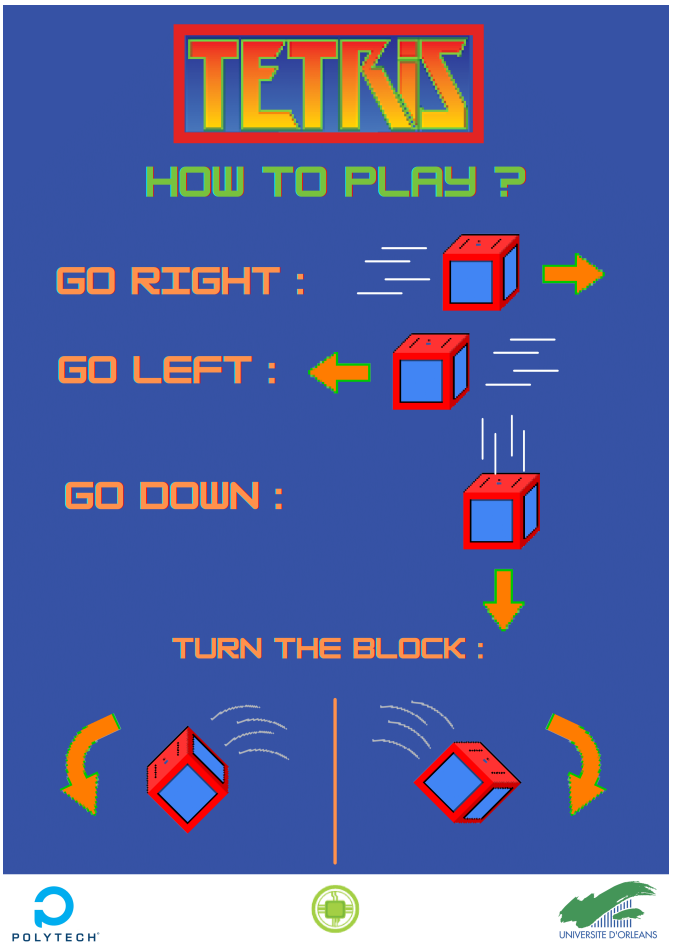


Figure 4.6: Movements of the game

To make understanding of the controls of the game more intuitive, a document which specifies all the possible movements of the game is made.

Due to lack of time, the aspect of the arcade terminal isn't made.

5.Conclusion

5.1 General conclusion

This project was divided principally in two parts. One part about the creation of a bluetooth controller and a second part for the assembly of the arcade terminal. For the first part, even if the gameplay isn't perfect with the use of the state machines, the criteria are mostly respected. Maybe the use of the interruption to send just at the right time the information could be a good option. For the second part, all the technical part is operational, just the aspect of it isn't here. For the aspect, it may be interesting to add some LED which light up at the same time of a particular action on the game.

5.2 Personal conclusion

Personally, this project was a very instructive project in terms of technical skills and project management.

With this project, I have developed my skills on programming with the comprehension of the Tetris program to adapt it with the project (the colors of Polytech school on the blocks for example). I discovered the use of the state machine with the different sensors. By modelling the cube, I learned a lot of things with the use of SolidWorks. The programming of the Arduino improves my skill on C language and the use of the bluetooth gives me a better understanding about the communication between 2 electronic boards (here the Arduino and the Raspberry).

In terms of project management, I upgraded my skills about time and tasks management. Indeed, being alone on this project gives me a global vision of each role that an engineer can have.

List of Figures

1.1 Diagram of the project 3

2.1 Header beast diagram 5
2.2 Diagram of the global operation 7

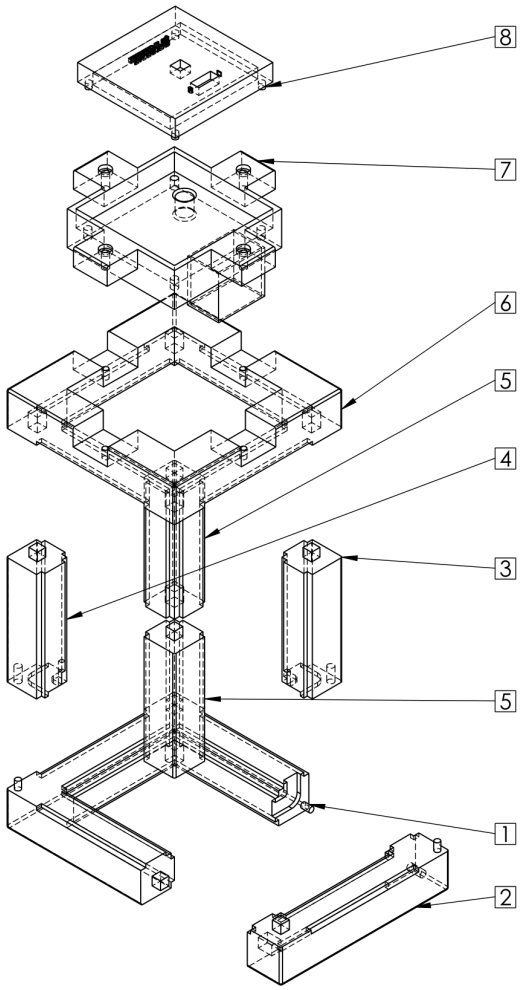
3.1 Photo of the controller 8
3.2 Model of the tops of the controller 9
3.3 Model of the posts of the controller 9
3.4 Model of the bases of the controller 10
3.5 Photo of the electronic part 11
3.6 Schematic of the electronic part 11
3.7 Schematic of the LED strip part 12
3.8 Schematic of the battery part 13
3.9 Schematic of the tension dividing bridge 14
3.10 Schematic of the MPU6050 part 15
3.11 Graphic of the evolution of acceleration on Y axis 15
3.12 Schematic of the state machine for the accelerometer 16
3.13 Graphic of the evolution of gyroscope on X axis 17
3.14 Schematic of the state machine for the gyroscope 18
3.15 Schematic of the blutooth part 19

4.1 Components of the arcade terminal 20
4.2 Opening screen of the game 21
4.3 Screen of the playable game 21
4.4 Game over screen 22
4.5 Letters associate with movement 23

4.6 Movements of the game 24

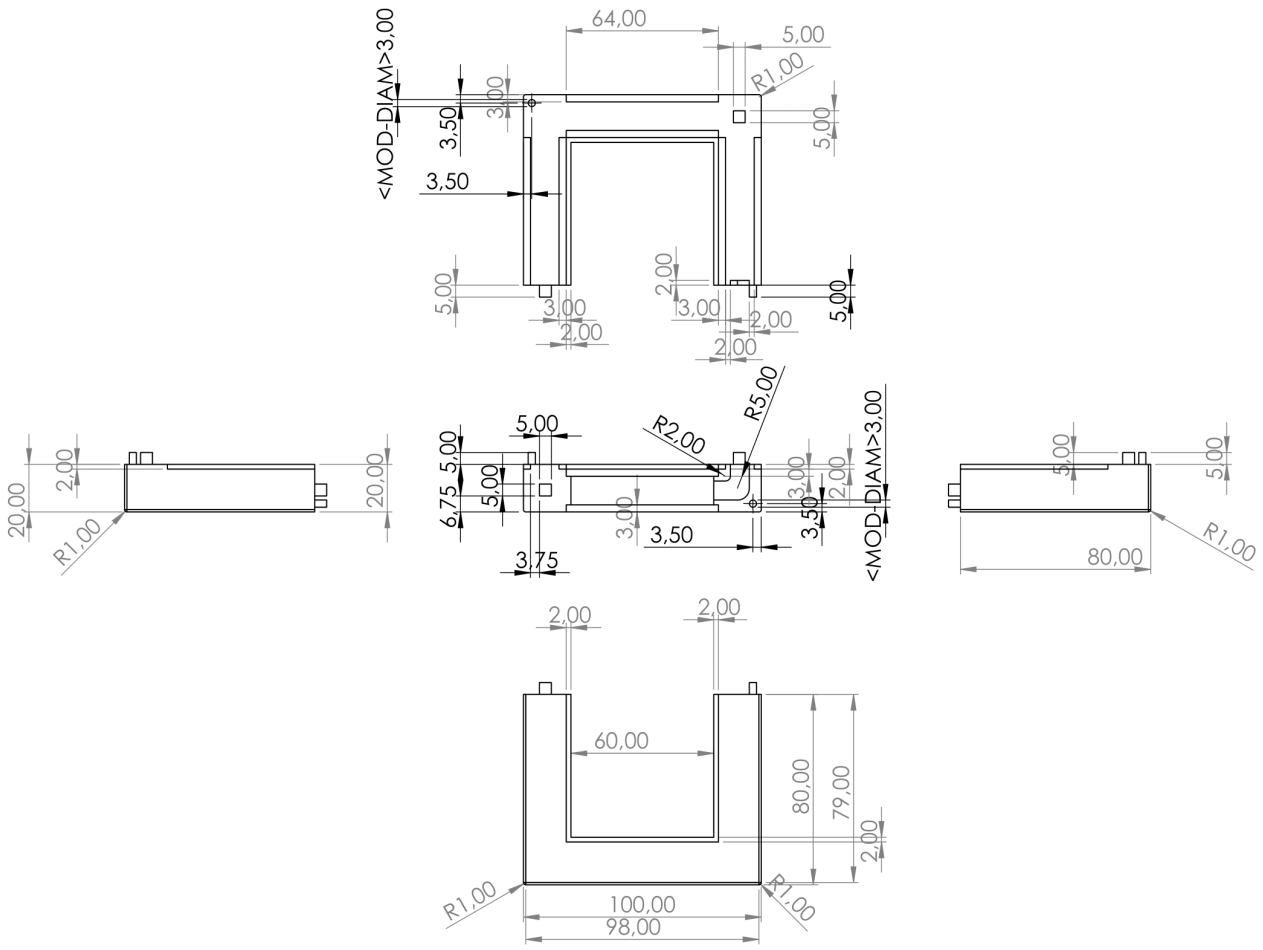
A.Appendix of the controller

A.1 Assembly instructions

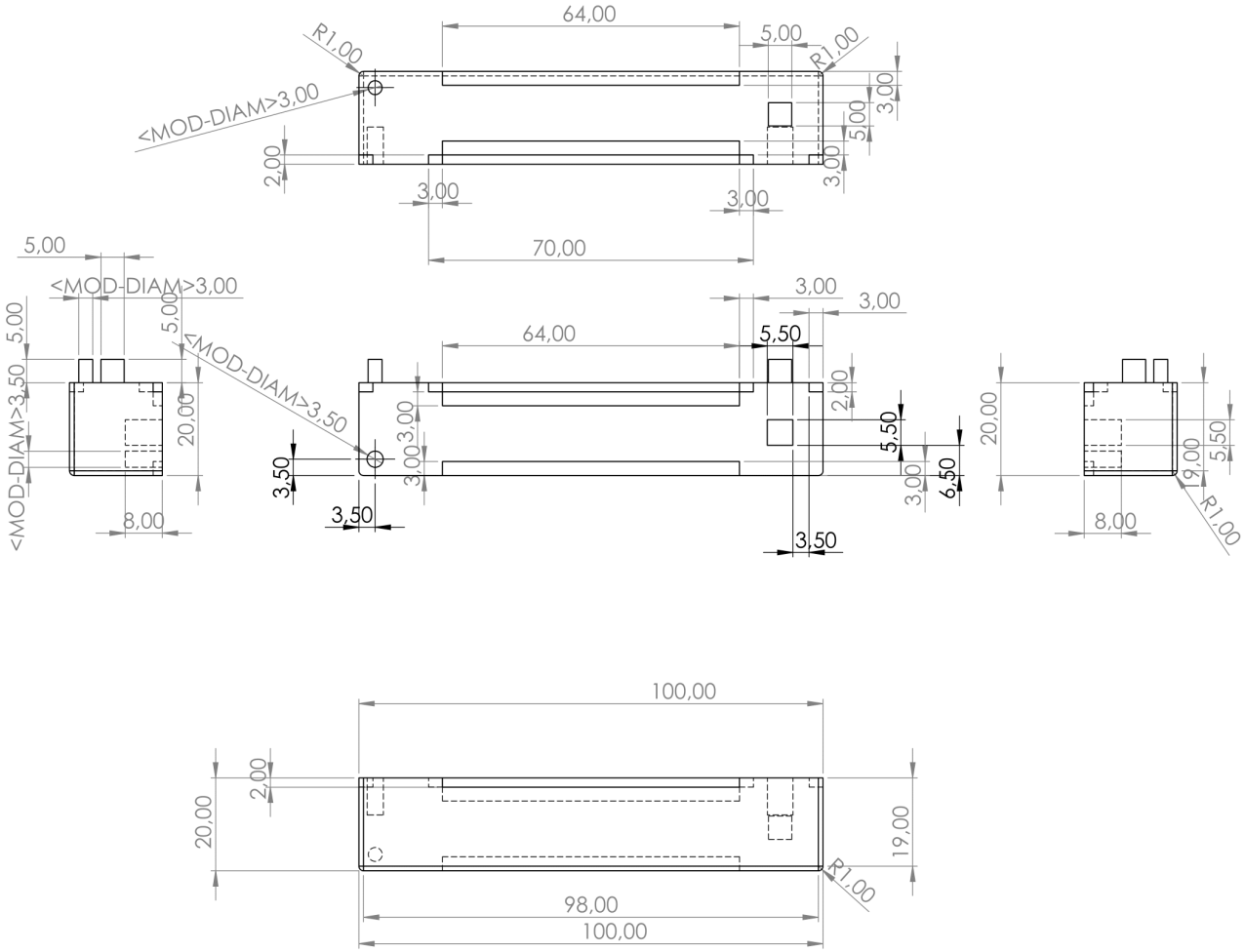


ITEM NO.	PART NUMBER	QTY.
1	base_1	1
2	base_2	1
3	poteau1	1
4	poteau3	1
5	poteaux24	2
6	toit1	1
7	toit2	1
8	toit3	1

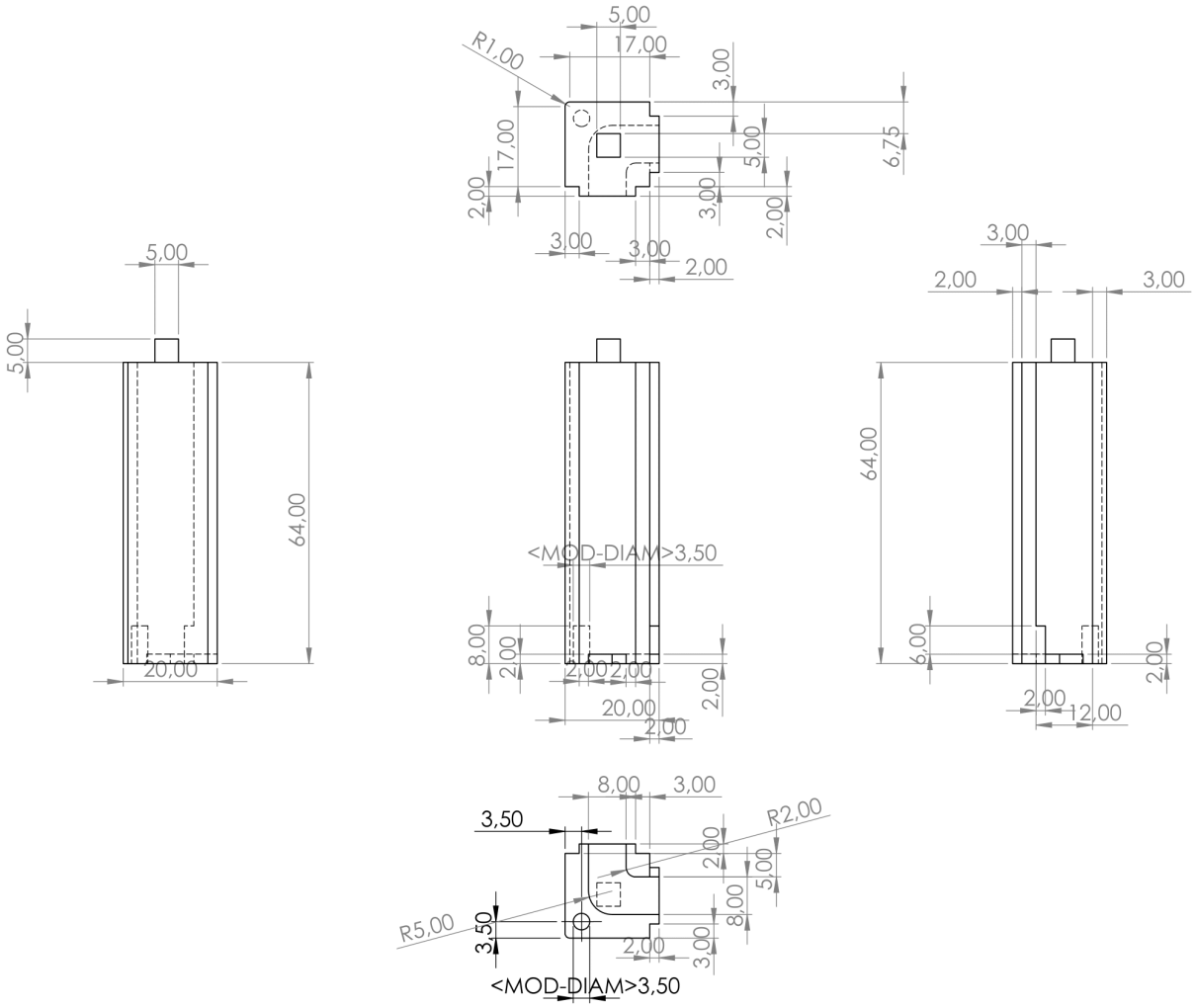
A.2 Drawing of base_1



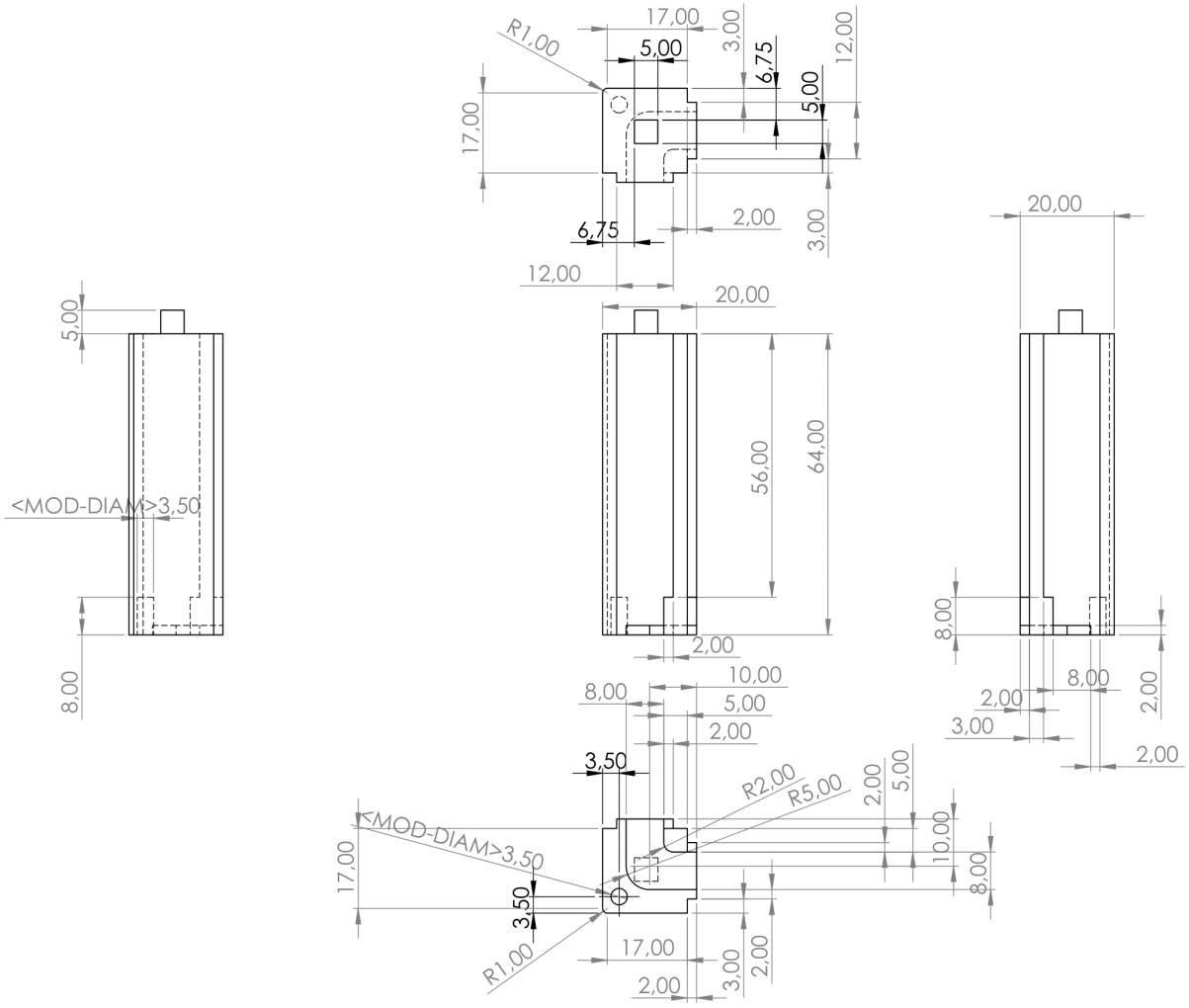
A.3 Drawing of base_2



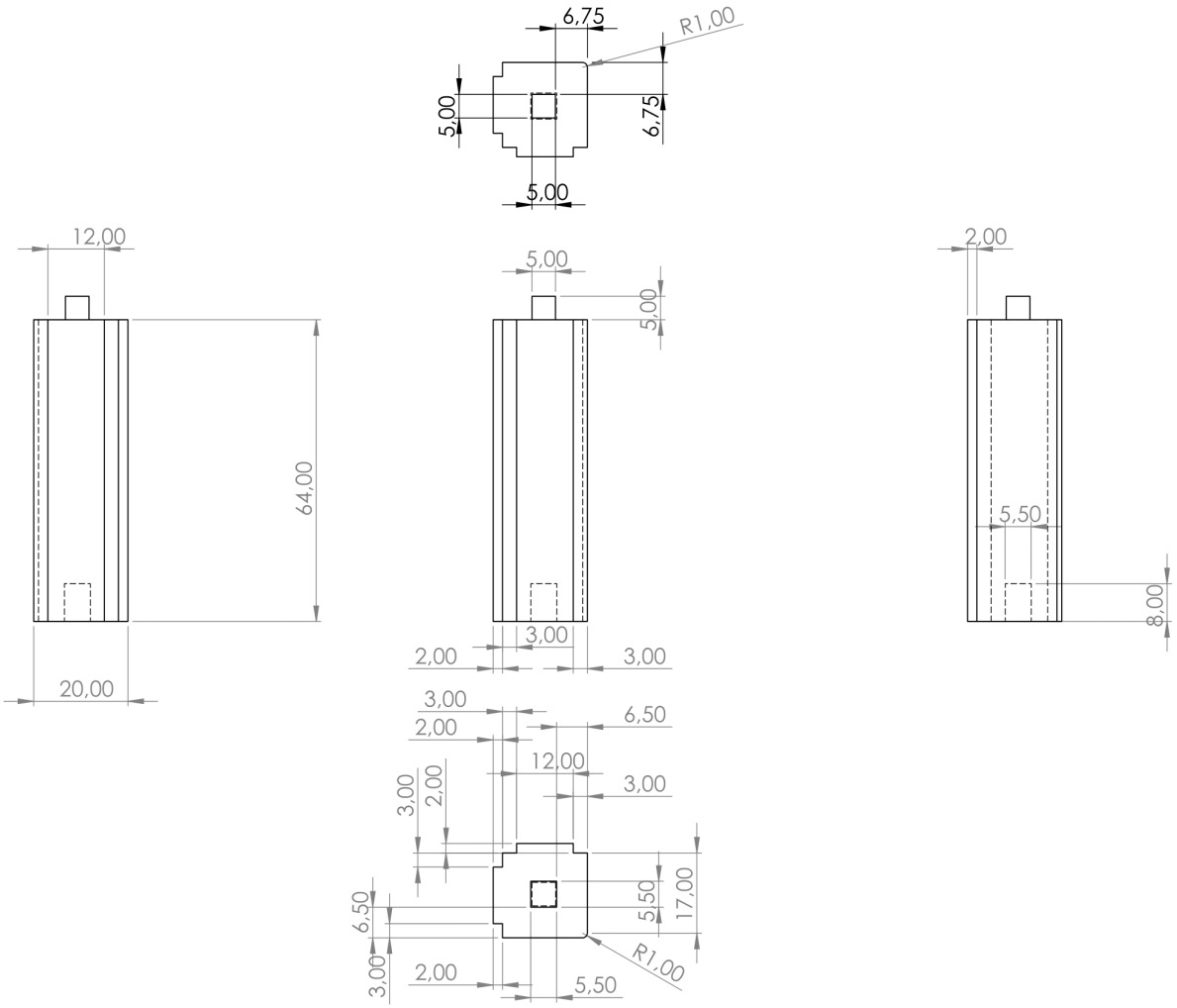
A.4 Drawing of poteau1



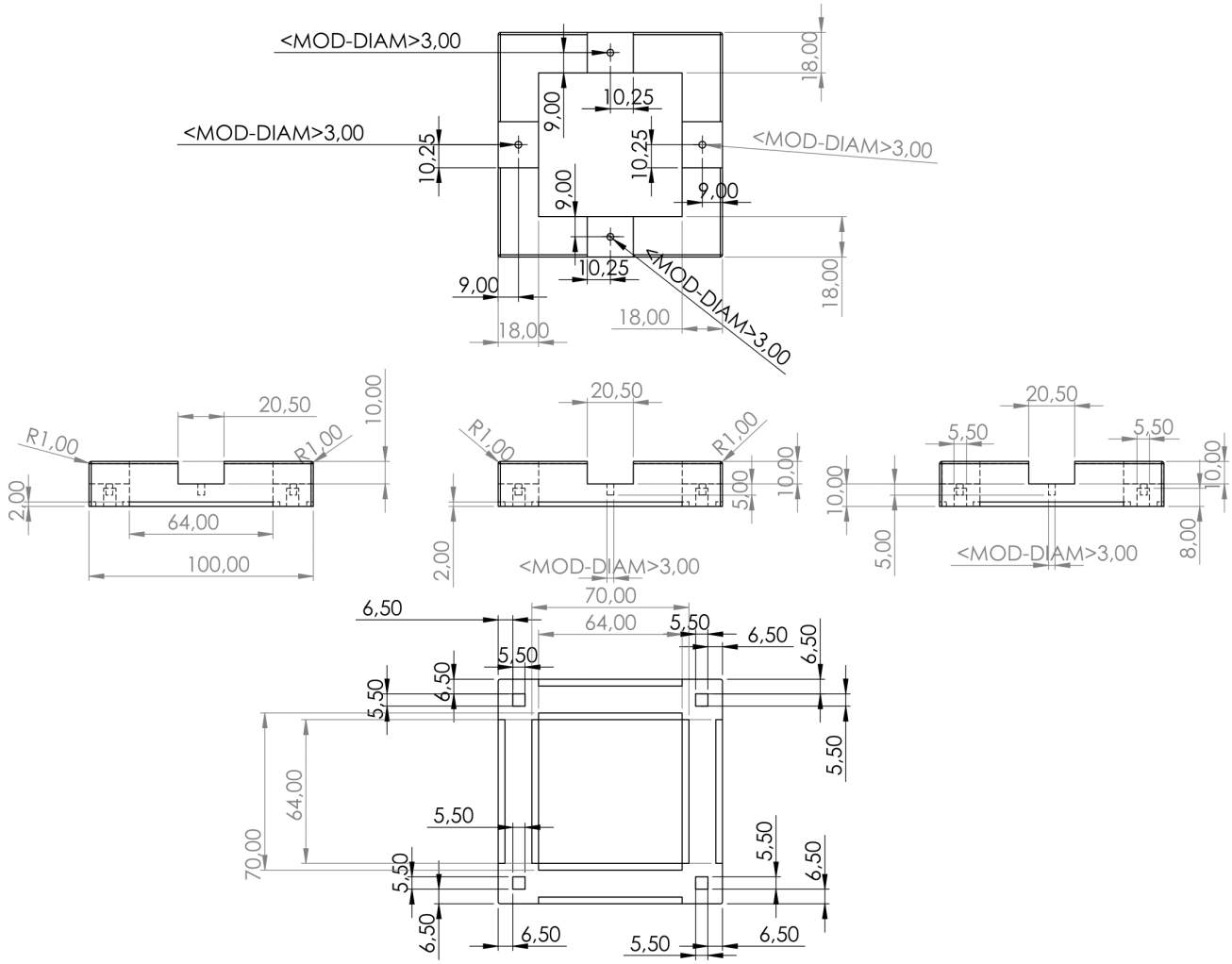
A.5 Drawing of poteau3



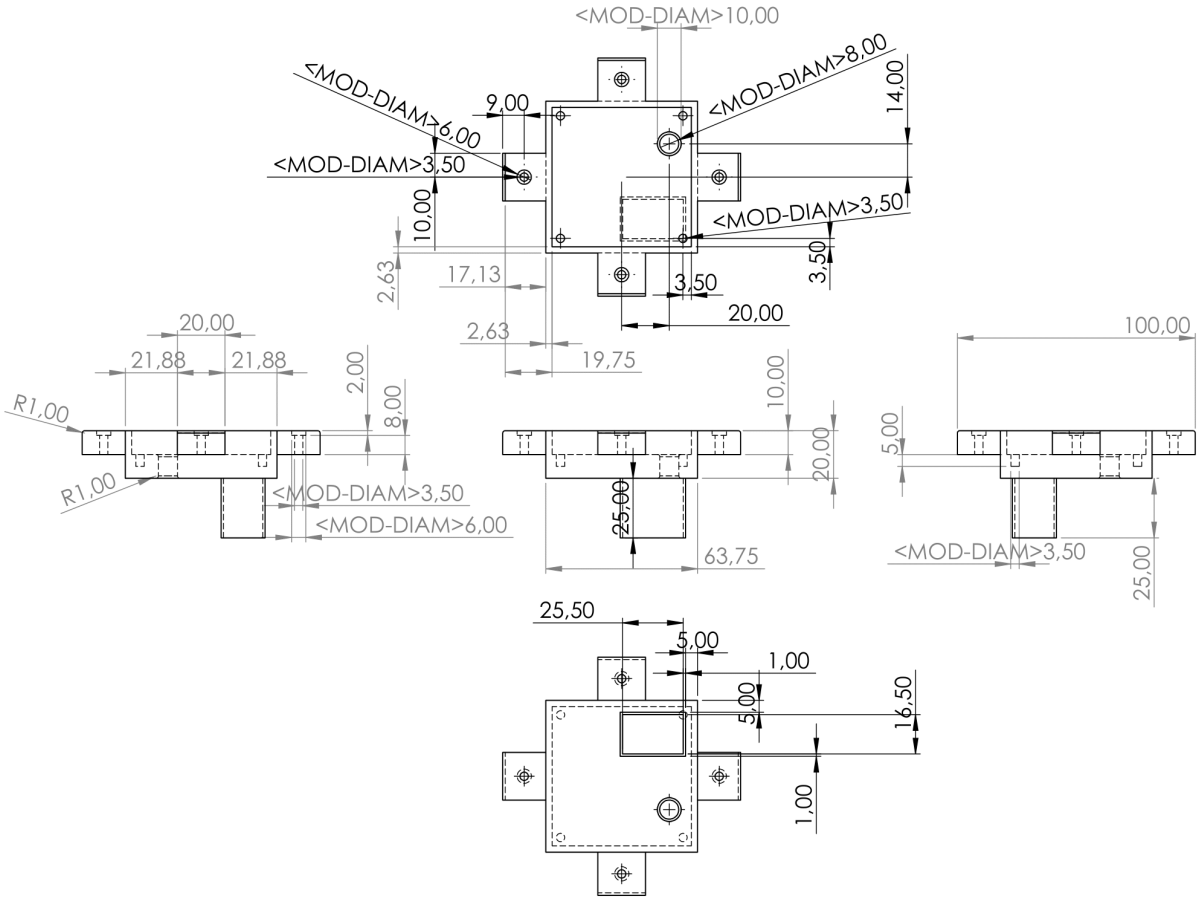
A.6 Drawing of poteaux24



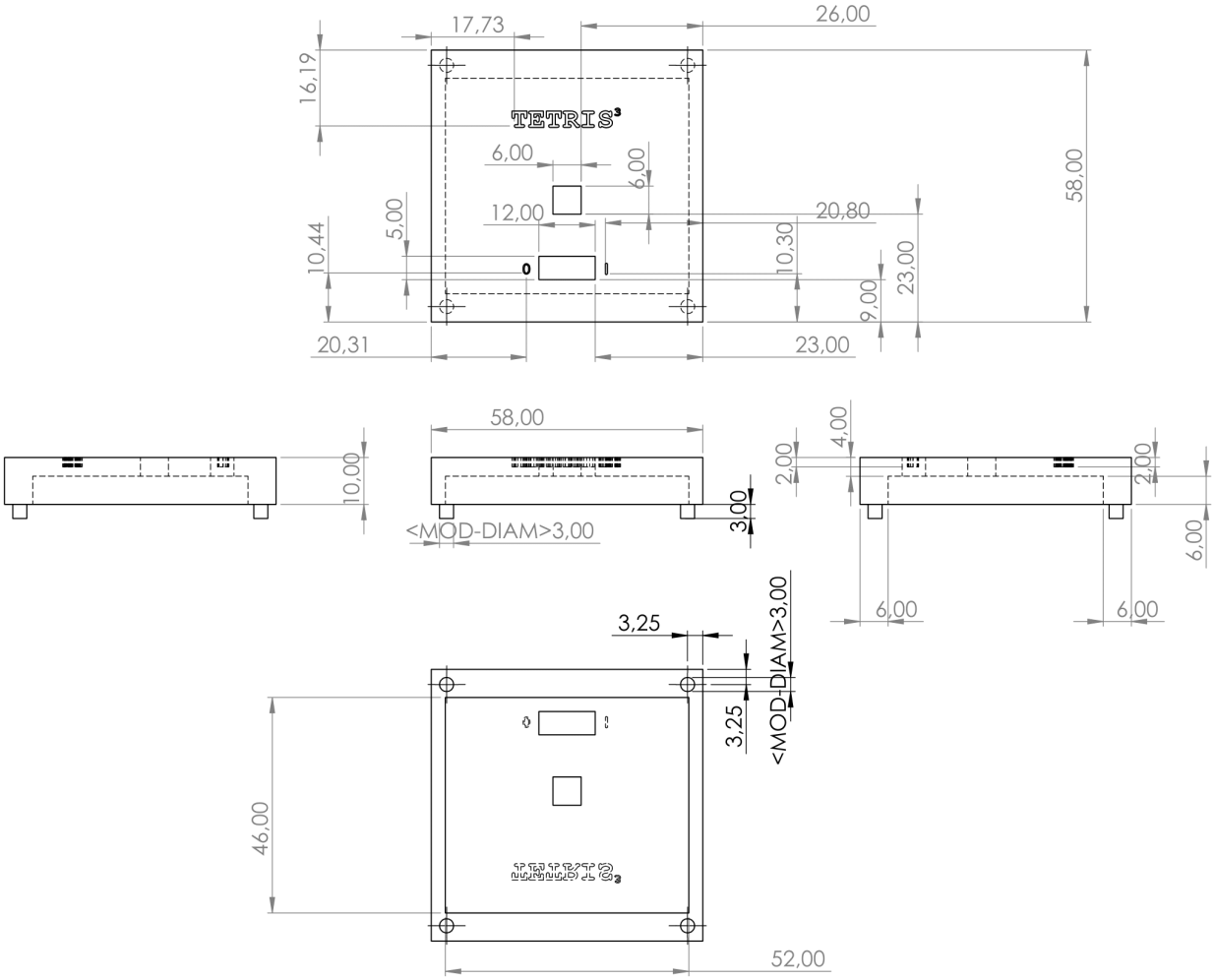
A.7 Drawing of toit1



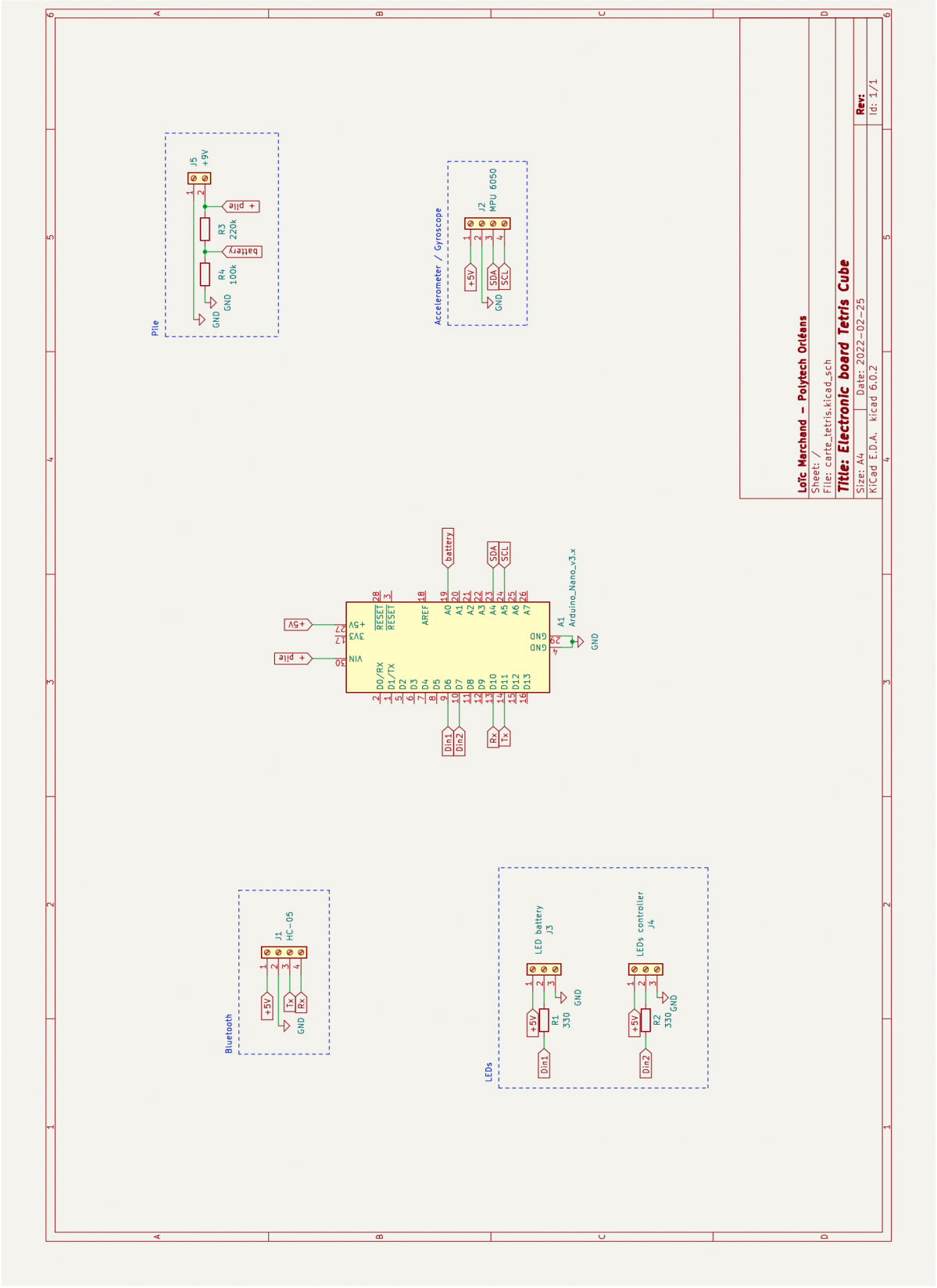
A.8 Drawing of toit2



A.9 Drawing of toit3



A.10 Detailed schematic of the electronic part



B.Appendix of the arcade terminal

B.1 Command to launch the game on the Raspberry

B.1.1 To connect to the HC-05

Turn on the controller

sudo rfcomm connect hci0 98:D3:91:FD:E6:9D 1

(XX:XX:XX:XX:XX:XX code depends on your stuff)

B.1.2 To run the game

cd tetris_bluetooth

run game.py (if it doesn't work, use : python3 game.py)

B.2 Github of the project

https://github.com/LoicM22/Tetris_project.git