# Task 1: Introduction to User Interface

- *(a) Introduction and High-Level Description*
    - The purpose of the task is to use STM32 to read the input from keyboard and print it on the computer. The solution to this requirement is using a virtual serial connection provided by USE connection. And the ANSI terminal standard to help format the printg.
    - The STM32 board is only connected to the computer by a USB wire. The flow chart is attached to the appendix(Figure 12: Flowchart of Task 1)
- *(b) Low Level Description*
    - The basic idea of this task is similar to what students done in its pre-requisite ENGR 2350 embedded control. Putty is used for the connection between the board and computer. All the information needed is in the putty. Using "getchar" for input from keyboard and "printf" function for printing in the Putty.
- *(c) Results and Analysis:*
    - The program is worked as expected.
    - The program can print the visible character in the keyboard, but it cannot print the invisible keys such as backspace, space, etc. This problem is fixed in the task2
    - The result is as followed(Figure 1: Task 1 result).

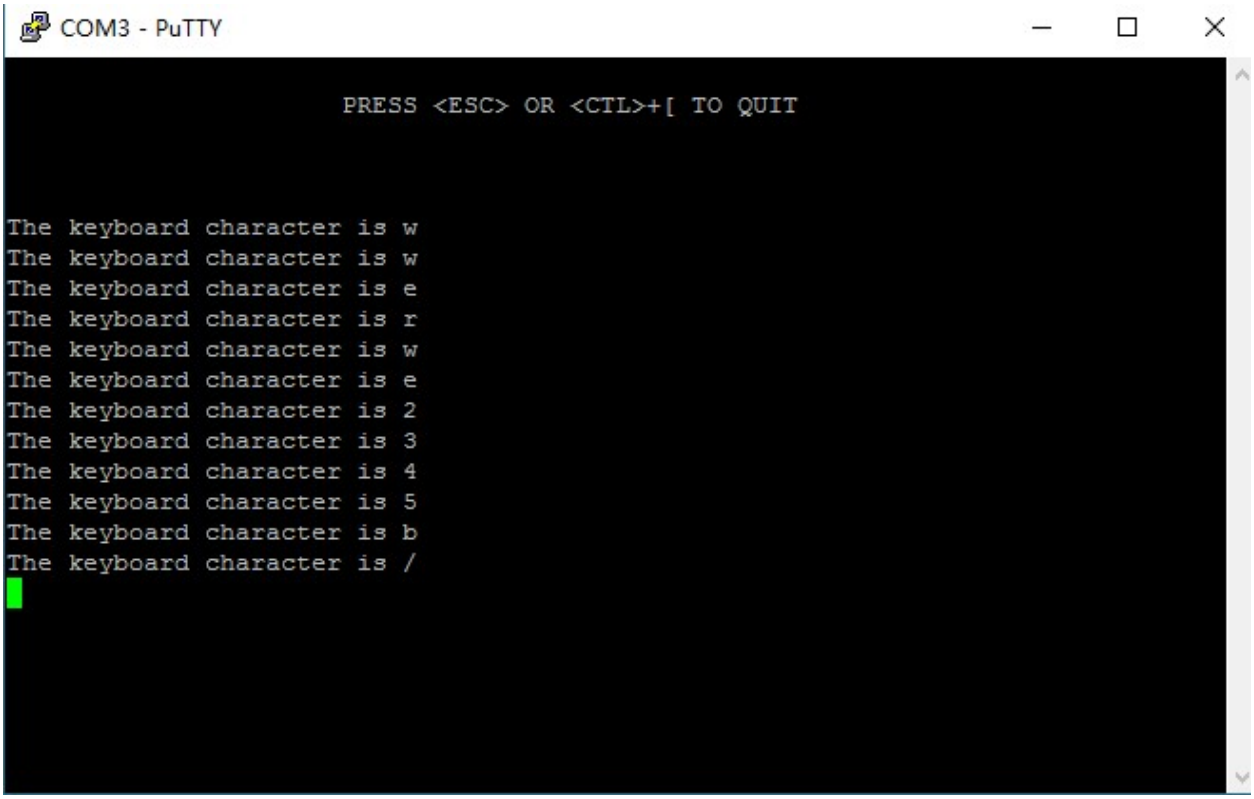


Figure 1: Task 1 result

***Code:***

```
/*
 * Task 1 print the input char
 */
#include "stm32f769xx.h"
#include "hello.h"

#include<stdint.h>

void top_msg(){ // top message
    printf("\n");
    printf("                    PRESS <ESC> OR <CTL>+[ TO QUIT\n");
    printf("\n");
```

```
    printf("\033[6;0H");    // move to line 6
}


int main(void)
{
    Sys_Init();
    top_msg();
    char choice;
    while(1){   //exit the porgramm with ESC or "^+["
        choice = getchar();
        if (choice == 0x1B)
            {return 1;}
        printf("The keyboard character is ");
        putchar(choice);
        printf("\r\n");
    }
}
```

# Task 2: VT100/ANSI Terminal Control Sequences

- *(a) Introduction and High-Level Description*
    - The purpose of this task is to setting the output interface presented to user. In the putty terminal, students need to change the color of the background blue and the change the color of characters to yellow. For inputs from keyboard, the visible character is printed only on line 6 with a red color character of the input. For unprintable ones, they should be printed in AscII code with hexadecimal form beginning from line 12, and when the lines reach the bottom of the terminal, the oldest line should be remove and all the part should be scrolled up. The printed lines for unprintable input should blink.
    - The solution to this task is straight. After the initialization of the putty interface, the program should whether the input character is printable and decide which play it should be printed.
    - The board is only connected to the computer. The flowchart is in the appendix(Figure 13: Flowchart of Task 2)
- *(b) Low Level Description*
    - The difficult part of this task is the correct formatting of the output. The terminal is 24*80. Students cannot move out of this section. All the actions should only in this area.
    - For visible input character, always erase the red character of the input and then print the new one. With the escape sequence such as `printf("\033[0;33;44m");` . students can change the printing style.

- For invisible input character, first set the blinking of the line. For the underline in the line with escape sequence, student did not use the attribute set but [1].
- The scroll up of the exiting printed lines is tricky. The scroll up function need a "\n" to refresh and present in the screen. However if use a "\n" on line 24, the top line will disappear and line 25 show up. The solution here is to move the cursor to line 23 so that the cursor will not go out of the existing area.

- *(c) Results and Analysis:*
  - The program is worked as expected.
  - A difficult problem for me in this problem is to set the whole background blue. It needs to set the background color blue before cleaning the screen.
  - The result is as followed(Figure 2: Task 2 result 1 and Figure 3: Task2 result 2).

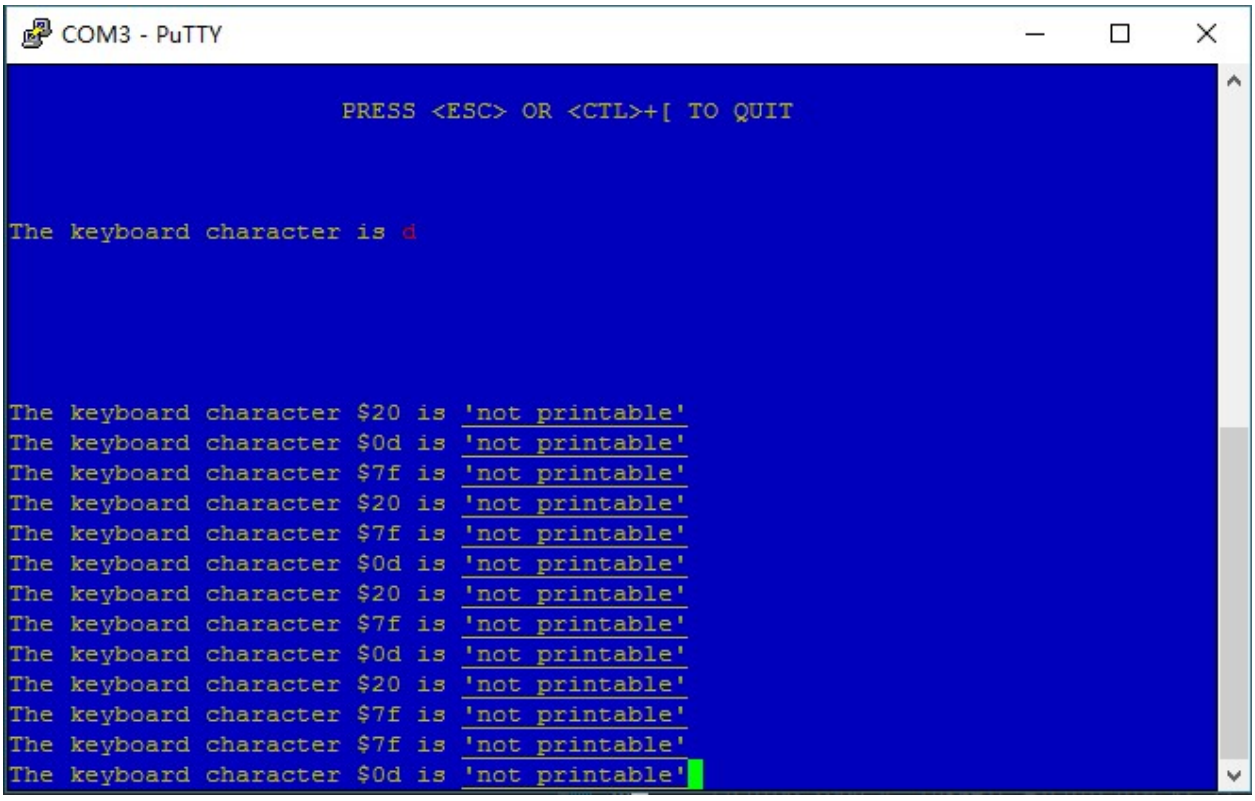


Figure 2: Task 2 result1


Figure 3: Task 2 result2 (blinking)

***Code:***

```
/*
 * Task 2 ANSI Terminal Control Sequences
 */
#include "stm32f769xx.h"
#include "hello.h"

#include<stdint.h>
void red(){ // print char color red
    printf("\033[1;31m");
}

void yellow(){  //print char color yellow
```

```
    printf("\033[1;33m");
}

void top_msg(){ //top message printed
    printf("\n");
    printf("                      PRESS <ESC> OR <CTL>+[ TO QUIT\n");
    printf("\n");
    printf("\033[6;0H");
}

int main(void)
{
    Sys_Init();

    char choice;
    int row = 12;
    int init_char = 0;

    printf("\033[0;33;44m");    //set background blue
    printf("\033[2J\033[;H");

    fflush(stdout);

    top_msg();

    while(1)
    {
        choice = getchar(); //get input
        // quit the programm with ESC pr "^+[""
```

```
        if (choice == 0x1B)
            {return 1;}


        // For the case which the char is not printable
        if(choice < 0x21 || choice > 0x7E){
            if (row < 25){
                printf("\033[5;33;44m");    //set blinking, font and backgrou
nd color
                printf("\033[%d;1H",row);
                if (row < 24){  // when the line is less than 24
                    // set underline
                    printf("The keyboard character $%02x is \e[4m\'not printa
ble\'\e[0m\a\n",choice);
                    row++;
                }
                else{   // when the line is 24
                    printf("The keyboard character $%02x is \e[4m\'not printa
ble\'\e[0m\a",choice);
                    fflush(stdout);
                    row++;
                }
            }
            else{   // when the number of line is larger than 24
                printf("\033[5;33;44m");//set attribute blinking
                printf("\033[12;24r");
                printf("\033M");    //scroll up
                printf("\033[24;1H");
                printf("\033[k");   //erase new line for blue background
                printf("\n"); // activate scroll up and erase
                printf("\033[24;1H");
```

```
                printf("The keyboard character $%02x is \e[4m\'not printable\
'\e[0m\a", choice);
                fflush(stdout);     //print the new line
            }
        }
        else {  //For the case that the character is printable
              if (init_char == 0){  //First time print on the screen
                  printf("\033[6;0H");
                  printf("\033[0;33;44m");
                   printf("\rThe keyboard character is ");
                   red();   // set the char red
                   putchar(choice);
                   yellow();
                   printf("\r\n");
                   init_char = 1;   // set the varibale to indicate the line
is printed once
              }
              else{
                  printf("\033[6;27H"); //change only the input character
                  printf("\033[k");
                  printf("\033[0;31;44m");
                  putchar(choice);
                  printf("\n");
                  printf("\033[0;33;44m");
              }
          }
    }
}
```

# Task 3: Port Input / Output

- *(a) Introduction and High-Level Description*
    - The purpose of this task is to let STM32 read the inputs from several Ports and Pins and make LEDs have different reactions to different inputs. When inputs are set to high by the Analog Discovery board, the corresponding LEDs should be lid. And when inputs are set to low by the Discovery board, the corresponding LEDs should be turned off.
    - The solution to this requirement is to check the status of the input pins. And then make reactions to the inputs.
    - Both Analog Discovery board and STM32 board are connected to the computer. Analog Discovery board is also connected to the D0, D1, D2, and D3 on the STM32 board.
- *(b) Low-Level Description*
    - All the ports which control LEDs should be set to digital output correctly. And the ports corresponding to D0, D1, D2, and D3 should be set to digital inputs with internal pull-up resistors enabled. The “set” state for LED4 is off, and the “reset” state for LED4 is on.
    - The program is in a while loop to check all the input pin status
    - For the register part, students first opened the clock for the corresponding GPIO port. Then set the input, output mode(MODER) and internal resistor mode(PUPDR) by direct change the register bit using the commands in [2, ch. 6.4, pp. 229-230]. Secondly, when the switch in the Analog Discovery board is toggled, the input pin on the STM32 board will read the input logic level by `GPIOx`

`-> IDR`. Then the program will change the status of the corresponding LED by `GPIOJ -> BSRR`[2, ch. 6.4, pp. 230-232].

- For HAL function part, it can be found in HAL function manual [3, ch. 26, pp. 372-375] that the required HAL function to enable the GPIO input/output pins. `HAL_GPIO_Init` is used to define the function of the corresponding pins. `AL_GPIO_ReadPin` and `HAL_GPIO_WritePin` are able to toggle the status of pins connected to the LEDs[3, ch. 26, pp. 374-375]. In the initialization section, the pointer, `GPIO_InitTypeDef,` that initializes the type of each GPIO pin needs to be defined as a global variable so that it can be used when initialization.
- Finally, if the bits are set to high, turn on the LEDs.

- *(c) Results and Analysis*
  - This program works as expected.
  - It took us quite a while to figure out how to check the specific pings without configuring the rest pings in the same port.
  - Here are the results.

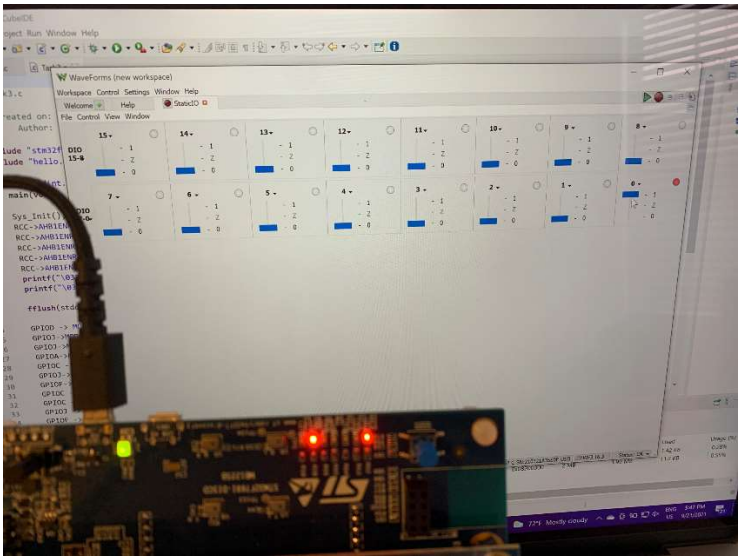

Figure 4: Task 3 Result LED1

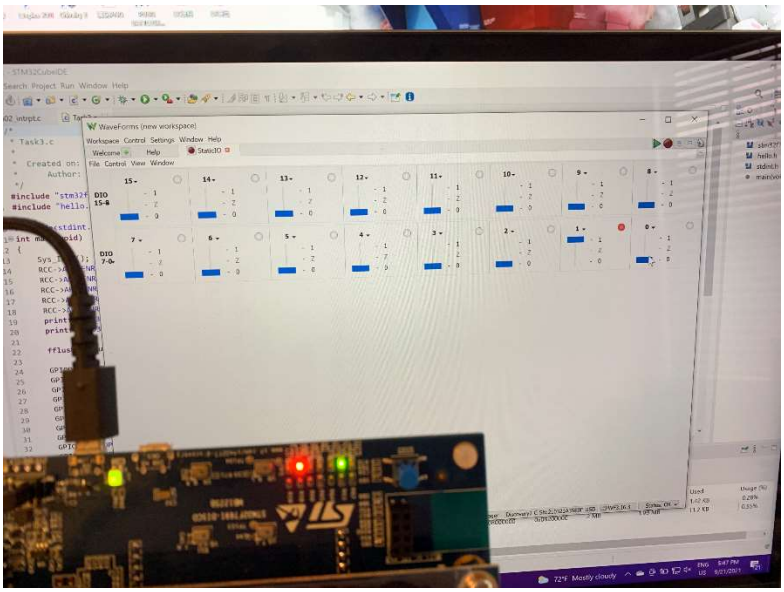

Figure 5: Task 3 Result LED2

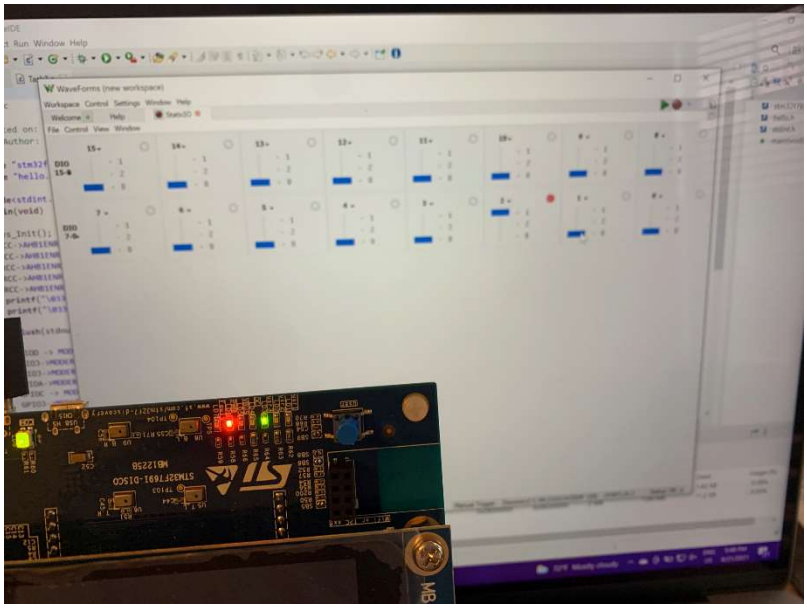

Figure 6: Task 3 Result LED3

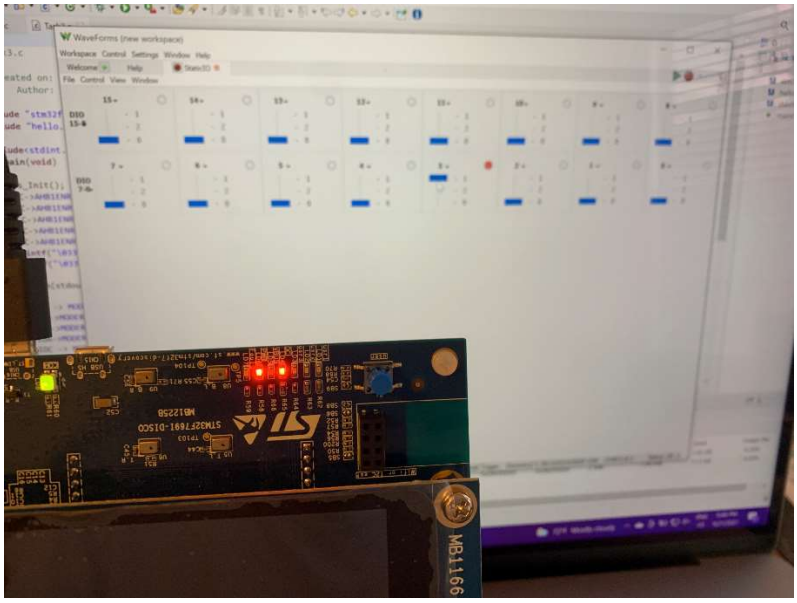

Figure 7: Task 3 Result LED4

***Code (register)***

```
/*
 * Task3.c
 * Register part
 */
#include "stm32f769xx.h"
#include "hello.h"

#include<stdint.h>
int main(void)
{
    Sys_Init();
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIODEN;// Set clock for GPIOD
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;// Set clock for GPIOC
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOJEN;// Set clock for GPIOJ
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;// Set clock for GPIOA
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOFEN;// Set clock for GPIOF
    printf("\033[0;33;44m");
    printf("\033[2J\033[;H"); // Erase screen & move cursor to home position

    fflush(stdout);

    GPIOD -> MODER |= 0x00000100U;// Set D4 to output mode
    GPIOJ->MODER |= 1024U;// Set J5 to output mode
    GPIOJ->MODER |= 67108864U;// Set J13 to output mode
```

```
    GPIOA->MODER |= 16777216U;// Set A12 to output mode
    GPIOC -> MODER &= 0xFFFF3FFFU;// Set C6 & C7 to input mode
    GPIOJ->MODER &= 0xfffffffc;// Set J1 to input mode
    GPIOF->MODER &= 0xffffcfff;// Set F6 to input mode
    GPIOC -> PUPDR |= 0X00004000U;// Set C6 to pull up
    GPIOC -> PUPDR |= 0X00005000U;// Set C7 to pull up
    GPIOJ -> PUPDR |= 0x00000004U;// Set J1 to pull up
    GPIOF -> PUPDR |= 0x00001000U;// Set F6 to pull up
    while(1)
    {
        //printf("Pin state %d\r\n",GPIOC -> IDR &= 0x00000080);
        if(GPIOC -> IDR &= 0x00000080)//led1 control
            GPIOJ -> BSRR = 0x00002000U;//led1 on
        else
            GPIOJ -> BSRR = 0x20000000U;//led1 off
        if(GPIOC -> IDR &= 0x00000040)//led2 control
            GPIOJ -> BSRR = 0x00000020U;//led2 on
        else
            GPIOJ -> BSRR = 0x00200000U;//led2 off
        if(GPIOJ -> IDR &= 0x00000020)//led3 control
            GPIOA -> BSRR = 0x00001000U;//led3 on
        else
            GPIOA -> BSRR = 0x10000000U;//led3 off
        if(GPIOF -> IDR &= 0x00000040)//led4 control
            GPIOD -> BSRR = 0x00100000;//led4 on
        else
            GPIOD -> BSRR = 0x00000010;//led4 off
    }

}
```

***Code (HAL function)***

```
/*
 * task.c
 * Task 3 HAL function part
 *  Created on: Sep 9, 2021
 *      Author: wangy62
 */
```

```c
#include "stm32f769xx.h"
#include "hello.h"

#include<stdint.h>

GPIO_InitTypeDef GPIO_InitStruct;

void LED_GPIO_Init(void);
void INPUT_GPIO_Init(void);



int main(void)
{
    Sys_Init(); // This always goes at the top of main (defined in init.c)
    LED_GPIO_Init();
    INPUT_GPIO_Init();
    printf("\033[2J\033[;H");
    fflush(stdout);
    printf("press ESC to quit.\r\n\n");

    //Set initial state of LEDs
    HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_5, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_SET);// LED4 logical invert
ed reset means set
    HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_13, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_RESET);

    while(1){
        printf("Pin state %d\r\n",HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_7));
        //D0 and LED1
```

```
        if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_7))
            HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_13, GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_13, GPIO_PIN_RESET);
        //D1 and LED2
        if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_6))
            HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_5, GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_5, GPIO_PIN_RESET);
        //D2 and LED3
        if (HAL_GPIO_ReadPin(GPIOJ, GPIO_PIN_1))
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, GPIO_PIN_RESET);
        //D3 and LED4
        if (HAL_GPIO_ReadPin(GPIOF, GPIO_PIN_6))
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_RESET);
        else
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_SET);
    }
}


void LED_GPIO_Init(void){   //Enable GPIO LED output pins
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();
    __HAL_RCC_GPIOJ_CLK_ENABLE();
    // --------------LED1 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
```

```
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
    HAL_GPIO_Init(GPIOJ, &GPIO_InitStruct); //GIOJ13
    // --------------LED2 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_5;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
    HAL_GPIO_Init(GPIOJ, &GPIO_InitStruct); //GPIOJ5
    // --------------LED2 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_12;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct); //GPIOA12
    // ------LED4 initialization------LED4 logical inverted
    GPIO_InitStruct.Pin = GPIO_PIN_4;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct); //GPIOD4
}

void INPUT_GPIO_Init(void){ //Enable GPIO logic input pins
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOF_CLK_ENABLE();
    // --------------D0 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
```

```
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct); //GPIOC7
    // --------------D1 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_6;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct); //GPIOC6
    // --------------D0 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOJ, &GPIO_InitStruct); //GPIOJ1
    // --------------D0 initialization----------------
    GPIO_InitStruct.Pin = GPIO_PIN_6;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOF, &GPIO_InitStruct); //GPIOF6
    }
```

# Task 4: [Depth] Maze

- *(a) Introduction and High-Level Description*
    - The purpose of this lab is to build a maze game in the putty serial. The wall of the maze should be a different color or character. And the player should be represented as a different sign to the wall. When finishing the game, a led is on to indicate the end of the game. And a push button should be connected to the board, which can restart the game.
    - The student's solution is to build the maze with the character "#," and the player's symbol is the cursor in the putty terminal. The player moves in the game with a,s,w, and d corresponding to left, down, up, and right.
    - When the player arrives at the destination, which is a "*" on the screen, the program turns on the led, and the information also shows up on the screen. Then the program will hold until the push-button is pushed. When restarting the game, the player position is reset by the program.
    - The flow chart is in the appendix(Figure 15: Flowchart of task 4).
    - The schematic:

Table 1: Schematic of Task 4

| | Connected to | Connected to |
|---|---|---|
| Push-button | STM32 board D0 pin | Analog Discovery digital pin |
| Ground | STM32 board GND pin | Analog Discovery ground pin |

- *(b) Low-Level Description*
    - To construct the maze, without using the complex data structure, students represent the location of each maze wall by a four-digit decimal number. The first two digits are the location of the column, and the last two digits are the location of the row. All the data is stored in a universal array called maze.
    - A function is constructed for checking if the position is a maze wall. If it is, it will return TRUE. Otherwise, it will return FALSE. However, unlike C, the Boolean type is not defined in the STM32. So it is necessary to define it in the head file. Student adds `”typedef enum {FALSE = 0, TRUE = 1} bool;”` in the head file.
    - Students use the cursor as the player, so there is no need to erase any line of the maze. Player moving in the maze is the moving of the cursor. With ANSI escape sequence, enable the moving of cursor as discussed in Task 2.
    - For the schematic part, the main idea is the same as discussed in Task 3. The program uses the HAL function to fulfill the initialization of GPIOs. For the push-button, it is connected to the Analog Discovery. The program will be held in a while loop until the player pushes the push-button. When the push-button is pressed, the Pin D0 on the board will receive a low logic input(since there is an internal pull-up)
    - The restarting of the game is simple, move the cursor to the beginning point and reset the variables for saving the player position. There is no need to redraw anything in the putty serial.

- *(c) Results and Analysis*
    - The program is worked as expected.
    - Overall the difficult part of this task is the push-button connection. Student first tried to use the user push-button on the board. However, student did not find the related information in the spec sheet. So student connect the board to a push-button them connect it to the analog discovery board.
    - The result is as followed.

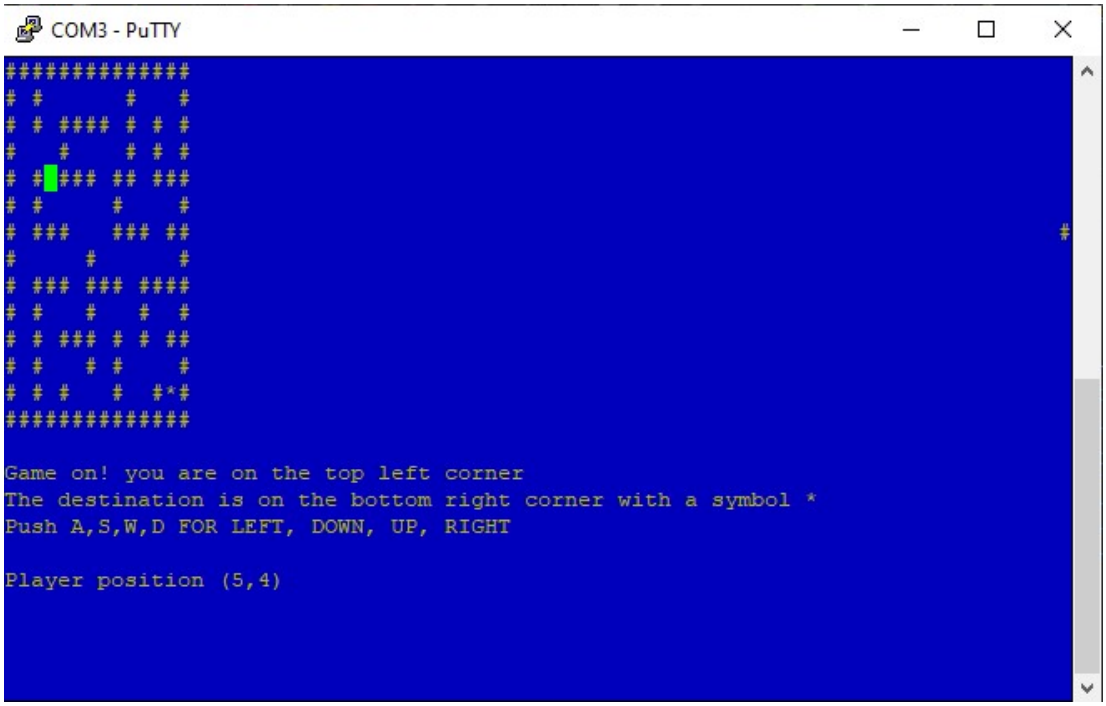


Figure 8: Task 4 Result Running the Game


Figure 9: Task 4 Result Finishing the Maze

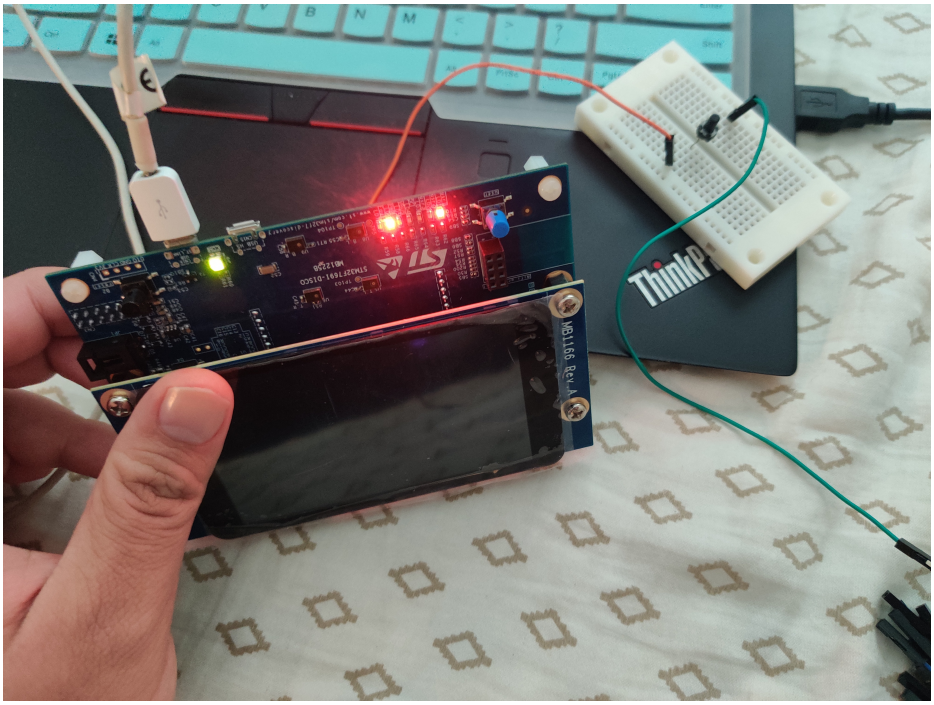

Figure 10: Task 4 Result LED Status After Finishing the Game

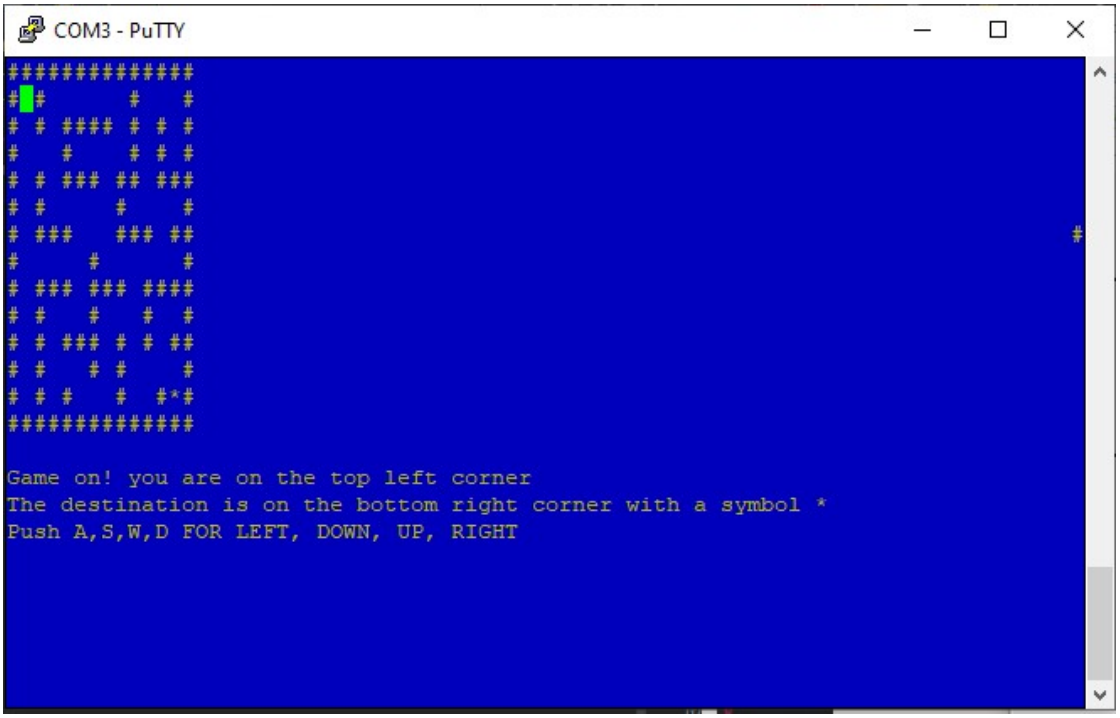


Figure 11: Task 4 Result Restarting the Game

***Code:***

```
/*
 * task4.c
 *
 *  Created on: Sep 14, 2021
 *      Author: wangy62
 */


//-----------------------------------------------------------------------------
// Includes
//-----------------------------------------------------------------------------
#include "stm32f769xx.h"
#include "hello.h"
```

```c
#include<stdint.h>


GPIO_InitTypeDef GPIO_InitStruct;
//set up the wall of maze
int maze[110]={101,102,103,104,105,106,107,108,109,110,111,112,113,114,
        201,214,301,302,303,305,306,307,309,310,311,312,313,314,401,407,
        409,414,501,503,504,505,507,509,511,513,514,601,603,605,611,614,
        701,703,705,706707,708,709,710,711,712,714,801,803,809,814,901,905,
        906,907,909,911,912,913,914,1001,1002,1003,1004,1005,1007,1014,1101,
        1107,1109,1110,1111,1114,1201,1203,1204,1205,1209,1213,1214,1301,1305
,
        1307,1309,1311,1314,1401,1402,1403,1404,1405,1406,1407,1408,1409,1410
,1411,1412,1413,1414};
int maze_x = 0;
int maze_y = 0;

void LED_GPIO_Init(void);
void INPUT_GPIO_Init(void);
void maze_init(void);
bool find_maze(int x, int y, int maze[]);

int main(void)
{
    Sys_Init();

    char choice;
    int player_y=2; //initialization of player position
    int player_x=2;

    printf("\033[0;33;44m");
```

```
    printf("\033[2J\033[;H"); // Erase screen & move cursor to home position
    fflush(stdout);

    LED_GPIO_Init();
    INPUT_GPIO_Init();
    maze_init();

    while(1){

        //input from keyboard
        choice = getchar();

        //player moving up
        if(choice == 0x57 || choice  == 0x77){ // type "a" or "A"
            player_y --;
            if (find_maze(player_y, player_x, maze)){   //Cannot move to that
position
                printf("\033[20;1H");
                printf("\033[K");
                printf("CANNOT MOVE TO THAT POSITION\n");
                player_y ++;
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
            }else{  //player moving up
                printf("\033[20;1H");
                printf("\033[K");
                printf("Player position (%d,%d)\r\n",player_y, player_x);
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
```

```
            }
        }

        //player moving right
        if(choice == 0x41 || choice  == 0x61){  //Cannot move to that positio
n
            player_x --;
            if (find_maze(player_y, player_x, maze)){
                printf("\033[20;1H");
                printf("\033[K");
                printf("CANNOT MOVE TO THAT POSITION\n");
                player_x ++;
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
            }else{  //player moving right
                printf("\033[20;1H");
                printf("\033[K");
                printf("Player position (%d,%d)\r\n",player_y, player_x);
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
            }
        }

        //player moving down
        if(choice == 0x53 || choice  == 0x73){  //Cannot move to that positio
n
            player_y ++;
            if (find_maze(player_y, player_x, maze)){
                printf("\033[20;1H");
                printf("\033[K");
```

```
                printf("CANNOT MOVE TO THAT POSITION\n");
                player_y --;
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
            }else{      //player moving down
                printf("\033[20;1H");
                printf("\033[K");
                printf("Player position (%d,%d)\r\n",player_y, player_x);
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
            }
        }

        //player moving left
        if(choice == 0x44 || choice  == 0x64){
           player_x ++;
           if (find_maze(player_y, player_x, maze)){    //Cannot move to that
position
                printf("\033[20;1H");
                printf("\033[K");
                printf("CANNOT MOVE TO THAT POSITION\n");
                player_x --;
                printf("\033[%d;%dH",player_y, player_x);
                fflush(stdout);
           }else{   //player moving down
                printf("\033[20;1H");
                printf("\033[K");
                printf("Player position (%d,%d)\r\n",player_y, player_x);
                printf("\033[%d;%dH",player_y, player_x);
```

```
                fflush(stdout);
            }
        }

        if (player_y == 13 && player_x == 13){
            printf("\033[20;1H");
            printf("\033[K");
            printf("YOU WIN!!! Push button to restart\n");
            printf("\033[13;13H");
            fflush(stdout);
            HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_13, GPIO_PIN_SET);    //turn on
 the LED1
            //Hold the porgram until the pushbutton pushed
            while(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_7));
            //Restart the maze game
            HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_13, GPIO_PIN_RESET);  //reset L
ED1
            choice = "";
            printf("\033[20;1H");   //remove the player status line
            printf("\033[K");
            printf("\n");
            printf("\033[2;2H");    //reset play location
            fflush(stdout);
            player_y=2;
            player_x=2;
        }
    }
}

//------------------open LED1 for end fo the game-----------------------
```

```
void LED_GPIO_Init(void){
    __HAL_RCC_GPIOJ_CLK_ENABLE();
    GPIO_InitStruct.Pin = GPIO_PIN_13;  //GPIO J13 for LED1
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_MEDIUM;
    HAL_GPIO_Init(GPIOJ, &GPIO_InitStruct);

}

//------------------pushbutton initialization----------------------------
void INPUT_GPIO_Init(void){
    __HAL_RCC_GPIOC_CLK_ENABLE();
    GPIO_InitStruct.Pin = GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}

//------------------Maze initialization----------------------------------
void maze_init(void){
    //construct the maze on screen
    for (int i=0; i<110; i++){
        maze_x = maze[i] % 100; //last 2 digits as row
        maze_y = maze[i] / 100; //first 2 digits as column
        printf("\033[%d;%dH",maze_x,maze_y);
        printf("#");
    }
    //Set destination
```

```
    printf("\033[13;13H");
    printf("*");
    fflush(stdout);
    //Print game information
    printf("\033[16;1H");
    printf("Game on! you are on the top left corner\r\n");
    printf("The destination is on the bottom right corner with a symbol *\r\n
");
    printf("Push A,S,W,D FOR LEFT, DOWN, UP, RIGHT\r\n");
    printf("\033[2;2H");
    fflush(stdout);
    //Turn off Led1
    HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_13, GPIO_PIN_RESET);
}


//find if the position is a maze wall if it si return TRUE ,if it is not retu
rn FALSE
bool find_maze(int x, int y, int maze[]){
    maze_x = 0;
    maze_y = 0;
    for (int i=0; i<110; i++){
        maze_x = maze[i] % 100;
        maze_y = maze[i] / 100;
        if (maze_x == x && maze_y == y){
            return TRUE;
        }
    }
    return FALSE;
}
```

# Conclusion

Through Task 1, we got a basic idea about the user interface. We learned how to read the input from the keyboard and how to print it out through the terminal. And by completing this task, we can also find out how to use escape codes. For task 2, we learned how to modify the display color for the characters and backgrounds. by using “if” functions, we can now let the program determine whether the input is printable or not.

In task 3, the code which adjusts input / output mode was used. Also, we found out how to only read and test the specified pins and leave the rest pins to remain the same. And finally for task 4, we combined everything we learned from the previous tasks to set up a maze. Users can use WASD keys to navigate the “player”. And when the “player” reaches the goal, the LED will be triggered.

The materials we learned from task 1 can be used to create interfaces. Such as the ones users will always see when they try to type their username or password. It can show what has been typed in and how much space is left for them to keep typing. The GPIO pins can allow people to read and write the logic state with the STM32 board. With this function, it is possible to control the switches by a terminal such as touching screen on the STM32 board. This function can be used in the industry where the switches are unsafe to be operated by humans at close distances. With the help of the board, it can be much safer for humans to operate dangerous machines.

# Reference

[1] egmont, “How to Do: Underline, bold, Italic, strikethrough, COLOR, background, and size in Gnome terminal?,” *Ask Ubuntu*, 01-Feb-1963. [Online]. Available: https://askubuntu.com/questions/528928/how-to-do-underline-bold-italic-strikethrough-color-background-and-size-i. [Accessed: 22-Sep-2021].

[2] *RM0410 Reference manual,* STMicroelectronics, NV, USA, 2018, pp. 229-232.

[3] *UM1905 User Manual,* STMicroelectronics, NV, USA, 2018, pp. 372-375.

# *Appendix*

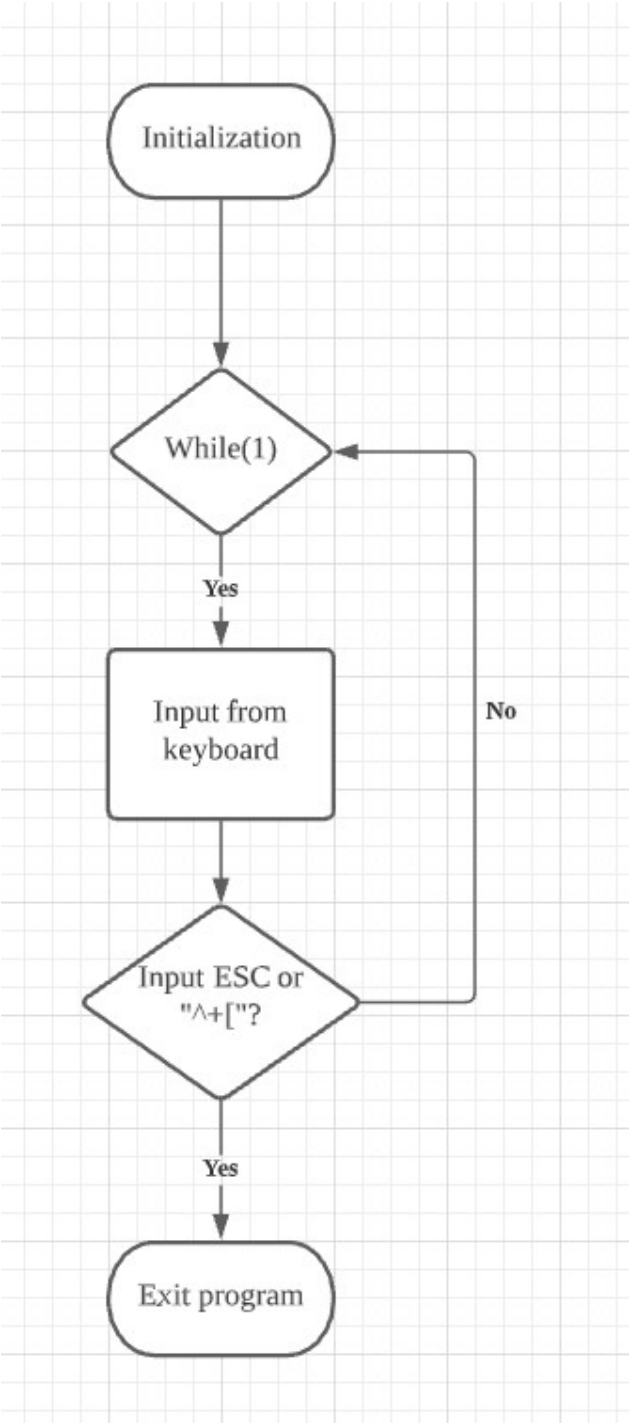


Figure 12: Flowchart of task 1

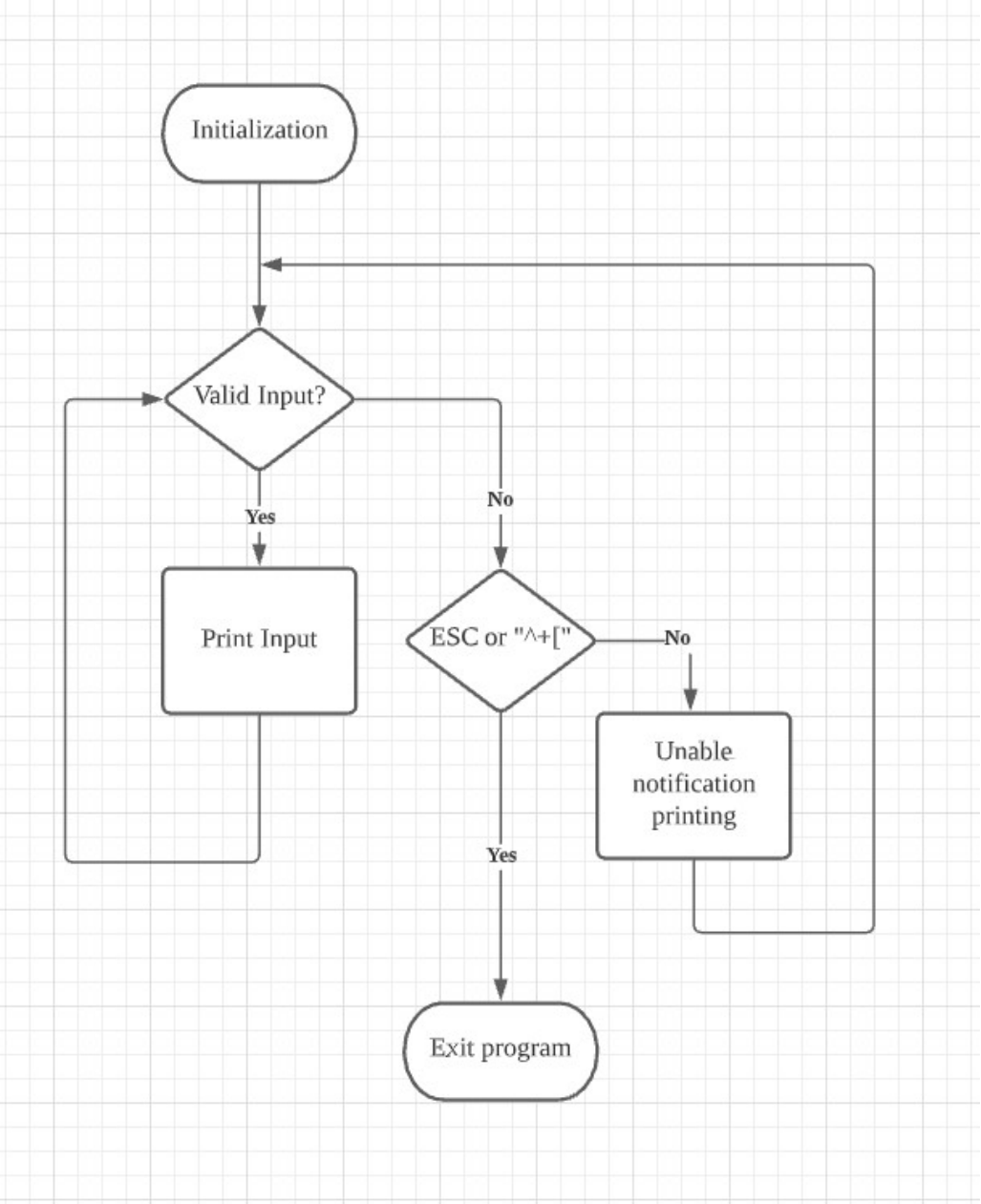


Figure 13: Flowchart of Task 2

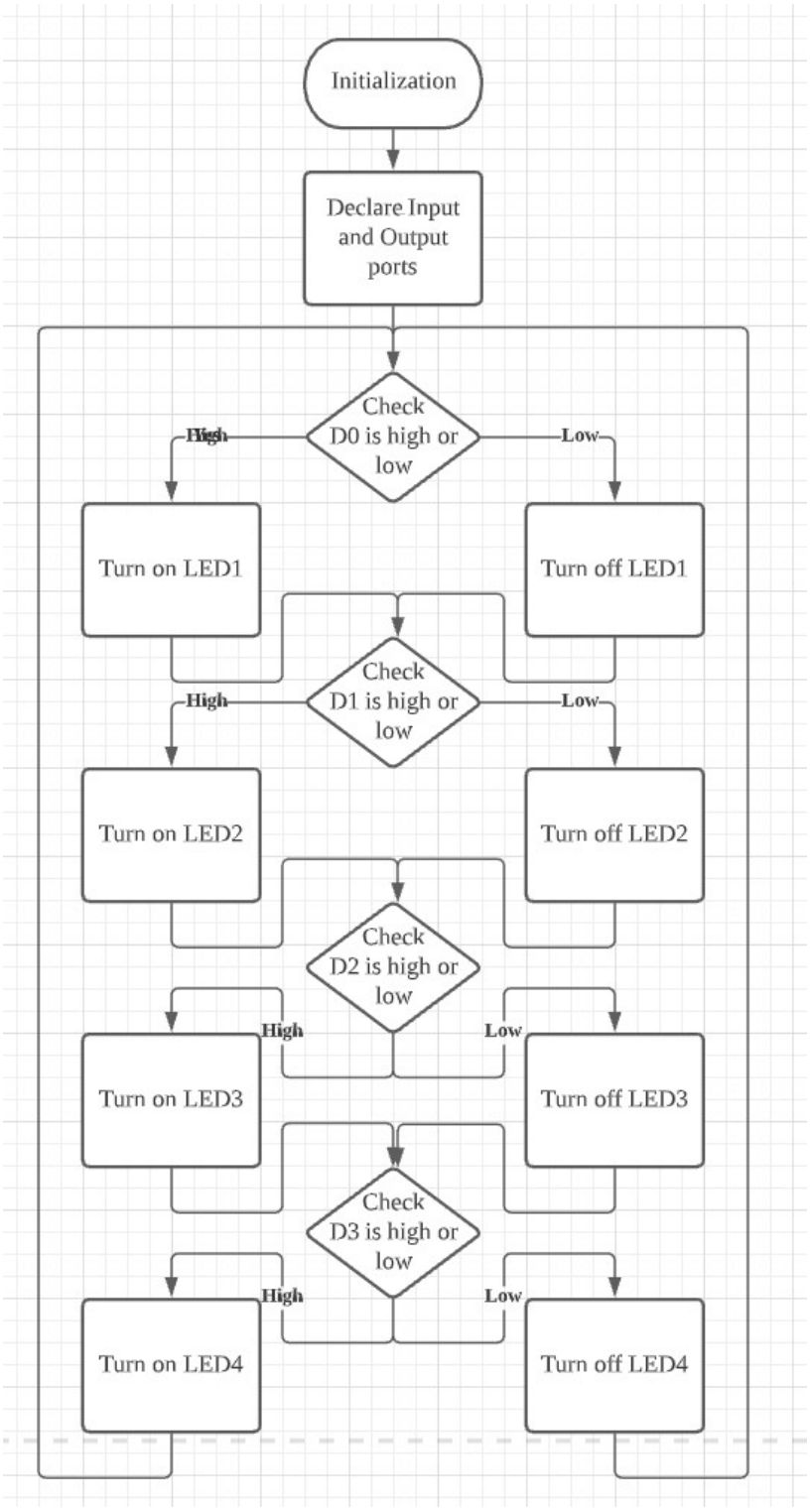


Figure 14: Flowchart of Task 3

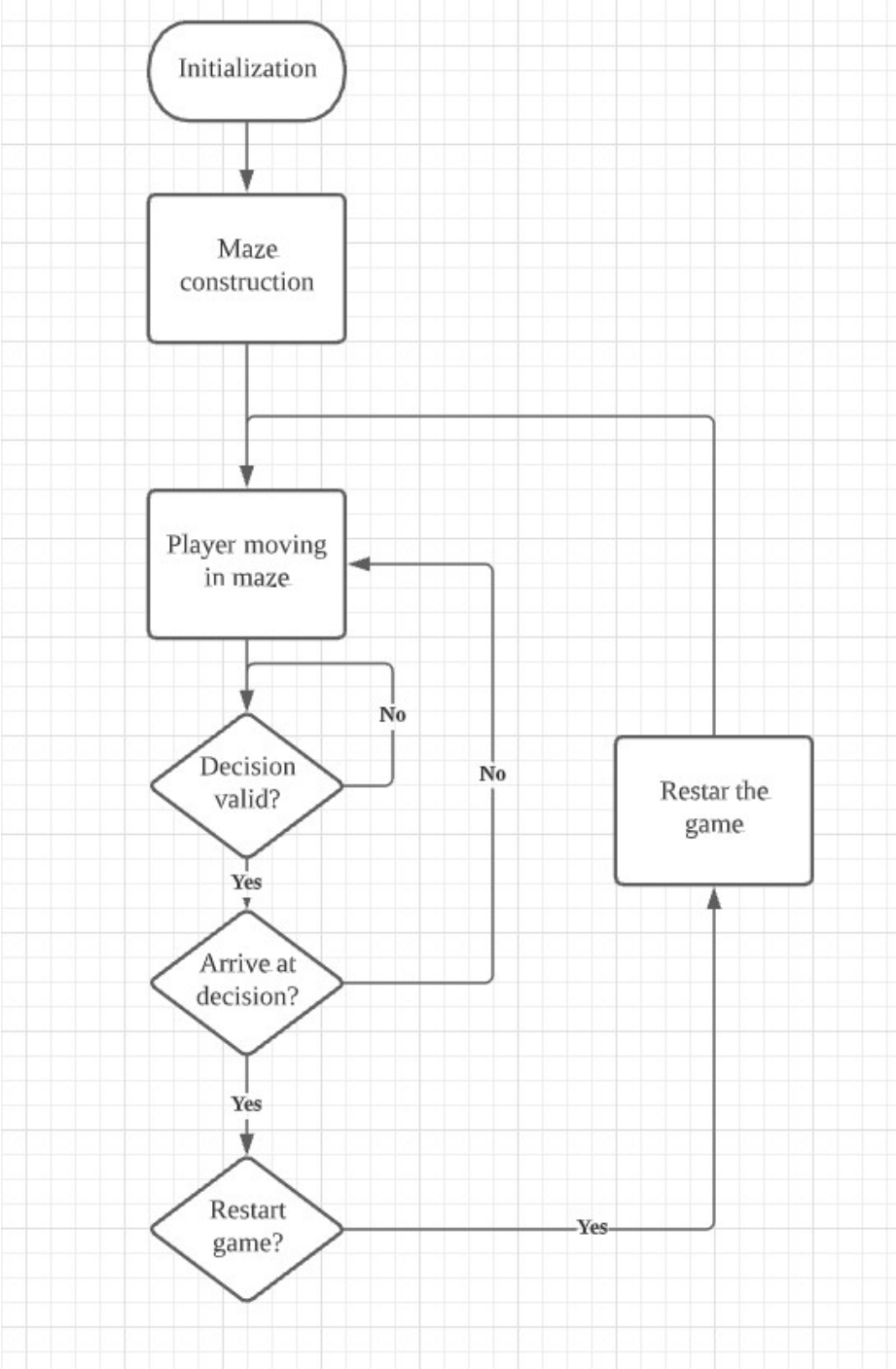


Figure 15: Flowchart of Task 4