Pong

DES Project

Matteo Rotundo

Year 2015/2016

Table of Contents

1. Introduction

The project implements the Pong Game. The code runs on the Arduino UNO board that uses an RGB matrix to visualize the game. For interacting with the game, it's provided to the users a controller each one with two push buttons.

2. User Requirements

The project shall realize a simulation of the Pong Game. This simple “tennis like” game features two paddles and a ball, the goal is to defeat the opponent by being the first one to gain 5 points. A player gets a point once the opponent misses a ball.

The game must be played with two human players, so it shall be provided to each player a pad in order to control the movements of the paddle. The possibile directions must be two: one at the left of the paddle and the last one on its right. It shall be provided an acceleration to the paddle by moving it in the same direction without releasing the button. This acceleration will reflect its behaviour when the paddle hits the ball, indeed the velocity of the ball must be proportional to the velocity of the paddle in case of impact. The direction of the ball after a hit shall be influenced by which point of the paddle hits the ball.

The dynamic of the game must be visualized on a RGB matrix. Also, it must be visualized dinamically the points done from each player. When a player gains the score to win, the match must be interrupted and it must be visualized on the matrix the name of the winner.

3. Functional Requirements

The project code runs on the Arduino UNO board. Due to this, the code has been written in C++ language. Besides, the matrix used to visualized the game is the Adafruit RGB 32x32. In the implementation, must be taken into account the combination of the two devices. Indeed, Arduino UNO provides a small amount of SRAM (2K) and vice-versa, the Adafruit matrix occupies an high amount of memory (1600 byte) with the respect to the available memory of the board. So, the code must be very efficient in order to avoid the possibility of running out of memory.

The input commands of the two players are taken from two pads. These game controllers are realized with buttons that switches the voltage with two possibile values (0V and 5V) to Arduino pins. These are protected with some pull-up circuits.

The application will run on the Arduino board without the support of an operating system, so it shall be needed a way to simulate a scheduling in order to guarantee different speeds among the paddles and the ball.

Paddles can be visualized on the matrix with a line of leds turned on. Also the ball could be reppresented like a led turned on. A goal can be done only if the ball goes after the x-coordinate where the paddle is set.

Handling the direction of the ball according to the point of the impact with the paddle is limited from the use of the resolution of RGB matrix. Indeed, in order to have also a good dynamic the ball can have only the direction of cardinal points. (N, S, E, O, NE, NO, SE, SO).

4. Design

The system is designed with Papyrus. The model is splitted in 3 parts:

- Functional;
- Platform;
- Mapping;

4.1 Functional

This parts models the logic of the software. The following image is the BDD of the Functional System. It's composed from a GraphicHandler that has to handle the events and draw it on the matrix. It's also composed from the SystemSupport which represents the standard API for Arduino and the MapGame which contains the information of the two paddles and the Ball.

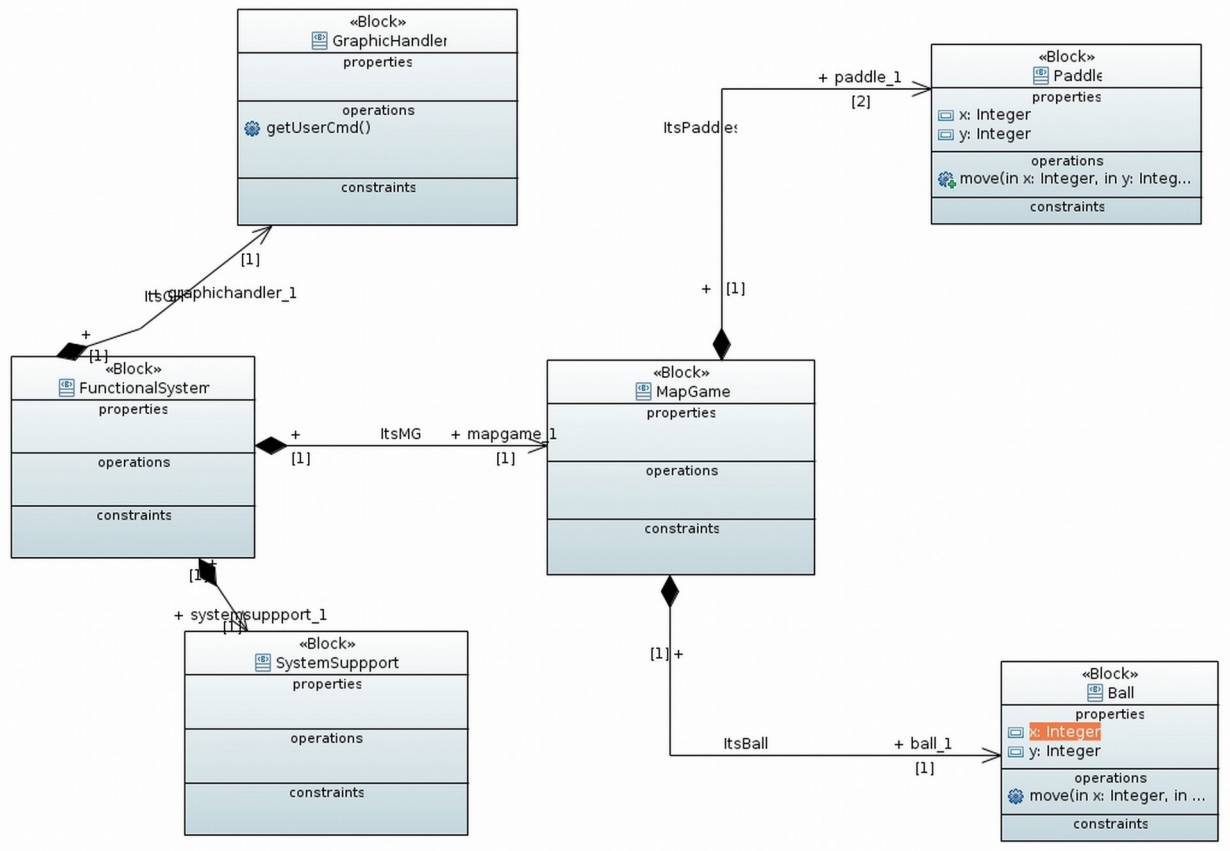

Illustration 1: BDD FunctionalSystem

In order to go deep in the model, we can see the IBD of the FunctionalSystem that explains the data operations that the GraphicHandler calls for reading the button through Arduino API and for drawing on the Matrix with its API.

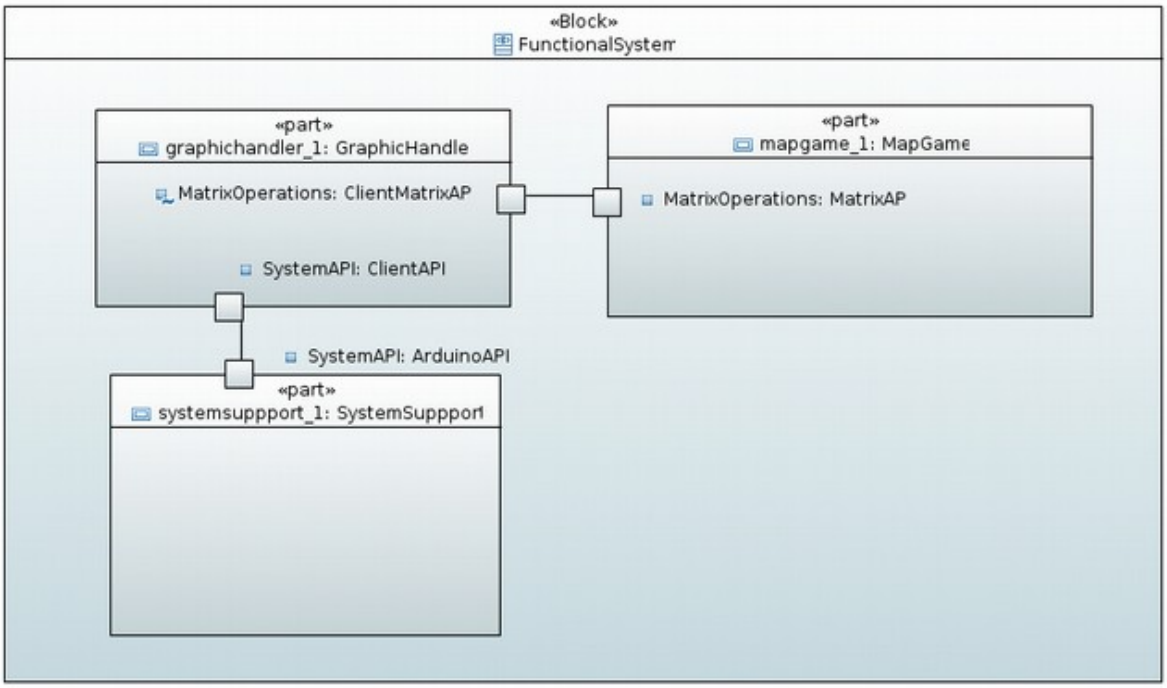

Illustration 2: IBD FunctionalSystem

4.2 Platform

This part describes the architecture. In the following illustration we can see that Platform System is composed from two GamePad, one RGBMatrix and an Arduino board.

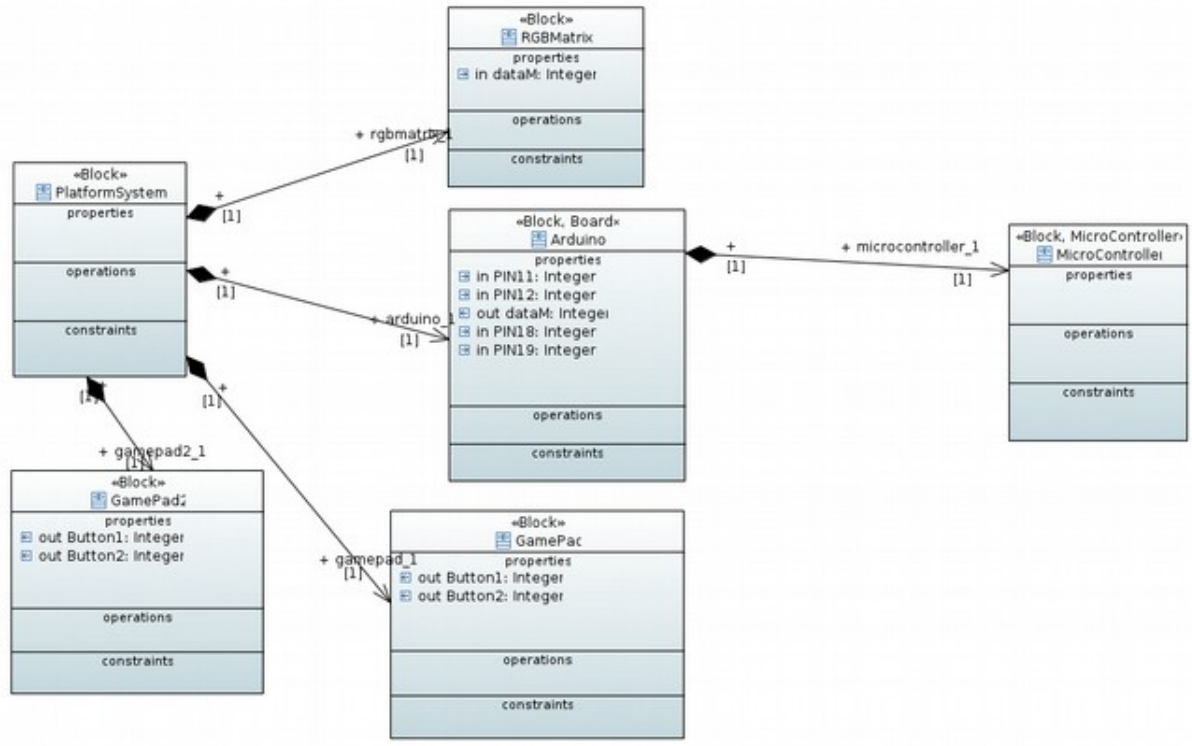

Illustration 3: BDD PlatformSystem

Finally in the IBD of the Platform System we can see the physical connections among the devices in the system.

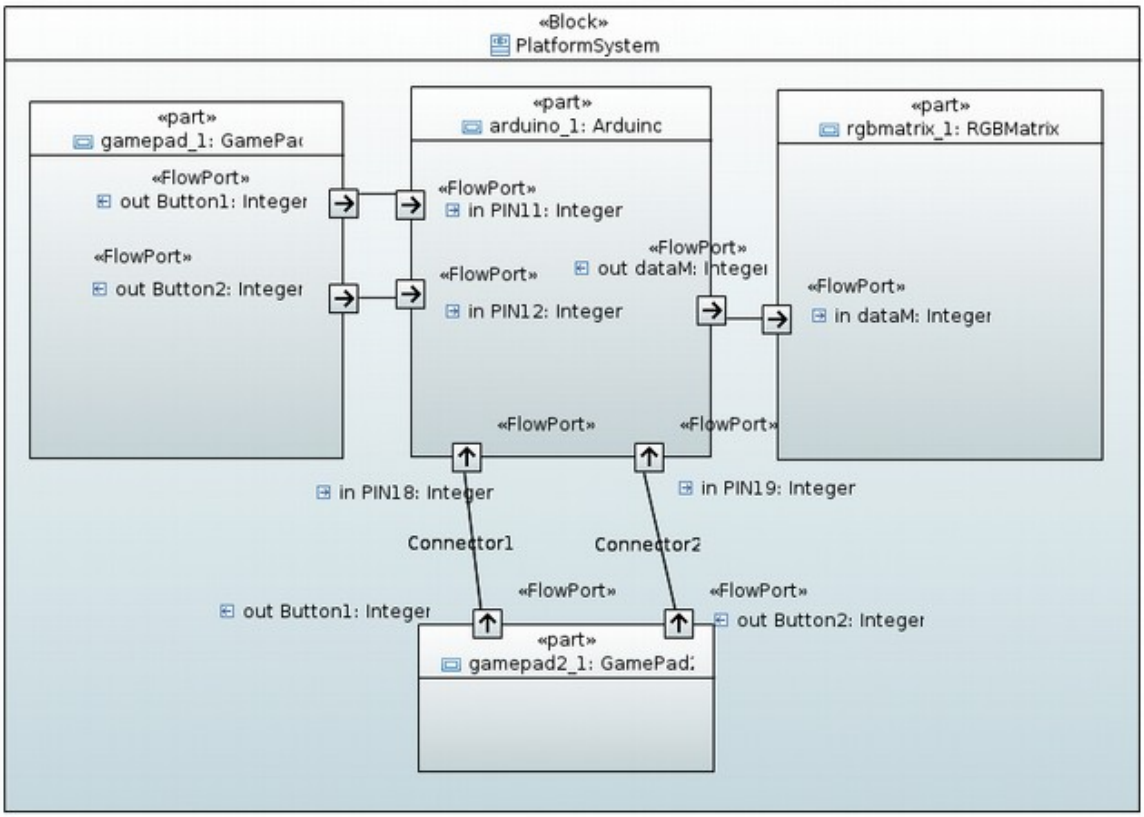

Illustration 4: IBD PlatformSystem

4.2 Mapping

This part models how it is mapped the functional system with the platform system. The mapped system is composed from the functional system, the platform system and a controller task.

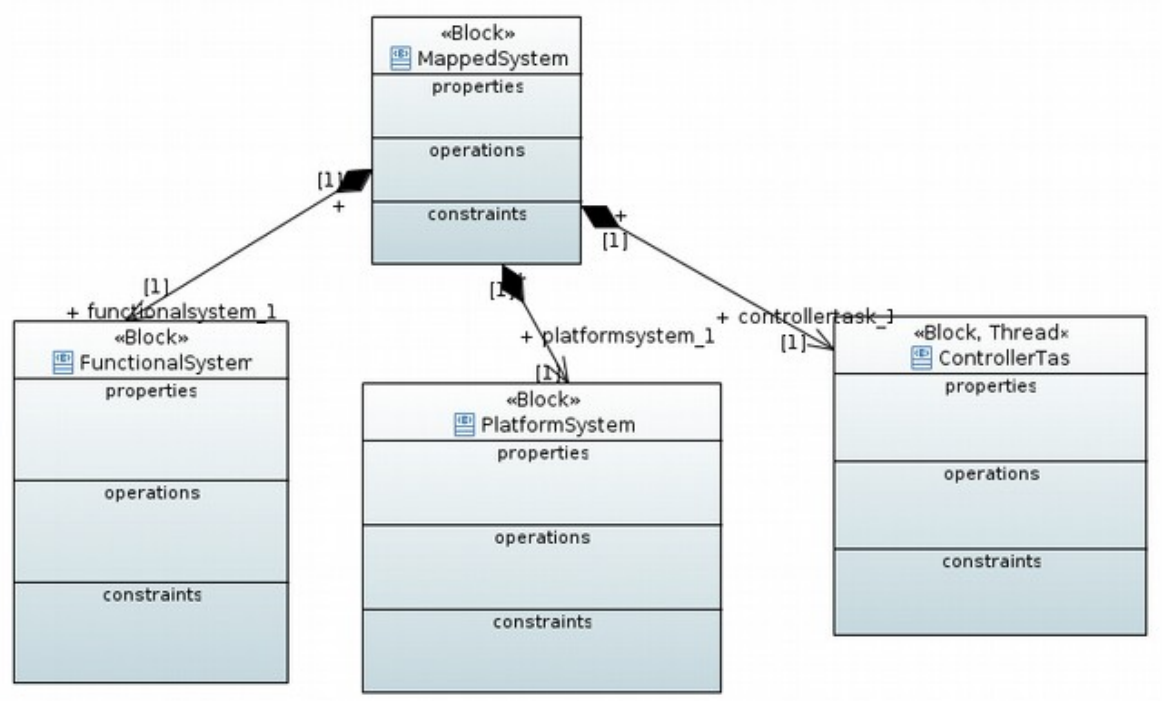

Illustration 5: BDD MappingSystem

The mapping is clearly illustrated in this IBD:

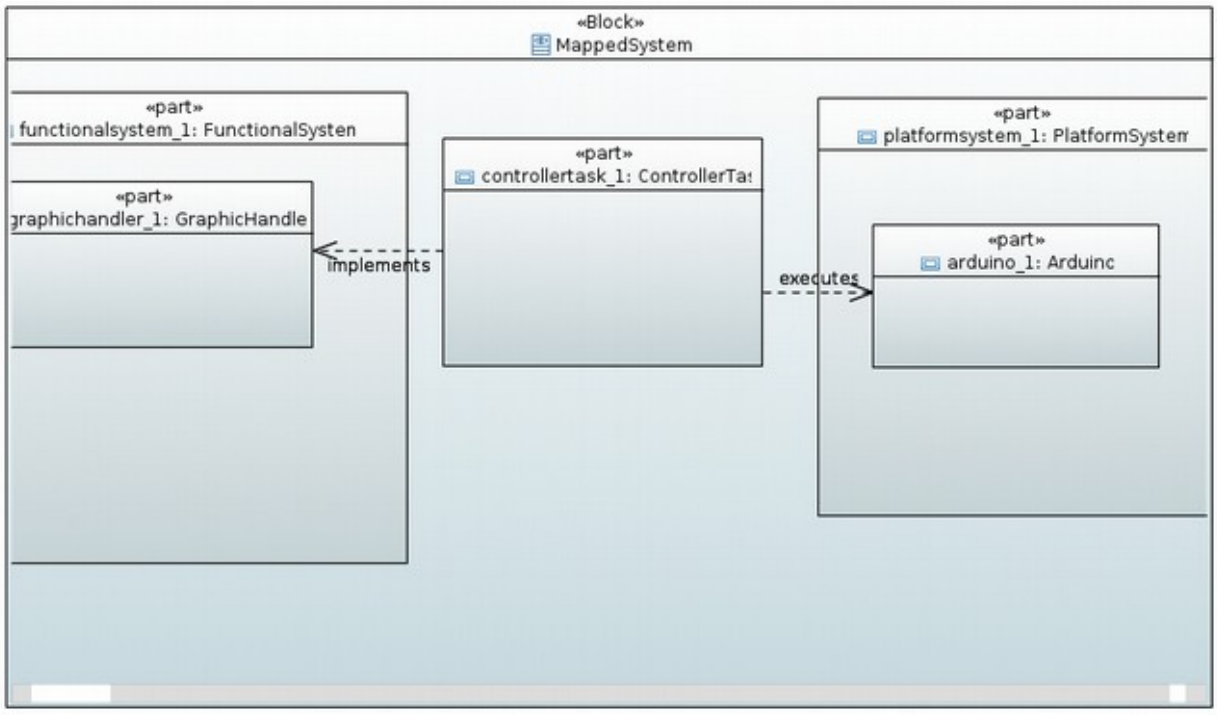

Illustration 6: IBD MappedSystem

5. Implementation

The project has been splitted in two main modules:

1. A support library;
2. The main sketch;

The support library in the files Pong.hpp and Pong.cpp provides the implementation of the class MapGame which contains all the data for the paddles and the ball. It also provides all the methods for moving them in the map.

The main sketch provides a class GraphicHandler that provides the methods for updating all objects on the matrix. Then, it implements with the function update when and how the process must react from the external events.

Due to the lack of an operative system, it's used a technique simil to the timeline scheduling. With the function **millis** it's got the time and with different periods are performed different actions.

The actions that are performed are:

- moving the first padde;
- moving the second paddle;
- updating the movement of the ball;

The periods selected are variable in a range which guarantees a minimum and a maximum velocity for each one. The pins, which are linked the buttons, are read with the same period of the corresponding paddle. This will not cause any problems because, the time needed for a button press is evaluated statically from an external sketch and the maximum period of the paddle is less or equal to the minimum value of the time for a button press.

In order to implement an acceleration for the paddle, every time that a button is pressed without releasing the button, the period of the paddle is decreased by some default units until its minimum value, otherwise it's set to its maximum value. When the ball hits a paddle, the period of updating the ball is changed according to this rule:

Tball = Tpaddle * α;

where α is found by an empiristic way in order to make a good dynamic game.

Due to the low resolution of the matrix the direction of the ball follows the cardinal directions. After an hit with the central part of the paddle or the front/bottom of the matrix the direction is inverted. Otherwise if the ball hits the extremes of the matrix or the extremes of the paddle, the direction becomes diagonale in order to simulate a shot shore.

Some methods of the class MapGame are:

```
void invertDirBall(); // invert the direction of the Ball
void movePad1(DirectionPad dirV); // move the paddle1
void movePad2(DirectionPad dirV); // move the paddle2
void moveBall();      // move the ball
void shotShore();     // change the direction of the ball for a shot shore
bool checkImpactPad1(int x, int y); // checks the impact with Pad1
bool checkImpactPad2(int x, int y); // checks the impact with Pad2
bool checkImpactWall(int x, int y); // checks the impact with the Wall
```

6. Test

In order to test the application it's used the framework CPPUNIT. For this purpose it's implemented in the MockMatrix library a mock for the matrix.

The mock is implemented with a matrix of RGB Led that is class with the following declaration:

```
class Led {
public:
      bool state;
public:
    virtual void turn_ON();
    virtual void turn_OFF();
};


class RGB_Led : public Led {
public:
      int color_R;
      int color_G;
      int color_B;
      void turn_ON(int R, int G, int B);
      void turn_OFF();
};
```

With this implementation a Led is ON if the state is TRUE and in the case of RGB Led, it must be the RGB state equal to the colour BLACK. (Like the real matrix)
For the same purpose, are also implemented the API for drawing on the matrix

```
void drawPixel(int x, int y, int R, int G, int B);
void drawLine(int x1, int y1, int x2, int y2, int R, int G, int B);
```

The test is performed on the methods for moving the paddles and the ball. After the operations it's checked on the mock matrix if the right leds are turned ON.