|  |  |
| --- | --- |
| Министерство образования Республики Беларусь | |
| Учреждение образования | |
| БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ | |
| ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | |
|  | |
|  | |
| Факультет компьютерных систем и сетей | |
| Кафедра программного обеспечения информационных технологий | |
| Дисциплина: ПОВС (Программное обеспечение встроенных систем) | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
| **ОТЧЁТ** | |
| по лабораторной работе № 2 | |
|  | |
| Вариант «Морской бой» | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
| Выполнили: | Киселёв А.С. |
|  |
|  | гр. 251004 |
|  |  |
| Проверил: | Деменковец Д. В. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| Минск 2025 | |

Исходный код программы

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* Copyright (c) 2025 STMicroelectronics.

\* All rights reserved.

\*

\* This software is licensed under terms that can be found in the LICENSE file

\* in the root directory of this software component.

\* If no LICENSE file comes with this software, it is provided AS-IS.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include <string.h>

#include <stdlib.h>

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

typedef enum {

SEG\_A = 0,

SEG\_B,

SEG\_C,

SEG\_D,

SEG\_E,

SEG\_F,

SEG\_G

} segment\_t;

typedef enum {

DIR\_UP = 0,

DIR\_DOWN,

DIR\_LEFT,

DIR\_RIGHT

} direction\_t;

typedef enum {

TRANS\_SAME\_DIGIT = 0,

TRANS\_NEXT\_DIGIT,

TRANS\_PREV\_DIGIT

} transition\_type\_t;

typedef struct {

segment\_t segment;

direction\_t direction;

transition\_type\_t trans\_type;

} transition\_t;

typedef struct {

uint8\_t digit;

segment\_t segment;

direction\_t direction;

uint8\_t alive;

} ship\_t;

typedef struct {

uint8\_t active;

uint8\_t digit;

uint32\_t start\_time;

} shot\_t;

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

#define SHIP\_MOVE\_INTERVAL\_MS 150

#define MAX\_TRANSITIONS 5

#define RESPAWN\_DELAY\_MS 2000

#define SHOT\_DURATION\_MS 300

#define SHOT\_FRAME\_MS 150

#define SHOT\_COOLDOWN\_MS (SHOT\_DURATION\_MS \* 2)

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

static uint32\_t last\_refresh\_tick = 0;

static uint8\_t current\_digit = 0;

volatile uint8\_t disp\_buf[4] = {0,0,0,0};

ship\_t ship = {0, SEG\_A, DIR\_RIGHT, 1};

uint32\_t last\_ship\_move = 0;

shot\_t shot = {0, 0, 0};

uint32\_t last\_shot\_time = 0;

uint8\_t cannon\_position = 0; // 0-3

uint8\_t score = 0; // (0-7)

volatile uint32\_t last\_button1\_press = 0;

volatile uint32\_t last\_button2\_press = 0;

volatile uint32\_t last\_button3\_press = 0;

// Transition table

typedef struct {

segment\_t from\_seg;

direction\_t from\_dir;

transition\_t transitions[MAX\_TRANSITIONS];

uint8\_t count;

} transition\_rule\_t;

static const transition\_rule\_t transition\_table[] = {

// a, RIGHT

{SEG\_A, DIR\_RIGHT, {{SEG\_B, DIR\_DOWN, TRANS\_SAME\_DIGIT},

{SEG\_A, DIR\_RIGHT, TRANS\_NEXT\_DIGIT},

{SEG\_F, DIR\_DOWN, TRANS\_NEXT\_DIGIT}}, 3},

// a, LEFT

{SEG\_A, DIR\_LEFT, {{SEG\_F, DIR\_DOWN, TRANS\_SAME\_DIGIT},

{SEG\_A, DIR\_LEFT, TRANS\_PREV\_DIGIT},

{SEG\_B, DIR\_DOWN, TRANS\_PREV\_DIGIT}}, 3},

// b, UP

{SEG\_B, DIR\_UP, {{SEG\_A, DIR\_LEFT, TRANS\_SAME\_DIGIT},

{SEG\_A, DIR\_RIGHT, TRANS\_NEXT\_DIGIT}}, 2},

// b, DOWN

{SEG\_B, DIR\_DOWN, {{SEG\_G, DIR\_LEFT, TRANS\_SAME\_DIGIT},

{SEG\_C, DIR\_DOWN, TRANS\_SAME\_DIGIT},

{SEG\_G, DIR\_RIGHT, TRANS\_NEXT\_DIGIT},

{SEG\_E, DIR\_DOWN, TRANS\_NEXT\_DIGIT}}, 4},

// g, RIGHT

{SEG\_G, DIR\_RIGHT, {{SEG\_B, DIR\_UP, TRANS\_SAME\_DIGIT},

{SEG\_C, DIR\_DOWN, TRANS\_SAME\_DIGIT},

{SEG\_F, DIR\_UP, TRANS\_NEXT\_DIGIT},

{SEG\_E, DIR\_DOWN, TRANS\_NEXT\_DIGIT},

{SEG\_G, DIR\_RIGHT, TRANS\_NEXT\_DIGIT}}, 5},

// g, LEFT

{SEG\_G, DIR\_LEFT, {{SEG\_F, DIR\_UP, TRANS\_SAME\_DIGIT},

{SEG\_E, DIR\_DOWN, TRANS\_SAME\_DIGIT},

{SEG\_G, DIR\_LEFT, TRANS\_PREV\_DIGIT},

{SEG\_C, DIR\_DOWN, TRANS\_PREV\_DIGIT},

{SEG\_B, DIR\_UP, TRANS\_PREV\_DIGIT}}, 5},

// c, UP

{SEG\_C, DIR\_UP, {{SEG\_G, DIR\_LEFT, TRANS\_SAME\_DIGIT},

{SEG\_B, DIR\_UP, TRANS\_SAME\_DIGIT},

{SEG\_G, DIR\_RIGHT, TRANS\_NEXT\_DIGIT},

{SEG\_F, DIR\_UP, TRANS\_NEXT\_DIGIT}}, 4},

// c, DOWN

{SEG\_C, DIR\_DOWN, {{SEG\_D, DIR\_LEFT, TRANS\_SAME\_DIGIT},

{SEG\_D, DIR\_RIGHT, TRANS\_NEXT\_DIGIT}}, 2},

// d, RIGHT

{SEG\_D, DIR\_RIGHT, {{SEG\_C, DIR\_UP, TRANS\_SAME\_DIGIT},

{SEG\_D, DIR\_RIGHT, TRANS\_NEXT\_DIGIT},

{SEG\_E, DIR\_UP, TRANS\_NEXT\_DIGIT}}, 3},

// d, LEFT

{SEG\_D, DIR\_LEFT, {{SEG\_E, DIR\_UP, TRANS\_SAME\_DIGIT},

{SEG\_D, DIR\_LEFT, TRANS\_PREV\_DIGIT},

{SEG\_C, DIR\_UP, TRANS\_PREV\_DIGIT}}, 3},

// e, UP

{SEG\_E, DIR\_UP, {{SEG\_G, DIR\_RIGHT, TRANS\_SAME\_DIGIT},

{SEG\_F, DIR\_UP, TRANS\_SAME\_DIGIT},

{SEG\_G, DIR\_LEFT, TRANS\_PREV\_DIGIT},

{SEG\_B, DIR\_UP, TRANS\_PREV\_DIGIT}}, 4},

// e, DOWN

{SEG\_E, DIR\_DOWN, {{SEG\_D, DIR\_RIGHT, TRANS\_SAME\_DIGIT},

{SEG\_D, DIR\_LEFT, TRANS\_PREV\_DIGIT}}, 2},

// f, DOWN

{SEG\_F, DIR\_DOWN, {{SEG\_G, DIR\_RIGHT, TRANS\_SAME\_DIGIT},

{SEG\_E, DIR\_DOWN, TRANS\_SAME\_DIGIT},

{SEG\_G, DIR\_LEFT, TRANS\_PREV\_DIGIT},

{SEG\_C, DIR\_DOWN, TRANS\_PREV\_DIGIT}}, 4},

// f, UP

{SEG\_F, DIR\_UP, {{SEG\_A, DIR\_RIGHT, TRANS\_SAME\_DIGIT},

{SEG\_A, DIR\_LEFT, TRANS\_PREV\_DIGIT}}, 2},

};

static const uint8\_t transition\_table\_size = sizeof(transition\_table) / sizeof(transition\_rule\_t);

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

/\* USER CODE BEGIN PFP \*/

static void shift\_out\_byte(uint8\_t b);

static void latch\_pulse(void);

static void display\_refresh\_cycle(void);

static void update\_display(void);

static void move\_ship(void);

static void spawn\_ship(void);

static void update\_score\_leds(void);

static void fire\_cannon(void);

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

static void shift\_out\_byte(uint8\_t b)

{

for (int i = 7; i >= 0; --i)

{

uint8\_t bit = (b >> i) & 0x1;

HAL\_GPIO\_WritePin(DATA\_DISP\_GPIO\_Port, DATA\_DISP\_Pin, bit ? GPIO\_PIN\_SET : GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(CLK\_DISP\_GPIO\_Port, CLK\_DISP\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(CLK\_DISP\_GPIO\_Port, CLK\_DISP\_Pin, GPIO\_PIN\_RESET);

}

}

static void latch\_pulse(void)

{

HAL\_GPIO\_WritePin(LATCH\_DISP\_GPIO\_Port, LATCH\_DISP\_Pin, GPIO\_PIN\_SET);

\_\_NOP();

HAL\_GPIO\_WritePin(LATCH\_DISP\_GPIO\_Port, LATCH\_DISP\_Pin, GPIO\_PIN\_RESET);

}

static void update\_display(void)

{

for (int i = 0; i < 4; i++)

{

disp\_buf[i] = 0xFF;

}

if (shot.active)

{

uint32\_t now = HAL\_GetTick();

uint32\_t elapsed = now - shot.start\_time;

if (elapsed < SHOT\_DURATION\_MS)

{

uint8\_t frame = (elapsed / SHOT\_FRAME\_MS) % 2;

uint8\_t shot\_segments = 0;

if (frame == 0)

{

shot\_segments = (1 << SEG\_G) | (1 << SEG\_C) | (1 << SEG\_D) | (1 << SEG\_E);

disp\_buf[shot.digit] = ~shot\_segments;

if (ship.alive && ship.digit == shot.digit)

{

if (ship.segment == SEG\_G || ship.segment == SEG\_C ||

ship.segment == SEG\_D || ship.segment == SEG\_E)

{

ship.alive = 0;

score = (score + 1) % 8;

update\_score\_leds();

}

}

}

else

{

shot\_segments = (1 << SEG\_A) | (1 << SEG\_B) | (1 << SEG\_G) | (1 << SEG\_F);

disp\_buf[shot.digit] = ~shot\_segments;

if (ship.alive && ship.digit == shot.digit)

{

if (ship.segment == SEG\_A || ship.segment == SEG\_B ||

ship.segment == SEG\_G || ship.segment == SEG\_F)

{

ship.alive = 0;

score = (score + 1) % 8;

update\_score\_leds();

}

}

}

}

else

{

shot.active = 0;

}

}

// ship

if (ship.alive)

{

if (!shot.active || shot.digit != ship.digit)

{

uint8\_t segment\_mask = (1 << ship.segment);

disp\_buf[ship.digit] = ~segment\_mask;

}

else

{

uint32\_t now = HAL\_GetTick();

uint32\_t elapsed = now - shot.start\_time;

uint8\_t frame = (elapsed / SHOT\_FRAME\_MS) % 2;

if ((elapsed % SHOT\_FRAME\_MS) < (SHOT\_FRAME\_MS / 4))

{

uint8\_t segment\_mask = (1 << ship.segment);

disp\_buf[ship.digit] = ~segment\_mask;

}

}

}

// cannon

disp\_buf[cannon\_position] &= ~(1 << 7);

}

static void spawn\_ship(void)

{

uint8\_t attempts = 0;

ship.digit = rand() % 4;

ship.segment = rand() % 7;

while (attempts < 50)

{

direction\_t dirs[] = {DIR\_UP, DIR\_DOWN, DIR\_LEFT, DIR\_RIGHT};

ship.direction = dirs[rand() % 4];

const transition\_rule\_t\* rule = NULL;

for (uint8\_t i = 0; i < transition\_table\_size; i++)

{

if (transition\_table[i].from\_seg == ship.segment &&

transition\_table[i].from\_dir == ship.direction)

{

rule = &transition\_table[i];

break;

}

}

if (rule != NULL && rule->count > 0)

{

ship.alive = 1;

return;

}

attempts++;

}

ship.digit = 1;

ship.segment = SEG\_A;

ship.direction = DIR\_RIGHT;

ship.alive = 1;

}

static void move\_ship(void)

{

if (!ship.alive)

return;

const transition\_rule\_t\* rule = NULL;

for (uint8\_t i = 0; i < transition\_table\_size; i++)

{

if (transition\_table[i].from\_seg == ship.segment &&

transition\_table[i].from\_dir == ship.direction)

{

rule = &transition\_table[i];

break;

}

}

/\*

if (rule == NULL || rule->count == 0)

return;

\*/

/\*

uint8\_t valid\_count = 0;

uint8\_t valid\_indices[MAX\_TRANSITIONS];

for (uint8\_t i = 0; i < rule->count; i++)

{

transition\_t trans = rule->transitions[i];

if (trans.trans\_type == TRANS\_NEXT\_DIGIT && ship.digit >= 3)

{

valid\_indices[valid\_count++] = i;

}

else if (trans.trans\_type == TRANS\_PREV\_DIGIT && ship.digit == 0)

{

valid\_indices[valid\_count++] = i;

}

else

{

valid\_indices[valid\_count++] = i;

}

}

if (valid\_count == 0)

return;

uint8\_t choice = valid\_indices[rand() % valid\_count];

transition\_t next = rule->transitions[choice];

\*/

uint8\_t choice = rand() % rule->count;

transition\_t next = rule->transitions[choice];

ship.segment = next.segment;

ship.direction = next.direction;

if (next.trans\_type == TRANS\_NEXT\_DIGIT)

{

ship.digit++;

if (ship.digit > 3) ship.digit = 0;

}

else if (next.trans\_type == TRANS\_PREV\_DIGIT)

{

if (ship.digit == 0)

ship.digit = 3;

else

ship.digit--;

}

}

static void fire\_cannon(void)

{

uint32\_t now = HAL\_GetTick();

if (now - last\_shot\_time < SHOT\_COOLDOWN\_MS)

return; // cooldown

shot.active = 1;

shot.digit = cannon\_position;

shot.start\_time = now;

last\_shot\_time = now;

}

static void update\_score\_leds(void)

{

if (score & 0x01)

HAL\_GPIO\_WritePin(LED\_D10\_GPIO\_Port, LED\_D10\_Pin, GPIO\_PIN\_RESET);

else

HAL\_GPIO\_WritePin(LED\_D10\_GPIO\_Port, LED\_D10\_Pin, GPIO\_PIN\_SET);

if (score & 0x02)

HAL\_GPIO\_WritePin(LED\_D11\_GPIO\_Port, LED\_D11\_Pin, GPIO\_PIN\_RESET);

else

HAL\_GPIO\_WritePin(LED\_D11\_GPIO\_Port, LED\_D11\_Pin, GPIO\_PIN\_SET);

if (score & 0x04)

HAL\_GPIO\_WritePin(LED\_D12\_GPIO\_Port, LED\_D12\_Pin, GPIO\_PIN\_RESET);

else

HAL\_GPIO\_WritePin(LED\_D12\_GPIO\_Port, LED\_D12\_Pin, GPIO\_PIN\_SET);

}

static void display\_refresh\_cycle(void)

{

uint32\_t now = HAL\_GetTick();

if ((now - last\_refresh\_tick) >= 1)

{

uint8\_t segments = disp\_buf[current\_digit];

uint8\_t digit\_mask = (1 << current\_digit);

shift\_out\_byte(segments);

shift\_out\_byte(digit\_mask);

latch\_pulse();

current\_digit++;

if (current\_digit >= 4) current\_digit = 0;

last\_refresh\_tick = now;

}

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

/\* USER CODE BEGIN 2 \*/

HAL\_GPIO\_WritePin(CLK\_DISP\_GPIO\_Port, CLK\_DISP\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(DATA\_DISP\_GPIO\_Port, DATA\_DISP\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(LATCH\_DISP\_GPIO\_Port, LATCH\_DISP\_Pin, GPIO\_PIN\_RESET);

HAL\_GPIO\_WritePin(LED\_D10\_GPIO\_Port, LED\_D10\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(LED\_D11\_GPIO\_Port, LED\_D11\_Pin, GPIO\_PIN\_SET);

HAL\_GPIO\_WritePin(LED\_D12\_GPIO\_Port, LED\_D12\_Pin, GPIO\_PIN\_SET);

uint32\_t seed = HAL\_GetTick();

if (seed == 0) seed = 42;

srand(seed);

spawn\_ship();

cannon\_position = 0;

score = 0;

update\_display();

update\_score\_leds();

last\_ship\_move = HAL\_GetTick();

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

uint32\_t now = HAL\_GetTick();

if (ship.alive && (now - last\_ship\_move >= SHIP\_MOVE\_INTERVAL\_MS))

{

move\_ship();

update\_display();

last\_ship\_move = now;

}

if (!ship.alive && (now - last\_ship\_move >= RESPAWN\_DELAY\_MS))

{

spawn\_ship();

update\_display();

last\_ship\_move = now;

}

if (shot.active)

{

update\_display();

}

display\_refresh\_cycle();

}

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSI;

RCC\_OscInitStruct.HSIState = RCC\_HSI\_ON;

RCC\_OscInitStruct.HSICalibrationValue = RCC\_HSICALIBRATION\_DEFAULT;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_NONE;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_HSI;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV1;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV1;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_0) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* USER CODE BEGIN MX\_GPIO\_Init\_1 \*/

/\* USER CODE END MX\_GPIO\_Init\_1 \*/

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOA, CLK\_DISP\_Pin|DATA\_DISP\_Pin|LED\_D12\_Pin|LED\_D11\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(USER\_BUZZER\_GPIO\_Port, USER\_BUZZER\_Pin, GPIO\_PIN\_SET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(LATCH\_DISP\_GPIO\_Port, LATCH\_DISP\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(LED\_D10\_GPIO\_Port, LED\_D10\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pins : CANON1\_Pin CANON2\_Pin \*/

GPIO\_InitStruct.Pin = CANON1\_Pin|CANON2\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : CANON3\_Pin \*/

GPIO\_InitStruct.Pin = CANON3\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_IT\_RISING;

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

HAL\_GPIO\_Init(CANON3\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : CLK\_DISP\_Pin DATA\_DISP\_Pin LED\_D12\_Pin LED\_D11\_Pin \*/

GPIO\_InitStruct.Pin = CLK\_DISP\_Pin|DATA\_DISP\_Pin|LED\_D12\_Pin|LED\_D11\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOA, &GPIO\_InitStruct);

/\*Configure GPIO pin : USER\_BUZZER\_Pin \*/

GPIO\_InitStruct.Pin = USER\_BUZZER\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_OD;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(USER\_BUZZER\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : LATCH\_DISP\_Pin \*/

GPIO\_InitStruct.Pin = LATCH\_DISP\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(LATCH\_DISP\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : LED\_D10\_Pin \*/

GPIO\_InitStruct.Pin = LED\_D10\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(LED\_D10\_GPIO\_Port, &GPIO\_InitStruct);

/\* EXTI interrupt init\*/

HAL\_NVIC\_SetPriority(EXTI0\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(EXTI0\_IRQn);

HAL\_NVIC\_SetPriority(EXTI1\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(EXTI1\_IRQn);

HAL\_NVIC\_SetPriority(EXTI4\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(EXTI4\_IRQn);

/\* USER CODE BEGIN MX\_GPIO\_Init\_2 \*/

/\* USER CODE END MX\_GPIO\_Init\_2 \*/

}

/\* USER CODE BEGIN 4 \*/

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

{

uint32\_t now = HAL\_GetTick();

if (GPIO\_Pin == CANON1\_Pin)

{

// left

if (now - last\_button1\_press > 200)

{

if (cannon\_position > 0)

{

cannon\_position--;

update\_display();

}

last\_button1\_press = now;

}

}

else if (GPIO\_Pin == CANON2\_Pin)

{

// right

if (now - last\_button2\_press > 200)

{

if (cannon\_position < 3)

{

cannon\_position++;

update\_display();

}

last\_button2\_press = now;

}

}

else if (GPIO\_Pin == CANON3\_Pin)

{

// fire

if (now - last\_button3\_press > 200)

{

fire\_cannon();

last\_button3\_press = now;

}

}

}

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

\_\_disable\_irq();

while (1)

{

/\* Optionally blink an LED here to indicate error \*/

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User may add reporting (e.g. log over UART) \*/

(void)file;

(void)line;

/\* USER CODE END 6 \*/

}

#endif /\* USE\_FULL\_ASSERT \*/

## Игра "Перехват корабля" на базе STM32F103RB и Arduino Multi-Function Shield V2

## 1. Аппаратная платформа

### 1.1 Плата расширения

**Arduino Multi-Function Shield V2**

Универсальная плата расширения, совместимая с платами Arduino Uno и STM32 Nucleo через разъёмы формата Arduino. Содержит следующие компоненты:

#### Основные элементы платы:

* **Четырёхразрядный семисегментный индикатор** с общим анодом
* **Три кнопки** (S1, S2, S3)
* **Три светодиода** (D10, D11, D12)
* **Пьезоэлектрический зуммер**
* **Потенциометр**
* **Фоторезистор и датчик температуры**

## 2. Распиновка и конфигурация периферии

### 2.1 Кнопки управления

Кнопки подключены к линиям GPIO с настроенными внешними прерываниями (EXTI) по фронту нарастания сигнала.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Компонент** | **Пин STM32** | **Порт** | **Прерывание** | **Назначение** |
| CANON1 (S1) | PA1 | GPIOA | EXTI1 | Перемещение пушки влево |
| CANON2 (S2) | PA4 | GPIOA | EXTI4 | Перемещение пушки вправо |
| CANON3 (S3) | PB0 | GPIOB | EXTI0 | Выстрел |

**Конфигурация:**

* Режим: GPIO\_MODE\_IT\_RISING (прерывание по нарастающему фронту)
* Подтяжка: GPIO\_PULLDOWN (к земле)
* Дребезг контактов: программная защита с задержкой 200 мс

### 2.2 Светодиоды индикации

Светодиоды используются для отображения счёта попаданий в двоичном формате (0-7).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Светодиод** | **Пин STM32** | **Порт** | **Разряд** | **Логика** |
| LED\_D10 | PB6 | GPIOB | Бит 0 | Инверсная |
| LED\_D11 | PA7 | GPIOA | Бит 1 | Инверсная |
| LED\_D12 | PA6 | GPIOA | Бит 2 | Инверсная |

**Инверсная логика светодиодов:**

* GPIO\_PIN\_SET (HIGH) = светодиод выключен
* GPIO\_PIN\_RESET (LOW) = светодиод включен

### 2.3 Семисегментный индикатор

Четырёхразрядный индикатор управляется через два последовательно включённых сдвиговых регистра 74HC595.

|  |  |  |  |
| --- | --- | --- | --- |
| **Сигнал** | **Пин STM32** | **Порт** | **Назначение** |
| CLK\_DISP | PA8 | GPIOA | Тактовый сигнал (SRCLK) |
| DATA\_DISP | PA9 | GPIOA | Последовательные данные (SER) |
| LATCH\_DISP | PB5 | GPIOB | Защёлка (RCLK) |

## 3. Принцип работы семисегментного индикатора

### 3.1 Архитектура управления

Индикатор использует **динамическую индикацию** с мультиплексированием. Четыре разряда отображаются последовательно с высокой частотой обновления, создавая иллюзию одновременного свечения.

### 3.2 Сдвиговые регистры 74HC595

Два микросхемы 74HC595 соединены каскадно:

**Регистр 1** (младший в цепи):

* Выходы Q0-Q7 управляют сегментами текущего разряда (a, b, c, d, e, f, g, DP)

**Регистр 2** (старший в цепи):

* Выходы Q0-Q3 управляют общими анодами четырёх разрядов
* Q4-Q7 не используются

### 3.3 Структура байта сегментов

Бит: 7 6 5 4 3 2 1 0

DP g f e d c b a

Схема разряда:

\_\_\_a\_\_\_

| |

f| |b

|\_\_\_g\_\_\_|

| |

e| |c

|\_\_\_d\_\_\_| .DP

## 4. Алгоритм игры

### 4.1 Игровые объекты

#### Корабль

* **Позиция:** разряд (0-3) и сегмент (a-g)
* **Направление:** UP, DOWN, LEFT, RIGHT
* **Движение:** по заранее определённой таблице переходов (граф состояний) между состояниями
* **Период движения:** 150 мс
* **Wrap-around:** возможен переход между крайними разрядами (3→0 и 0→3)

#### Пушка

* **Позиция:** разряд (0-3), отображается точкой (DP)
* **Управление:** кнопками CANON1 (влево) и CANON2 (вправо)
* **Ограничения:** без wrap-around (остаётся в пределах 0-3)

#### Снаряд

* **Анимация:** двухкадровая (600 мс)
  + Кадр 1 (300 мс): нижняя половина квадрата (сегменты g, c, d, e)
  + Кадр 2 (300 мс): верхняя половина квадрата (сегменты a, b, g, f)
* **Кулдаун:** 800 мс между выстрелами

### 4.2 Механика столкновений

Уничтожение корабля происходит при **посегментном пересечении**:

* Корабль должен находиться в том же разряде, что и снаряд
* Сегмент корабля должен совпадать с одним из активных сегментов текущего кадра снаряда

### 4.3 Система подсчёта очков

Счёт отображается на трёх светодиодах в двоичном формате (0-7):

Счёт: 0 → LED: 000 (все выключены)

Счёт: 1 → LED: 001 (D10)

Счёт: 2 → LED: 010 (D11)

Счёт: 3 → LED: 011 (D10, D11)

...

Счёт: 7 → LED: 111 (все включены)

Счёт: 8 → переполнение → 0

## 5. Программная реализация

### 5.1 Машина состояний корабля

Переходы описаны таблицей из 14 правил, определяющих возможные движения из каждого состояния (сегмент + направление).

a, RIGHT - (b, DOWN), (a, NEXT\_DIGIT\_RIGHT), (f, NEXT\_DIGIT\_DOWN)

a, LEFT - (f, DOWN), (a, PREV\_DIGIT\_LEFT), (b, PREV\_DIGIT\_DOWN)

b, UP - (a, LEFT), (a, NEXT\_DIGIT\_RIGHT)

b, DOWN - (g, LEFT), (c, DOWN), (g, NEXT\_DIGIT\_RIGHT), (e, NEXT\_DIGIT\_DOWN)

g, RIGHT - (b, UP), (c, DOWN), (f, NEXT\_DIGIT\_UP), (e, NEXT\_DIGIT\_DOWN), (g, NEXT\_DIGIT\_RIGHT)

g, LEFT - (f, UP), (e, DOWN), (g, PREV\_DIGIT\_LEFT), (c, PREV\_DIGIT\_DOWN), (b, PREV\_DIGIT\_UP)

c, UP - (g, LEFT), (b, UP), (g, NEXT\_DIGIT\_RIGHT), (f, NEXT\_DIGIT\_UP)

c, DOWN - (d, LEFT), (d, NEXT\_DIGIT\_RIGHT)

d, RIGHT - (c, UP), (d, NEXT\_DIGIT\_RIGHT), (e, NEXT\_DIGIT\_UP)

d, LEFT - (e, UP), (d, PREV\_DIGIT\_LEFT), (c, PREV\_DIGIT\_UP)

e, UP - (g, RIGHT), (f, UP), (g, PREV\_DIGIT\_LEFT), (b, PREV\_DIGIT\_UP)

e, DOWN - (d, RIGHT), (d, PREV\_DIGIT\_LEFT)

f, DOWN - (g, RIGHT), (e, DOWN), (g, PREV\_DIGIT\_LEFT), (c, PREV\_DIGIT\_DOWN)

f, UP - (a, RIGHT), (a, PREV\_DIGIT\_LEFT)