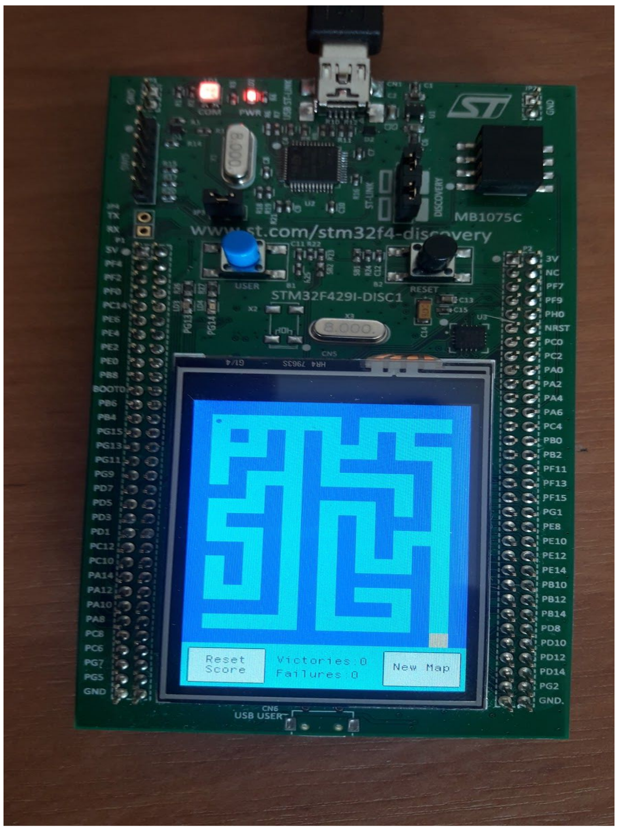
Introduction

The main target for this was to create using maze algorithm generation a simple maze game with ball moving between using gyroscope. Project was created on STM32F429 with Keil µVision. If there any inconsistencies in licensing are found please inform me. I have tried to give credit best to my knowledge but overlooks might have happened.

User guide

Usage of game is simple. The game starts with ball in the left top part of the maze. Changing an orientation of the board changes the position of the ball in the labyrinth. When wall is encountered the game ends with failure and if the ball reaches orange square games ends with a victory.



Pic. 1. Example

At the bottom located are touch buttons “Reset Score” and “New Map”. Those are pretty self explanatory. Resetting score will change values of victories and failures to zero and with the other button we can generate new map, numbers of victories and failures will stay the same if we generate new map.

Working principle

The basic working principle has been shown at the diagram below. The maze will be created of blocks 16 x 16 pixels each and the dimensions of the whole labyrinth are 17 block length and 15 block width.

1. At first using maze algorithm generation we create an array of 17 x 15 = 255 elements in which each one will be a wall and the zero will be a path. (In the code maze[255]).

2. The second step is to create another array where each pixel which creates an edge of the wall will have its own unique ID number. To do this we are using pairing function which is a bijection. It allows to differentiate clearly for example pair of 0 and 15 pixels from pair of 15 and 0 pixels. (In the code array_of_walls_ID_numbers[8700]).

3. Then we sort this ID array so we can later use binary search to detect if our ball has collided with the wall or has entered into the winning square

4. We enter the main game where in while loop gyroscope detects any change of board orientation, when change of orientation occurs new position is calculated ball is moved. Then calculated is an ID of the pixels creating an edge of the ball. If ID was found in an ID array then ball had hit the wall. If ID checks with winning square then the game ends with a win. Otherwise game continuous.

Additional information

The maze game was made entirely in file main.c, no changes in libraries were made. Dimension of array “maze” is 255 since maze consists of 15 x 17 block equal to 255. Dimension of array “array_of_walls_ID_numbers” is equal to 8700 since there are 255 blocks and about 55% of them are walls and each wall consists of 60 pixels which gives in total around 8415. Because no RTOS was used in this project usage of dynamic array was not a recommended option.