**Embedded System Programming**

**3D Labyrinth Game**

## Contents

This Test Project proposal consists of the following documentation/files:

1. Task Board Schematic
2. World Skills CPU Board Schematic
3. STM32L0x2K6Tx Datasheet
4. STM32L0 Reference Manual
5. STM32L0xx HAL Driver’s description
6. I2C serial interface Module datasheet for Character LCD
7. Shift Register\_A2982 Datasheet
8. Row Driver\_CD4022B Datasheet
9. Column Driver\_74HC595 Datasheet
10. RG Dot Matrix Pinout
11. Project Files Task Phase 1
12. Project Files Task Phase 2

## Introduction

## Description of project and tasks

The competition test project is a “blind” 3D labyrinth. The device’s user interface consists of

* A 8x8 RG dot matrix display (CA)
* Six push buttons
* A RGB LED (CC)
* A piezo-buzzer and
* USB as COM port interface.

The idea of the game is to find a route from start to finish blindfolded. In other words the game does not show the player where the walls are. The player has to “feel” his way while playing. When the player hits a wall, the device beeps telling the player that a wall blocks the route in the direction the player tried to go. Every time the player hits a wall he gets a penalty point. When the player gets to the finish, the game displays his penalty points and starts a new game.

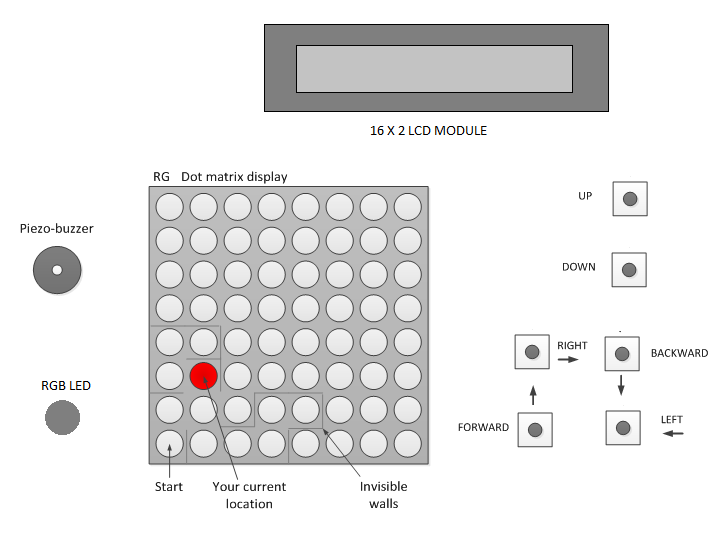


Figure 1: Component used in the project

The labyrinth is in 3D. You move from floor to floor using the up or down key OR Page Up or Page Down keys from PC keyboard via virtual COM terminal, but you must be in a position that allows up or down movement. Each floor is identified by a different led color (table 1). An RGB led is used here to denote the color.

Table 1: Floor colors

|  |  |
| --- | --- |
| Floor | LED color |
| 1 | RED |
| 2 | GREEN |
| 3 | YELLOW |
| 4 | BLUE |
| 5 | VIOLET |
| 6 | CYAN |
| 7 | WHITE |



Figure 2: 3D Model for RG 8X8 Dot Matrix

The game starts on floor 1, row 0 and column 0. The game is over when you reach floor 7, row 7 and column 7.

## Instructions to the Competitor

### Programming Environment

### C:\Users\USER\AppData\Local\Microsoft\Windows\INetCache\Content.Word\IMG_20171003_175827.jpg

2

1

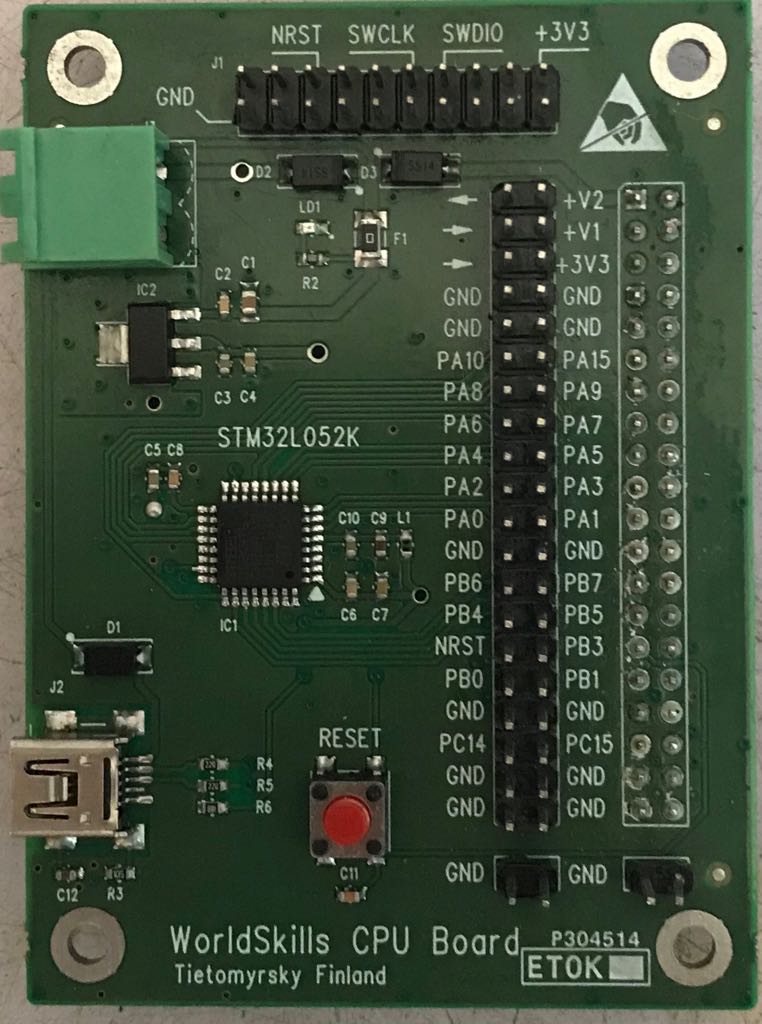
3

Figure 3: Programming setup

|  |  |
| --- | --- |
| 1 | Task Board |
| 2 | USB |
| 3 | World Skills CPU Board |

### Overview of PROGRAMMING ENVIRONMENT

### STM32L052K CPU board



5

4

3

2

1

Figure 4: CPU Board

|  |  |
| --- | --- |
| 1 | 12 V DC |
| 2 | USB Connector |
| 3 | Reset Switch |
| 4 | CPU Connector |
| 5 | Programming Pins |

### TASK board

### C:\Users\USER\Documents\3D labyrinth game\IMG_20171003_173021.jpg

10

9

8

7

6

5

4

3

2

1

Figure 5: Task Board

|  |  |
| --- | --- |
| 1 | 16 X 2 LCD MODULE |
| 2 | BUTTON – UP |
| 3 | BUTTON – DOWN |
| 4 | RG DOT MATRIX |
| 5 | BUTTON – RIGHT |
| 6 | BUTTON – BACKWARD |
| 7 | BUTTON – LEFT |
| 8 | BUTTON – FORWARD |
| 9 | PIEZO – BUZZER |
| 10 | RGB LED |

### Table 2. signals between CPU board and Task board

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Conn.  Pin | CPU  GPIO | Data  Direction | Signal  Name | Notes |
| 11 | PA10 | Output | COL\_LOAD | Load shift register data to outputs |
| 12 | PA15 | Output | COL\_DATA | Shift register serial data input |
| 13 | PA8 | Output | COL\_RST | Reset shift register |
| 14 | PA9 | Output | COL\_CLK | Shift register clock input |
| 15 | PA6 | Output | ROW\_CLK | Increment counter value (count up) |
| 16 | PA7 | Output | ROW\_RST | Reset counter |
| 17 | PA4 | Input | SW5 | Button DOWN |
| 18 | PA5 | Input | SW6 | Button UP |
| 19 | PA2 | Input | SW3 | Button RIGHT |
| 20 | PA3 | Input | SW4 | Button BACKWARD |
| 21 | PA0 | Input | SW1 | Button FORWARD |
| 22 | PA1 | Input | SW2 | Button LEFT |
| 28 | PB5 | Output | PZ | Piezo summer |
| 30 | PB3 | Output | red\_led | LED red |
| 31 | PB0 | Output | blue\_led | LED blue |
| 32 | PB1 | Output | green\_led | LED green |

In table 2, there are all labyrinth board driving signals. Use signal names from table 2 with **HAL\_GPIO\_WritePin** and **HAL\_GPIO\_ReadPin** functions.

Here are examples how to use HAL\_GPIO\_WritePin function:



Next is an example how read with HAL\_GPIO\_ReadPin function:

but = HAL\_GPIO\_ReadPin(BUT\_READ\_GPIO\_Port, BUT\_READ\_Pin);

### Programming Tasks

This programming task will be done in two parts. Before you begin you will be able to see a finished presentation of the task.

You will then get a project file template. In this file all CPU Hardware Abstraction Layer (HAL) and General-Purpose Inputs/Outputs (GPIO) initializations have been done. There are also parts of code where you can find examples on how to use some library functions.

You get a precompiled HEX file to test the functionality of the program and after testing it you will hopefully understand it. Your task is to develop a similar program for the game device. The development will be done in multiple phases

The First part is a hardware dependent phase. Once you get the phase done, call a judge to check that the function performs as asked. Do not proceed to the second part until you have permission from the judge.

When you are done developing your program, return the source file, where you have written your name and other details.

In the second part you will get a new project file. In this project the previous hardware tasks in part-1 have been completed for you.

For both sections, you also receive demonstration files and projects. You can use these projects/files for downloading and viewing/testing for the required functionality of the tasks.

You can either load the demonstration .hex file using ST-LINK Utility

Use ‘File->Open file.’ to load the ‘.hex’ file, and ‘Target->Program & Verify.’ to write the demonstration file to the task board.

Alternatively you can load the demonstration projects into Keil and download the demonstration code onto the task board.

Open the project file in Keil using ‘Project-> Open Project.’ and press the load icon, or use ‘Flash->Download’, or press F8.

### Programming PHASE 1

**Load up the phase1 project in Keil**

**Phase 1.1: Push buttons and piezo-buzzer**

With reference to the data sheets and schematics, complete the following function**.**

Table 3

|  |  |  |  |
| --- | --- | --- | --- |
| **BUTTON SW6(UP) or ‘w’ key of PC Keyboard** | **PIEZO-BUZZER ACTION** | **ON-TIME** | **OFF-TIME** |
| Pressed | Beep twice | 50ms | 50ms |

Table 4

|  |  |
| --- | --- |
| **FUNCTIONS** | **WORKING** |
| static int8\_t CDC\_Receive\_FS (uint8\_t\* Buf, uint32\_t \*Len) | This function is used to receive data from usb device. It is present in the usb\_cdc\_if.c file |
| void beep(void) | This function is used to generate beep sound |

**Phase 1.2: Turn on led’s on the display**

Add a test in the main loop to control the column drivers of 8 x 8 dot matrix, using the functions shown in Table 5.

Table 5

|  |  |
| --- | --- |
| **FUNCTIONS** | **WORKING** |
| void shift\_register\_reset(void) | This function resets all the column driver shift registers. |
| void shift\_register\_clk\_pulse(void) | This function applies one clock pulse to the column shift registers. |
| void shift\_register\_load\_pulse(void) | This function loads the shift register data to its storage register outputs. |
| void shift\_register\_write(uint8\_t data) | This function shifts in 8 bit data to the column shift register. This function should use the three previous shift register functions. |

Test the function with these 8 bit values:

**shift\_register\_write(0xFE); // 1111 1110**

**shift\_register\_write(0xEF); // 1110 1111**

**shift\_register\_write(0xCC); // 1100 1100**

**Phase 1.3 : Turn on any led on the display in a given row**

Add a test in the main loop to control the row drivers of 8 x 8 dot matrix, using the functions shown in Table 6.

Table 6

|  |  |
| --- | --- |
| **FUNCTIONS** | **WORKING** |
| void row\_counter\_reset(void) | This function reset row counter. After reset row 7 on the display is the active row. |
| void row\_counter\_clk\_pulse(void) | This function applies one clock pulse to the row counter. |
| void set\_row(uint8\_t row) | This function selects the row that is given as a parameter. |

Test your code to set a green dot to row 0 column 0 on the display.

**Phase 1.4 : Turn on a led on the display given row, column and color**

Add a test in the main loop to control the row drivers of 8 x 8 dot matrix, using the functions shown in Table 7.

Table 7

|  |  |
| --- | --- |
| **FUNCTIONS** | **WORKING** |
| void clr\_all\_dots(void) | This function turns off all the led’s on the display. |
| void set\_dot(uint8\_t row, uint8\_t col, color\_t color) | This function turns on a led on the display at given row, column and color. |

Test your code to set violet led on row 3, column 4.

**Phase 1.5 : Move led position and change color**

Modify your existing program so that the led can “move” on the display with push buttons OR with **q**(up), **e**(down), **w**(forward), **s**(backward), **a**(Left) & **d**(Right) keys of PC keyboard via virtual COM terminal. If you try to move a led over the display edge, the program should beep twice. You must use switch de-bouncing method. This means if the button is pressed down, the led position will change immediately. If you keep the button pressed down the led position does not change. On releasing the button, led position will not change.

**After you have made this function and tested the code of phase 1, call a judge and show it to them. You will be marked for phase 1 at this time.**

**You can move onto phase 2 without completing phase 1, but you cannot return to complete phase 1 later, and will only receive marks for phase 1 if shown to a judge at this stage.**

### PROGRAMMING PHASE 2

**You can see presentation of finished program by loading the demonstration version of phase 2. You can view this at any time, but please use the correct Project for phase2 for this task.**

### Program Requirements

**In the second phase you need to display the player’s position, number of hits and number of steps.**

Load part 2 project in Keil.

**Phase 2.1 : Position display**

Modify your program so it displays floor number (1…7), row (0…7) and column (0…7) of your position in the labyrinth.





**Phase 2.2 : Labyrinth walls**

The project file you have includes a 3 dimensional array. Each location in the labyrinth has its own cell in the array. The six lower bits of the cell indicate which way you can’t go. They correspond to the walls, floor and roof of the cell.

|  |  |
| --- | --- |
|  | Bit value 0 = you can go this direction  Bit value 1 = you can’t go this direction |

Examples:

If the value of the cell is 0011 1101 = 0x3D, you can go only to the right direction.

If the value of the cell is 0011 0001 = 0x31, you can go from this cell left, backward and right.

If the value of the cell is 0010 1110 = 0x2E, you can go from this cell floor down and forward.

The code below shows the organization of the labyrinth floor arrays.



These images are from Excel file which designed the labyrinth.



Modify your existing program so that every time when you press a direction push buttons OR **q**(up), **e**(down), **w**(forward), **s**(backward), **a**(Left) & **d**(Right) keys of PC, the program checks if the movement is possible or not.

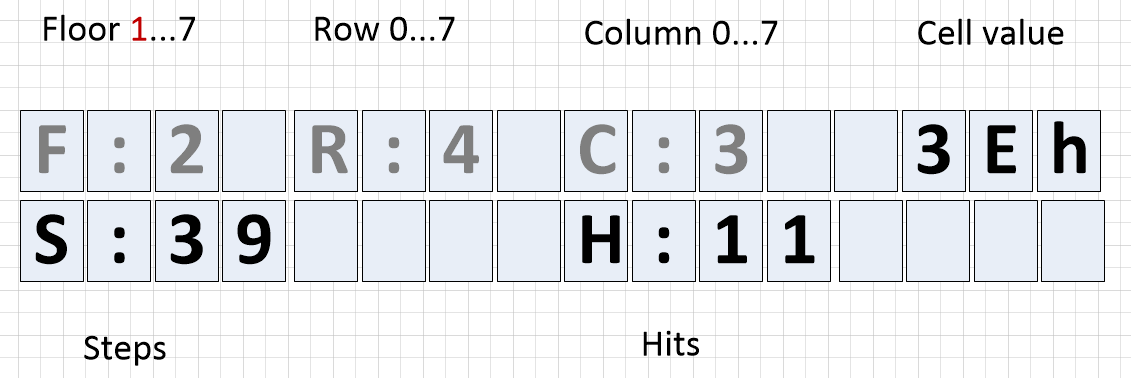
* If the movement is possible, move the LED to the new location.
* If the movement is not possible, beep twice

**Phase 2.3 : Creating the penalty counter, step counter and cell value display**

You need to add a penalty point and a step counter to your program. Every time when a player hits a wall, program increments the penalty counter. Every movement increments the step counter.

Also add “hint display” to show cell value of your current position. Display this information to the LCD display as picture below.

At this stage, every time on press of up or down buttons OR **q** or **e** keys of PC keyboard, the status of Floors, Rows, Columns, Cell Value, Steps and Hits has shown on LCD should be as it is there on virtual COM terminal via USB as COM interface.



**Phase 2.4: Short cut to upper floor**

Add to your program short cut to floor 7. You can use **j** button of your computer to jump floor 7 to any position you chose.

This helps you to test the labyrinth.

**Phase 2.5 : Game over and new game**

Add to your program the “Game Over effect”. The Game Over effect should reward for finishing the game. Be as interesting and creative as you can. You can use piezo and/or LCD display in your show.