|  |  |
| --- | --- |
| ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»  Факультет компьютерных наук  Департамент программной инженерии | |
| **СОГЛАСОВАНО**  Профессор департамента программной инженерии факультета компьютерных наук, Профессор.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И. Р. Агамирзян  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. | **УТВЕРЖДАЮ**  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Шилов  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. |
| **ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС УПРАВЛЕНИЯ УМНЫМ ВЕЛОСИПЕДНЫМ ЗАМКОМ**  Текст программы  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.01.01-01 12 01-1-ЛУ** | |
|  | Исполнитель студент группы БПИ173  \_\_\_\_\_\_\_\_\_\_ / Дубина Д. О. /  «\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |
| --- | --- |
| Подп. И дата |  |
| Инв. № дубл. |  |
| Взам. Инв. № |  |
| Подп. И дата |  |
| Инв. № подл. |  |

Москва 2019

|  |  |
| --- | --- |
| **УТВЕРЖДЕНО**  **RU.17701729.01.01-01 12 01-1-ЛУ** |  |
|  |  |
| **ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС УПРАВЛЕНИЯ УМНЫМ ВЕЛОСИПЕДНЫМ ЗАМКОМ**  Текст программы  RU.17701729.01.01-01 12 01-1  **Листов 59** | |

|  |  |
| --- | --- |
| Подп. И дата |  |
| Инв. № дубл. |  |
| Взам. Инв. № |  |
| Подп. И дата |  |
| Инв. № подл. |  |

Москва 2019

Содержание

1. Основные термины и определения3

2. Текст программы комплекса4

2.1. Файл Images.h 4

2.2. Файл Defines.h 15

2.3. Файл OLED.h 16

2.4. Файл OLED.cpp 16

2.5. Файл TWI.h 17

2.6. Файл TWI.cpp 18

2.7. Файл main.cpp 18

3. Текст программы модели корпуса50

3.1. Файл bot.scad 50

3.2. Файл buttons.scad 52

3.3. Файл left.scad 53

3.4. Файл right.scad 54

3.5. Файл top.scad 54

3.6. Файл U.scad57

Лист регистрации изменений59

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Изм. | Лист | № докум. | Подп. | Дата |
|  |  |  |  |  |
| Инв. № подл. | Подп. И дата | Взам. Инв. № | Инв. № дубл. | Подп. и дата |
| RU.17701729.01.01-01 12 01-1 |  |  |  |  |

1. Основные термины и определения

**.scad** - формат файлов программы OpenScad для 3D моделирования.

**UART** - Универсальный асинхронный приёмопередатчик (англ. Universal Asynchronous Receiver-Transmitter, UART) — узел вычислительных устройств, предназначенный для организации связи с другими цифровыми устройствами. Преобразует передаваемые данные в последовательный вид так, чтобы было возможно передать их по одной физической цифровой линии другому аналогичному устройству. Метод преобразования хорошо стандартизован и широко применяется в компьютерной технике.

**I2C** - последовательная асимметричная шина для связи между интегральными схемами внутри электронных приборов. Использует две двунаправленные линии связи (SDA и SCL), применяется для соединения низкоскоростных периферийных компонентов с процессорами и микроконтроллерами (например, на материнских платах, во встраиваемых системах, в мобильных телефонах).

**Прерывание** - сигнал от программного или аппаратного обеспечения, сообщающий процессору о наступлении какого-либо события, требующего немедленного внимания. Прерывание извещает процессор о наступлении высокоприоритетного события, требующего прерывания текущего кода, выполняемого процессором. Процессор отвечает приостановкой своей текущей активности, сохраняя свое состояние и выполняя функцию, называемую обработчиком прерывания (или программой обработки прерывания), которая реагирует на событие и обслуживает его, после чего возвращает управление в прерванный код.

**Регистр** - последовательное или параллельное логическое устройство, используемое для хранения n-разрядных двоичных чисел и выполнения преобразований над ними.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Изм. | Лист | № докум. | Подп. | Дата |
|  |  |  |  |  |
| Инв. № подл. | Подп. И дата | Взам. Инв. № | Инв. № дубл. | Подп. и дата |
| RU.17701729.01.01-01 12 01-1 |  |  |  |  |

2. Текст программы комплекса

2.1. Файл Images.h

#ifndef IMAGES\_H\_

#define IMAGES\_H\_

#include "Defines.h"

*uint8\_t* zero[]={

0b00000000,

0b00111100,

0b00100100,

0b00100100,

0b00100100,

0b00100100,

0b00111100,

0b00000000

};

*uint8\_t* one[]={

0b00000000,

0b00011000,

0b00111000,

0b00001000,

0b00001000,

0b00001000,

0b00111100,

0b00000000

};

*uint8\_t* two[]={

0b00000000,

0b00111100,

0b00000100,

0b00000100,

0b00011000,

0b00110000,

0b00111100,

0b00000000

};

*uint8\_t* three[]={

0b00000000,

0b00111100,

0b00000100,

0b00111100,

0b00000100,

0b00000100,

0b00111100,

0b00000000

};

*uint8\_t* four[]={

0b00000000,

0b00100100,

0b00100100,

0b00100100,

0b00111100,

0b00000100,

0b00000100,

0b00000000

};

*uint8\_t* five[]={

0b00000000,

0b00111100,

0b00100000,

0b00111100,

0b00000100,

0b00000100,

0b00111100,

0b00000000

};

*uint8\_t* six[]={

0b00000000,

0b00111100,

0b00100000,

0b00111100,

0b00100100,

0b00100100,

0b00111100,

0b00000000

};

*uint8\_t* seven[]={

0b00000000,

0b00111100,

0b00000100,

0b00001000,

0b00010000,

0b00100000,

0b00100000,

0b00000000

};

*uint8\_t* eight[]={

0b00000000,

0b00111100,

0b00100100,

0b00111100,

0b00100100,

0b00100100,

0b00111100,

0b00000000

};

*uint8\_t* nine[]={

0b00000000,

0b00111100,

0b00100100,

0b00100100,

0b00111100,

0b00000100,

0b00111100,

0b00000000

};

*uint8\_t* none[]={

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000

};

*uint8\_t* line[]={

0b00000000,

0b00000000,

0b01111110,

0b00000000,

0b00000000,

0b00000000,

0b00000000,

0b00000000

};

*uint8\_t* arrow[]={

0b00000000,

0b00100000,

0b01111110,

0b01111110,

0b00100110,

0b00000110,

0b00000000,

0b00000000

};

*uint8\_t* upp[]={

0b00011000,

0b00111100,

0b01111110,

0b11111111,

0b00000000,

0b00000000,

0b00000000,

0b00000000

};

*uint8\_t* downn[]={

0b00000000,

0b11111111,

0b01111110,

0b00111100,

0b00011000,

0b00000000,

0b00000000,

0b00000000

};

*uint8\_t* gear[] ={

0B00000000, 0B00000001,0B10000000, 0B00000000,

0B00000000, 0B00000011,0B11000000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00011110, 0B00001111,0B11110000, 0B01111000,

0B00011111, 0B10011110,0B01111001, 0B11111000,

0B00011111, 0B11111100,0B00111111, 0B11111000,

0B00011111, 0B11100000,0B00000111, 0B11111000,

0B00001111, 0B00000000,0B00000000, 0B11111000,

0B00001110, 0B00000111,0B11100000, 0B01110000,

0B00000110, 0B00001110,0B01110000, 0B01100000,

0B00000110, 0B00011000,0B00011000, 0B01100000,

0B00001100, 0B00110000,0B00001100, 0B00110000,

0B00011100, 0B01100001,0B10000110, 0B00111000,

0B00111100, 0B11000011,0B11000011, 0B00111100,

0B01111001, 0B10000111,0B11100001, 0B10011110,

0B11110001, 0B10001110,0B01110001, 0B10011111,

0B11110001, 0B10001110,0B01110001, 0B10011111,

0B01111001, 0B10000111,0B11100001, 0B10011110,

0B00111100, 0B11000011,0B11000011, 0B00111100,

0B00011100, 0B01100001,0B10000110, 0B00111000,

0B00001100, 0B00110000,0B00001100, 0B00110000,

0B00000110, 0B00011000,0B00011000, 0B01100000,

0B00000110, 0B00001110,0B01110000, 0B01100000,

0B00001110, 0B00000111,0B11100000, 0B01110000,

0B00001111, 0B00000000,0B00000000, 0B11111000,

0B00011111, 0B11100000,0B00000111, 0B11111000,

0B00011111, 0B11111100,0B00111111, 0B11111000,

0B00011111, 0B10011110,0B01111001, 0B11111000,

0B00011110, 0B00001111,0B11110000, 0B01111000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000011,0B11000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000001,0B10000000, 0B00000000

};

*uint8\_t* info[] ={

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00001111,0B11110000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B11111111,0B11111111, 0B00000000,

0B00000000, 0B11111111,0B11111111, 0B00000000,

0B00000000, 0B11111111,0B11111111, 0B00000000,

0B00000000, 0B11111111,0B11111111, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000

};

*uint8\_t* noneh[] ={

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000

};

*uint8\_t* lock[] =

{

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000001,0B10000000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00011100,0B00111000, 0B00000000,

0B00000000, 0B01110000,0B00001110, 0B00000000,

0B00000000, 0B11100000,0B00000111, 0B00000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111100,0B00111111, 0B11000000,

0B00000011, 0B11111100,0B00111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111100,0B01111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000

};

*uint8\_t* rt[] =

{

0B10000000,

0B11000000,

0B11100000,

0B11110000,

0B11111000,

0B11111100,

0B11111110,

0B11111111,

0B11111111,

0B11111110,

0B11111100,

0B11111000,

0B11110000,

0B11100000,

0B11000000,

0B10000000

};

*uint8\_t* unlock[] =

{

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000001,0B10000000, 0B00000000,

0B00000000, 0B00000111,0B11100000, 0B00000000,

0B00000000, 0B00011100,0B00111000, 0B00000000,

0B00000000, 0B01110000,0B00001110, 0B00000000,

0B00000000, 0B11100000,0B00000111, 0B00000000,

0B00000001, 0B11000000,0B00000011, 0B010000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000001, 0B11000000,0B00000011, 0B10000000,

0B00000000, 0B10000000,0B00000011, 0B10000000,

0B00000000, 0B00000000,0B00000011, 0B10000000,

0B00000000, 0B00000000,0B00000011, 0B10000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111100,0B00111111, 0B11000000,

0B00000011, 0B11111100,0B00111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111100,0B01111111, 0B11000000,

0B00000011, 0B11111110,0B01111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000011, 0B11111111,0B11111111, 0B11000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000

};

*uint8\_t* lt[] =

{

0B00000001,

0B00000011,

0B00000111,

0B00001111,

0B00011111,

0B00111111,

0B01111111,

0B11111111,

0B11111111,

0B01111111,

0B00111111,

0B00011111,

0B00001111,

0B00000111,

0B00000011,

0B00000001

};

*uint8\_t* upwd[] =

{

0B00000001,0B10000000,

0B00000011,0B11000000,

0B00000111,0B11100000,

0B00001111,0B11110000,

0B00011111,0B11111000,

0B00111111,0B11111100,

0B01111111,0B11111110,

0B11111111,0B11111111

};

*uint8\_t* downwd[] =

{

0B11111111,0B11111111,

0B01111111,0B11111110,

0B00111111,0B11111100,

0B00011111,0B11111000,

0B00001111,0B11110000,

0B00000111,0B11100000,

0B00000011,0B11000000,

0B00000001,0B10000000

};

*uint8\_t* batary5[] =

{

0B00111111, 0B11111111,0B11111111, 0B11111100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B01111011,0B11011110, 0B11110100,

0B00101111, 0B01111011,0B11011110, 0B11110100,

0B00101111, 0B01111011,0B11011110, 0B11110100,

0B00101111, 0B01111011,0B11011110, 0B11110100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00111111, 0B11111111,0B11111111, 0B11111100

};

*uint8\_t* batary4[] =

{

0B00111111, 0B11111111,0B11111111, 0B11111100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B01111011,0B11011110, 0B00000100,

0B00101111, 0B01111011,0B11011110, 0B00000100,

0B00101111, 0B01111011,0B11011110, 0B00000100,

0B00101111, 0B01111011,0B11011110, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00111111, 0B11111111,0B11111111, 0B11111100

};

*uint8\_t* batary3[] =

{

0B00111111, 0B11111111,0B11111111, 0B11111100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B01111011,0B11000000, 0B00000100,

0B00101111, 0B01111011,0B11000000, 0B00000100,

0B00101111, 0B01111011,0B11000000, 0B00000100,

0B00101111, 0B01111011,0B11000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00111111, 0B11111111,0B11111111, 0B11111100

};

*uint8\_t* batary2[] =

{

0B00111111, 0B11111111,0B11111111, 0B11111100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B01111000,0B00000000, 0B00000100,

0B00101111, 0B01111000,0B00000000, 0B00000100,

0B00101111, 0B01111000,0B00000000, 0B00000100,

0B00101111, 0B01111000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00111111, 0B11111111,0B11111111, 0B11111100

};

*uint8\_t* batary1[] =

{

0B00111111, 0B11111111,0B11111111, 0B11111100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B00000000,0B00000000, 0B00000100,

0B00101111, 0B00000000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00111111, 0B11111111,0B11111111, 0B11111100

};

*uint8\_t* batary0[] =

{

0B00111111, 0B11111111,0B11111111, 0B11111100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00100000, 0B00000000,0B00000000, 0B00000100,

0B00111111, 0B11111111,0B11111111, 0B11111100

};

*uint8\_t* Net4[] =

{

0B11100000, 0B00000000,

0B11100000, 0B00000000,

0B11101110, 0B00000000,

0B11101110, 0B00000000,

0B11101110, 0B11100000,

0B11101110, 0B11100000,

0B11101110, 0B11101110,

0B11101110, 0B11101110,

};

*uint8\_t* Net3[] =

{

0B11100000, 0B00000000,

0B10100000, 0B00000000,

0B10101110, 0B00000000,

0B10101110, 0B00000000,

0B10101110, 0B11100000,

0B10101110, 0B11100000,

0B10101110, 0B11101110,

0B11101110, 0B11101110,

};

*uint8\_t* Net2[] =

{

0B11100000, 0B00000000,

0B10100000, 0B00000000,

0B10101110, 0B00000000,

0B10101010, 0B00000000,

0B10101010, 0B11100000,

0B10101010, 0B11100000,

0B10101010, 0B11101110,

0B11101110, 0B11101110,

};

*uint8\_t* Net1[] =

{

0B11100000, 0B00000000,

0B10100000, 0B00000000,

0B10101110, 0B00000000,

0B10101010, 0B00000000,

0B10101010, 0B11100000,

0B10101010, 0B10100000,

0B10101010, 0B10101110,

0B11101110, 0B11101110,

};

*uint8\_t* Net0[] =

{

0B11100000, 0B00000000,

0B10100000, 0B00001010,

0B10101110, 0B00000100,

0B10101010, 0B00001010,

0B10101010, 0B11100000,

0B10101010, 0B10100000,

0B10101010, 0B10101110,

0B11101110, 0B10101110,

};

*uint8\_t* A[] =

{

0B00000000,

0B00011000,

0B00111100,

0B01100110,

0B01111110,

0B01100110,

0B01100110,

0B00000000

};

*uint8\_t* B[] =

{

0B00000000,

0B01111100,

0B01000010,

0B01111100,

0B01000010,

0B01000010,

0B01111100,

0B00000000

};

*uint8\_t* C[] =

{

0B00000000,

0B00111110,

0B01100000,

0B01000000,

0B01000000,

0B01100000,

0B00111110,

0B00000000

};

*uint8\_t* D[] =

{

0B00000000,

0B01111000,

0B01000100,

0B01000010,

0B01000010,

0B01000100,

0B01111000,

0B00000000

};

*uint8\_t* E[] =

{

0B00000000,

0B01111110,

0B01000000,

0B01111110,

0B01000000,

0B01000000,

0B01111110,

0B00000000

};

*uint8\_t* F[] =

{

0B00000000,

0B01111110,

0B01000000,

0B01111110,

0B01000000,

0B01000000,

0B01000000,

0B00000000

};

*uint8\_t* G[] =

{

0B00000000,

0B00111110,

0B01100000,

0B01000000,

0B01000000,

0B01100110,

0B00111110,

0B00000000

};

*uint8\_t* H[] =

{

0B00000000,

0B01000010,

0B01000010,

0B01111110,

0B01000010,

0B01000010,

0B01000010,

0B00000000

};

*uint8\_t* I[] =

{

0B00000000,

0B00111100,

0B00011000,

0B00011000,

0B00011000,

0B00011000,

0B00111100,

0B00000000

};

*uint8\_t* J[] =

{

0B00000000,

0B00000110,

0B00000110,

0B00000110,

0B00000110,

0B01100110,

0B00111100,

0B00000000

};

*uint8\_t* K[] =

{

0B00000000,

0B01000100,

0B01001000,

0B01110000,

0B01110000,

0B01001000,

0B01000100,

0B00000000

};

*uint8\_t* L[] =

{

0B00000000,

0B01000000,

0B01000000,

0B01000000,

0B01000000,

0B01000000,

0B01111110,

0B00000000

};

*uint8\_t* M[] =

{

0B00000000,

0B01000010,

0B01100110,

0B01011010,

0B01000010,

0B01000010,

0B01000010,

0B00000000

};

*uint8\_t* N[] =

{

0B00000000,

0B01000010,

0B01100010,

0B01010010,

0B01001010,

0B01000110,

0B01000010,

0B00000000

};

*uint8\_t* O[] =

{

0B00000000,

0B00111100,

0B01100110,

0B01000010,

0B01000010,

0B01100110,

0B00111100,

0B00000000

};

*uint8\_t* P[] =

{

0B00000000,

0B01111100,

0B01000010,

0B01000010,

0B01111100,

0B01000000,

0B01000000,

0B00000000

};

*uint8\_t* Q[] =

{

0B00000000,

0B00111100,

0B01000010,

0B01000010,

0B01000010,

0B01000010,

0B00111110,

0B00000001

};

*uint8\_t* R[] =

{

0B00000000,

0B01111100,

0B01000010,

0B01000010,

0B01111100,

0B01000100,

0B01000010,

0B00000000

};

*uint8\_t* S[] =

{

0B00000000,

0B00111110,

0B01000000,

0B00111100,

0B00000010,

0B00000010,

0B01111100,

0B00000000

};

*uint8\_t* T[] =

{

0B00000000,

0B01111110,

0B00011000,

0B00011000,

0B00011000,

0B00011000,

0B00011000,

0B00000000

};

*uint8\_t* U[] =

{

0B00000000,

0B01000010,

0B01000010,

0B01000010,

0B01000010,

0B01000010,

0B00111100,

0B00000000

};

*uint8\_t* V[] =

{

0B00000000,

0B01000010,

0B01000010,

0B01000010,

0B00100100,

0B00100100,

0B00011000,

0B00000000

};

*uint8\_t* W[] =

{

0B00000000,

0B01000010,

0B01000010,

0B01000010,

0B01011010,

0B01011010,

0B00100100,

0B00000000

};

*uint8\_t* X[] =

{

0B00000000,

0B01000010,

0B00100100,

0B00011000,

0B00011000,

0B00100100,

0B01000010,

0B00000000

};

*uint8\_t* Y[] =

{

0B00000000,

0B01000010,

0B00100100,

0B00011000,

0B00011000,

0B00011000,

0B00011000,

0B00000000

};

*uint8\_t* Z[] =

{

0B00000000,

0B01111110,

0B00000100,

0B00001000,

0B00010000,

0B00100000,

0B01111110,

0B00000000

};

*uint8\_t* what[] =

{

0B00000000,

0B01111110,

0B01000100,

0B00011000,

0B00011000,

0B00000000,

0B00011000,

0B00000000

};

*uint8\_t* NL[] =

{

0B00000000,

0B01111100,

0B00001100,

0B00001100,

0B00001100,

0B00011110,

0B00001100,

0B00000000

};

*uint8\_t* dot[] =

{

0B00000000,

0B00000000,

0B00000000,

0B00000000,

0B00000000,

0B00110000,

0B00110000,

0B00000000

};

*uint8\_t* SL[] =

{

0B00000000, 0B00000000, 0B00000000, 0B00000000,

0B00000011, 0B11000000, 0B00001111, 0B00000000,

0B00001111, 0B11110000, 0B00001111, 0B00000000,

0B00111100, 0B00111100, 0B00001111, 0B00000000,

0B00111100, 0B00001100, 0B00011110, 0B00000000,

0B00011110, 0B00000000, 0B00011110, 0B00000000,

0B00001111, 0B10000000, 0B00011110, 0B00000000,

0B00000111, 0B00000000, 0B00111100, 0B00000000,

0B00000011, 0B10000000, 0B00111100, 0B00000000,

0B00000001, 0B11100000, 0B00111100, 0B00000000,

0B00000000, 0B01111000, 0B01111000, 0B00000000,

0B00110000, 0B00111100, 0B01111000, 0B00000000,

0B00111100, 0B00111100, 0B01111000, 0B00000000,

0B00001111, 0b11110000, 0B01111111, 0B11110000,

0B00000011, 0B11000000, 0B01111111, 0B11110000,

0B00000000, 0B00000000, 0B00000000, 0B00000000,

};

*uint8\_t* backk[] =

{

0B00000001, 0B00000000,

0B00000011, 0B11100000,

0B00000011, 0B11110000,

0B00000001, 0B00011000,

0B00011000, 0B00011000,

0B00001110, 0B01110000,

0B00000011, 0B11000000,

0B00000000, 0B00000000

};

*uint8\_t* full[] =

{

0B00000000,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B00000000

};

*uint8\_t* dfull[] =

{

0B00000000,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B01111110,

0B00000000

};

*uint8\_t* clear\_row[] =

{

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,

0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000,0B00000000, 0B00000000

};

#endif /\* IMAGES\_H\_ \*/

**2.2. Файл Defines.h**

#ifndef DEFINES\_H\_

#define DEFINES\_H\_

#include<avr/io.h>

#include<avr/interrupt.h>

#include <util/delay.h>

#include <avr/wdt.h>

#include <avr/sleep.h>

#define OLED\_WIDTH 128

#define OLED\_HEIGHT 64

#define CPU\_F 20000000UL

#define SCL\_F 1250000UL

#define Skip\_Flag 0xff

#define Dev\_Adr 0b00111100

#endif /\* DEFINES\_H\_ \*/

**2.3. Файл OLED.h**

class OLED

{

private:

*uint8\_t* oled\_bufer[(OLED\_HEIGHT/8)\*OLED\_WIDTH];

TWI wire;

public:

OLED(TWI \_wire,*uint8\_t* light);

void OLED\_Command(int command);

void OLED\_Data(int data);

void OLED\_Write\_Bufer();

void OLED\_Bufer\_Clear();

void OLED\_Clear\_Bufer\_part(int x,int y,int width,int height);

void OLED\_Write\_To\_Bufer(int x,int y,int width,int height,const *uint8\_t*\* img);

};

#endif /\* OLED\_H\_ \*/

**2.4. Файл OLED.cpp**

#include "OLED.h"

OLED::OLED(TWI \_wire,*uint8\_t* light)

{

wire=\_wire;

wire.twi\_Init();

wire.twi\_Start();

wire.twi\_SendAdress();

OLED\_Command(0xA8);

OLED\_Command(0x3F);

OLED\_Command(0xD3);

OLED\_Command(0x00);

OLED\_Command(0x40);

OLED\_Command(0xA1);

OLED\_Command(0xC8);

OLED\_Command(0xDA);

OLED\_Command(0x12);

OLED\_Command(0x81);

OLED\_Command(light);

OLED\_Command(0xA4);

OLED\_Command(0xA6);

OLED\_Command(0xD5);

OLED\_Command(0x80);

OLED\_Command(0x8D);

OLED\_Command(0x14);

OLED\_Command(0xAF);

OLED\_Command(0x20);

OLED\_Command(0x00);

OLED\_Command(0x21);

OLED\_Command(0);

OLED\_Command(127);

OLED\_Command(0x22);

OLED\_Command(0);

OLED\_Command(7);

TWDR=0x40;

TWCR=(1<<TWINT)|(1<<TWEN);

OLED\_Bufer\_Clear();

OLED\_Data(0b00000000);

OLED\_Write\_Bufer();

}

void OLED::OLED\_Command(int command)

{

TWDR=0x80;

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR & (1<<TWINT))){};

TWDR=command;

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR & (1<<TWINT))){};

}

void OLED::OLED\_Data(int data)

{

TWDR=data;

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR & (1<<TWINT))){};

}

void OLED::OLED\_Write\_Bufer()

{

for(int i=0;i<(OLED\_HEIGHT/8)\*OLED\_WIDTH;i++)

{

OLED\_Data(oled\_bufer[i]);

}

}

void OLED::OLED\_Bufer\_Clear()

{

for(int i=0;i<(OLED\_HEIGHT/8)\*OLED\_WIDTH;i++)

{

oled\_bufer[i]=0b00000000;

}

}

void OLED::OLED\_Clear\_Bufer\_part(int x,int y,int width,int height)

{

for(int j=0;j<height;j++)

{

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

{

for(int k=0;k<8;k++)

{

oled\_bufer[(x+i\*8+k)+(y+j/8)\*OLED\_WIDTH]=0b00000000;

}

}

}

}

void OLED::OLED\_Write\_To\_Bufer(int x,int y,int width,int height,const *uint8\_t*\* img)

{

for(int j=0;j<height;j++)

{

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

{

for(int k=0;k<8;k++)

{

oled\_bufer[(x+i\*8+k)+(y+j/8)\*OLED\_WIDTH]|=((img[j\*width+i]<<k)&0b10000000)>>(7-j%8);

}

}

}

}

**2.5. Файл TWI.h**

#ifndef TWI\_H\_

#define TWI\_H\_

#include "Defines.h"

class TWI

{

public:

TWI();

void twi\_Init(void);

void twi\_SendAdress(void);

void twi\_SendByte(int Inf);

void twi\_Start(void);

void twi\_Stop(void);

};

#endif

**2.6. Файл TWI.cpp**

#include "TWI.h"

TWI::TWI(){}

void TWI::twi\_Init(void)

{

TWBR=(((CPU\_F)/(SCL\_F)-16)/2);

TWSR=0;

}

void TWI::twi\_SendAdress(void)

{

TWDR=(Dev\_Adr<<1)|0;

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR & (1<<TWINT))){};

}

void TWI::twi\_Start(void)

{

TWCR=(1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while(!(TWCR & (1<<TWINT))){};

}

void TWI::twi\_Stop(void)

{

TWCR=(1<<TWINT)|(1<<TWSTO)|(1<<TWEN);

}

void TWI::twi\_SendByte(int Inf)

{

TWDR=Inf;

TWCR=(1<<TWINT)|(1<<TWEN);

while(!(TWCR & (1<<TWINT))){};

}

**2.7. Файл main.cpp**

#include "Defines.h"

#include "Images.h"

#include "TWI.h"

#include "OLED.h"

#include <avr/eeprom.h>

int brigtnesslvl=1;

TWI wire;

OLED oled(wire,brigtnesslvl\*25);

bool Main\_Menu\_Status=true;

bool Password\_Menu\_Status=false;

bool Settings\_Menu\_Status=false;

bool Info\_Menu\_Status=false;

bool User\_Info\_Status=false;

bool Device\_Info\_Status=false;

bool User\_Add\_Status=false;

bool Reset\_Status=false;

bool Brightness\_Status=false;

bool Waiting\_Status=false;

bool locktimer=false;

bool locked=false;

bool dream=false;

bool dreamreset=false;

bool newdevice=false;

char\* owner\_number="89260755725";

*uint8\_t* password[5]={0,0,0,0,0};

*uint8\_t* passwin[5]={0,0,0,0,0};

class Menu\_Element

{

protected:

int pointer;

public:

virtual void Default();

virtual void refresh();

virtual void close();

virtual void next();

virtual void previous();

virtual void choise();

virtual void back();

virtual void animate();

virtual void actions();

};

#define MAIN\_MENU\_POINER\_COUNT 2

#define MAIN\_MENU\_IMG\_X 50

#define MAIN\_MENU\_IMG\_Y 2

class Main\_Menu

{

protected:

int pointer=0;

public:

Main\_Menu()

{

};

void Default()

{

oled.OLED\_Write\_To\_Bufer(MAIN\_MENU\_IMG\_X-28,MAIN\_MENU\_IMG\_Y+1,1,16,lt);

oled.OLED\_Write\_To\_Bufer(MAIN\_MENU\_IMG\_X+52,MAIN\_MENU\_IMG\_Y+1,1,16,rt);

};

void refresh()

{

//oled.OLED\_Bufer\_Clear();

Default();

actions();

};

void close()

{

oled.OLED\_Bufer\_Clear();

};

void next()

{

pointer++;

if(pointer>MAIN\_MENU\_POINER\_COUNT) pointer=0;

actions();

};

void previous()

{

pointer--;

if(pointer<0) pointer=MAIN\_MENU\_POINER\_COUNT;

actions();

};

void choise()

{

switch(pointer)

{

case 0:

{

Main\_Menu\_Status=false;

Password\_Menu\_Status=true;

close();

break;

}

case 1:

{

Main\_Menu\_Status=false;

Settings\_Menu\_Status=true;

close();

break;

}

case 2:

{

Main\_Menu\_Status=false;

Info\_Menu\_Status=true;

close();

break;

}

}

};

void back()

{

};

void animate()

{

};

void actions()

{

switch(pointer)

{

case 0:

{

oled.OLED\_Clear\_Bufer\_part(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32);

if(!locked) oled.OLED\_Write\_To\_Bufer(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32,unlock);

else oled.OLED\_Write\_To\_Bufer(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32,lock);

break;

}

case 1:

{

oled.OLED\_Clear\_Bufer\_part(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32);

oled.OLED\_Write\_To\_Bufer(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32,gear);

break;

}

case 2:

{

oled.OLED\_Clear\_Bufer\_part(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32);

oled.OLED\_Write\_To\_Bufer(MAIN\_MENU\_IMG\_X,MAIN\_MENU\_IMG\_Y,4,32,info);

break;

}

}

};

};

#define PASSWORD\_MENU\_POINER\_COUNT 4

#define PASSWORD\_MENU\_SIMVOLS\_COUNT 10

#define PASSWORD\_MENU\_IMG\_X\_0 12

#define PASSWORD\_MENU\_IMG\_X\_1 36

#define PASSWORD\_MENU\_IMG\_X\_2 60

#define PASSWORD\_MENU\_IMG\_X\_3 84

#define PASSWORD\_MENU\_IMG\_X\_4 108

#define PASSWORD\_MENU\_IMG\_Y 3

class Password\_Menu

{

protected:

int pointer=0;

int symbol\_pointer=0;

int x[5]={PASSWORD\_MENU\_IMG\_X\_0,PASSWORD\_MENU\_IMG\_X\_1,PASSWORD\_MENU\_IMG\_X\_2,PASSWORD\_MENU\_IMG\_X\_3,PASSWORD\_MENU\_IMG\_X\_4};

int y=PASSWORD\_MENU\_IMG\_Y;

public:

Password\_Menu()

{

};

void Default()

{

oled.OLED\_Write\_To\_Bufer(x[0],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[1],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[2],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[3],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[4],y+1,1,8,line);

};

void refresh()

{

//oled.OLED\_Bufer\_Clear();

Default();

actions();

};

void close()

{

pointer=0;

symbol\_pointer=0;

Password\_Menu\_Status=false;

Main\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

symbol\_pointer++;

if(symbol\_pointer>PASSWORD\_MENU\_SIMVOLS\_COUNT) symbol\_pointer=0;

actions();

};

void previous()

{

symbol\_pointer--;

if(symbol\_pointer<0) symbol\_pointer=PASSWORD\_MENU\_SIMVOLS\_COUNT;

actions();

};

void choise()

{

switch(symbol\_pointer)

{

case 0:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 1:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 2:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 3:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 4:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 5:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 6:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 7:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 8:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 9:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

passwin[pointer-1]=symbol\_pointer;

}

break;

}

case 10:

{

back();

break;

}

}

if(pointer>PASSWORD\_MENU\_POINER\_COUNT) {

if(!locked)

{

password[0]=passwin[0];

password[1]=passwin[1];

password[2]=passwin[2];

password[3]=passwin[3];

password[4]=passwin[4];

passwin[0]=0;

passwin[1]=0;

passwin[2]=0;

passwin[3]=0;

passwin[4]=0;

locked=!locked;

}

else

{

if((password[0]==passwin[0])&(password[1]==passwin[1])&(password[2]==passwin[2])&(password[3]==passwin[3])&(password[4]==passwin[4]))

locked=!locked;

}

Main\_Menu\_Status=true;

Password\_Menu\_Status=false;

close();

}

else

{

symbol\_pointer=0;

actions();

}

};

void back()

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Clear\_Bufer\_part(x[pointer],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[pointer],y-1,1,8);

pointer--;

if(pointer<0)

{

Password\_Menu\_Status=false;

Main\_Menu\_Status=true;

close();

pointer=0;

}

else actions();

};

void animate()

{

};

void actions()

{

if(Password\_Menu\_Status){

switch(symbol\_pointer)

{

case 0:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,zero);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 1:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,one);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 2:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,two);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 3:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,three);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 4:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,four);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 5:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,five);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 6:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,six);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 7:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,seven);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 8:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,eight);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 9:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,nine);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 10:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,arrow);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

}

}

};

};

#define SETTINGS\_MENU\_POINER\_COUNT 3

#define SETTINGS\_MENU\_IMG\_X\_0 8

#define SETTINGS\_MENU\_IMG\_X\_1 16

#define SETTINGS\_MENU\_IMG\_X\_2 28

#define SETTINGS\_MENU\_IMG\_X\_3 56

#define SETTINGS\_MENU\_IMG\_Y 4

class Settings\_Menu

{

protected:

int pointer=0;

int x[4]={SETTINGS\_MENU\_IMG\_X\_0,SETTINGS\_MENU\_IMG\_X\_1,SETTINGS\_MENU\_IMG\_X\_2,SETTINGS\_MENU\_IMG\_X\_3};

int y=SETTINGS\_MENU\_IMG\_Y;

public:

Settings\_Menu()

{

};

void Default()

{

oled.OLED\_Write\_To\_Bufer(56,y+2,2,8,downwd);

oled.OLED\_Write\_To\_Bufer(56,y-2,2,8,upwd);

};

void refresh()

{

//oled.OLED\_Bufer\_Clear();

Default();

actions();

};

void close()

{

pointer=0;

Settings\_Menu\_Status=false;

Main\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

pointer++;

if(pointer>SETTINGS\_MENU\_POINER\_COUNT) pointer=0;

actions();

};

void previous()

{

pointer--;

if(pointer<0) pointer=SETTINGS\_MENU\_POINER\_COUNT;

actions();

};

void choise()

{

switch(pointer)

{

case 0:

{

Settings\_Menu\_Status=false;

Brightness\_Status=true;

//close();

oled.OLED\_Bufer\_Clear();

break;

}

case 1:

{

Settings\_Menu\_Status=false;

Reset\_Status=true;

//close();

oled.OLED\_Bufer\_Clear();

break;

}

case 2:

{

Settings\_Menu\_Status=false;

User\_Add\_Status=true;

oled.OLED\_Bufer\_Clear();

break;

}

case 3:

{

back();

break;

}

}

};

void back()

{

close();

pointer=0;

};

void animate()

{

};

void actions()

{

if(Settings\_Menu\_Status){

switch(pointer)

{

case 0:

{

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer]+0,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[pointer]+8,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+16,y,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[pointer]+32,y,1,8,B);

oled.OLED\_Write\_To\_Bufer(x[pointer]+40,y,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[pointer]+48,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[pointer]+56,y,1,8,G);

oled.OLED\_Write\_To\_Bufer(x[pointer]+64,y,1,8,H);

oled.OLED\_Write\_To\_Bufer(x[pointer]+72,y,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[pointer]+80,y,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[pointer]+88,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+96,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[pointer]+104,y,1,8,S);

break;

}

case 1:

{

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer]+0,y,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[pointer]+8,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+16,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[pointer]+24,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+32,y,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[pointer]+48,y,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[pointer]+56,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+64,y,1,8,V);

oled.OLED\_Write\_To\_Bufer(x[pointer]+72,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[pointer]+80,y,1,8,C);

oled.OLED\_Write\_To\_Bufer(x[pointer]+88,y,1,8,E);

break;

}

case 2:

{

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer]+0,y,1,8,A);

oled.OLED\_Write\_To\_Bufer(x[pointer]+8,y,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[pointer]+16,y,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[pointer]+32,y,1,8,O);

oled.OLED\_Write\_To\_Bufer(x[pointer]+40,y,1,8,W);

oled.OLED\_Write\_To\_Bufer(x[pointer]+48,y,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[pointer]+56,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+64,y,1,8,R);

break;

}

case 3:

{

//oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,2,8,backk);

break;

}

}

}

};

};

#define INFO\_MENU\_POINER\_COUNT 2

#define INFO\_MENU\_IMG\_X\_0 24

#define INFO\_MENU\_IMG\_X\_1 32

#define INFO\_MENU\_IMG\_X\_2 54

#define INFO\_MENU\_IMG\_Y 4

class Info\_menu

{

protected:

int pointer=0;

int x[3]={ INFO\_MENU\_IMG\_X\_0, INFO\_MENU\_IMG\_X\_1, INFO\_MENU\_IMG\_X\_2};

int y= INFO\_MENU\_IMG\_Y;

public:

Info\_menu()

{

};

void Default()

{

oled.OLED\_Write\_To\_Bufer(56,y+2,2,8,downwd);

oled.OLED\_Write\_To\_Bufer(56,y-2,2,8,upwd);

};

void refresh()

{

Default();

actions();

};

void close()

{

pointer=0;

Info\_Menu\_Status=false;

Main\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

pointer++;

if(pointer>INFO\_MENU\_POINER\_COUNT) pointer=0;

actions();

};

void previous()

{

pointer--;

if(pointer<0) pointer=INFO\_MENU\_POINER\_COUNT;

actions();

};

void choise()

{

switch(pointer)

{

case 0:

{

Info\_Menu\_Status=false;

Device\_Info\_Status=true;

//close();

oled.OLED\_Bufer\_Clear();

break;

}

case 1:

{

Info\_Menu\_Status=false;

User\_Info\_Status=true;

//close();

oled.OLED\_Bufer\_Clear();

break;

}

case 2:

{

back();

break;

}

}

};

void back()

{

close();

pointer=0;

};

void animate()

{

};

void actions()

{

if(Info\_Menu\_Status){

switch(pointer)

{

case 0:

{

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer]+0,y,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[pointer]+8,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+16,y,1,8,V);

oled.OLED\_Write\_To\_Bufer(x[pointer]+24,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[pointer]+32,y,1,8,C);

oled.OLED\_Write\_To\_Bufer(x[pointer]+40,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+56,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[pointer]+64,y,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[pointer]+72,y,1,8,F);

oled.OLED\_Write\_To\_Bufer(x[pointer]+80,y,1,8,O);

break;

}

case 1:

{

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer]+0,y,1,8,U);

oled.OLED\_Write\_To\_Bufer(x[pointer]+8,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[pointer]+16,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[pointer]+24,y,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[pointer]+40,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[pointer]+48,y,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[pointer]+56,y,1,8,F);

oled.OLED\_Write\_To\_Bufer(x[pointer]+64,y,1,8,O);

break;

}

case 2:

{

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,2,8,backk);

break;

}

}

}

};

};

#define US\_INFO\_POINER\_COUNT 4

#define US\_INFO\_IMG\_X\_0 8

#define US\_INFO\_IMG\_X\_1 44

#define US\_INFO\_IMG\_X\_2 8

#define US\_INFO\_IMG\_X\_3 20

#define US\_INFO\_IMG\_Y 2

class User\_Info

{

protected:

int pointer=0;

int x[4]={ US\_INFO\_IMG\_X\_0, US\_INFO\_IMG\_X\_1, US\_INFO\_IMG\_X\_2,US\_INFO\_IMG\_X\_3};

int y= US\_INFO\_IMG\_Y;

public:

User\_Info()

{

};

void Default()

{

};

void refresh()

{

Default();

actions();

};

void close()

{

pointer=0;

User\_Info\_Status=false;

Info\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

close();

actions();

};

void previous()

{

close();

actions();

};

void choise()

{

close();

};

void back()

{

close();

pointer=0;

};

void animate()

{

};

void actions()

{

if(User\_Info\_Status){

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[0]+0,y,1,8,U);

oled.OLED\_Write\_To\_Bufer(x[0]+8,y,1,8,J);

oled.OLED\_Write\_To\_Bufer(x[0]+16,y,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[0]+24,y,1,8,O);

oled.OLED\_Write\_To\_Bufer(x[0]+32,y,1,8,B);

oled.OLED\_Write\_To\_Bufer(x[0]+40,y,1,8,U);

oled.OLED\_Write\_To\_Bufer(x[0]+48,y,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[0]+56,y,1,8,O);

oled.OLED\_Write\_To\_Bufer(x[0]+64,y,1,8,V);

oled.OLED\_Write\_To\_Bufer(x[0]+72,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[0]+80,y,1,8,K);

oled.OLED\_Write\_To\_Bufer(x[0]+88,y,1,8,A);

oled.OLED\_Write\_To\_Bufer(x[0]+96,y,1,8,Y);

oled.OLED\_Write\_To\_Bufer(x[0]+104,y,1,8,A);

oled.OLED\_Clear\_Bufer\_part(0,y+1,16,8);

oled.OLED\_Write\_To\_Bufer(x[1]+0,y+1,1,8,six);

oled.OLED\_Write\_To\_Bufer(x[1]+8,y+1,1,8,six);

oled.OLED\_Write\_To\_Bufer(x[1]+24,y+1,1,8,six);

oled.OLED\_Write\_To\_Bufer(x[1]+32,y+1,1,8,nine);

oled.OLED\_Clear\_Bufer\_part(0,y+2,16,8);

oled.OLED\_Write\_To\_Bufer(x[2]+0,y+2,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[2]+8,y+2,1,8,M);

oled.OLED\_Write\_To\_Bufer(x[2]+16,y+2,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[2]+24,y+2,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[2]+32,y+2,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[2]+40,y+2,1,8,Y);

oled.OLED\_Write\_To\_Bufer(x[2]+56,y+2,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[2]+64,y+2,1,8,U);

oled.OLED\_Write\_To\_Bufer(x[2]+72,y+2,1,8,B);

oled.OLED\_Write\_To\_Bufer(x[2]+80,y+2,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[2]+88,y+2,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[2]+96,y+2,1,8,A);

oled.OLED\_Clear\_Bufer\_part(0,y+3,16,8);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+3,1,8,eight);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+3,1,8,nine);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+3,1,8,two);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+3,1,8,six);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+3,1,8,zero);

oled.OLED\_Write\_To\_Bufer(x[3]+40,y+3,1,8,seven);

oled.OLED\_Write\_To\_Bufer(x[3]+48,y+3,1,8,five);

oled.OLED\_Write\_To\_Bufer(x[3]+56,y+3,1,8,five);

oled.OLED\_Write\_To\_Bufer(x[3]+64,y+3,1,8,seven);

oled.OLED\_Write\_To\_Bufer(x[3]+72,y+3,1,8,two);

oled.OLED\_Write\_To\_Bufer(x[3]+80,y+3,1,8,five);

}

};

};

#define DV\_INFO\_POINER\_COUNT 0

#define DV\_INFO\_IMG\_X\_0 0

#define DV\_INFO\_IMG\_X\_1 32

#define DV\_INFO\_IMG\_Y 2

class Device\_info:public Menu\_Element

{

protected:

int x[2]={ DV\_INFO\_IMG\_X\_0, DV\_INFO\_IMG\_X\_1};

int y= DV\_INFO\_IMG\_Y;

public:

Device\_info()

{

};

void Default()

{

};

void refresh()

{

Default();

actions();

};

void close()

{

pointer=0;

Device\_Info\_Status=false;

Info\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

close();

};

void previous()

{

close();

};

void choise()

{

close();

};

void back()

{

close();

};

void animate()

{

};

void actions()

{

if(Device\_Info\_Status){

oled.OLED\_Clear\_Bufer\_part(0,y,16,8);

oled.OLED\_Write\_To\_Bufer(x[0]+0,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[0]+8,y,1,8,O);

oled.OLED\_Write\_To\_Bufer(x[0]+16,y,1,8,F);

oled.OLED\_Write\_To\_Bufer(x[0]+24,y,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[0]+32,y,1,8,W);

oled.OLED\_Write\_To\_Bufer(x[0]+40,y,1,8,A);

oled.OLED\_Write\_To\_Bufer(x[0]+48,y,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[0]+56,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[0]+72,y,1,8,V);

oled.OLED\_Write\_To\_Bufer(x[0]+80,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[0]+88,y,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[0]+96,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[0]+104,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[0]+112,y,1,8,O);

oled.OLED\_Write\_To\_Bufer(x[0]+120,y,1,8,N);

oled.OLED\_Clear\_Bufer\_part(0,y+1,16,8);

oled.OLED\_Write\_To\_Bufer(x[1]+0,y+1,1,8,zero);

oled.OLED\_Write\_To\_Bufer(x[1]+8,y+1,1,8,dot);

oled.OLED\_Write\_To\_Bufer(x[1]+16,y+1,1,8,one);

oled.OLED\_Write\_To\_Bufer(x[1]+32,y+1,1,8,B);

oled.OLED\_Write\_To\_Bufer(x[1]+40,y+1,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[1]+48,y+1,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[1]+56,y+1,1,8,A);

}

};

};

#define BRGTNS\_SETTINGS\_MENU\_POINER\_COUNT 10

#define BRGTNS\_SETTINGS\_MENU\_IMG\_X\_0 0

#define BRGTNS\_SETTINGS\_MENU\_IMG\_X\_1 112

#define BRGTNS\_SETTINGS\_MENU\_IMG\_X\_2 8

#define BRGTNS\_SETTINGS\_MENU\_IMG\_X\_3 20

#define BRGTNS\_SETTINGS\_MENU\_IMG\_Y 3

class Bright\_set

{

protected:

int pointer=0;

int x[4]={BRGTNS\_SETTINGS\_MENU\_IMG\_X\_0,BRGTNS\_SETTINGS\_MENU\_IMG\_X\_1,BRGTNS\_SETTINGS\_MENU\_IMG\_X\_2,BRGTNS\_SETTINGS\_MENU\_IMG\_X\_3};

int y=BRGTNS\_SETTINGS\_MENU\_IMG\_Y;

public:

Bright\_set()

{

pointer= brigtnesslvl;

};

void Default()

{

pointer= brigtnesslvl;

oled.OLED\_Write\_To\_Bufer(BRGTNS\_SETTINGS\_MENU\_IMG\_X\_0,y+1,1,16,lt);

oled.OLED\_Write\_To\_Bufer(BRGTNS\_SETTINGS\_MENU\_IMG\_X\_1,y+1,1,16,rt);

};

void refresh()

{

//oled.OLED\_Bufer\_Clear();

Default();

actions();

};

void close()

{

pointer=0;

Settings\_Menu\_Status=true;

Brightness\_Status=false;

oled.OLED\_Bufer\_Clear();

};

void next()

{

pointer++;

if(pointer>BRGTNS\_SETTINGS\_MENU\_POINER\_COUNT+1) pointer=BRGTNS\_SETTINGS\_MENU\_POINER\_COUNT+1;

actions();

brigtnesslvl=pointer;

};

void previous()

{

pointer--;

if(pointer<1) pointer=1;

actions();

brigtnesslvl=pointer;

};

void choise()

{

brigtnesslvl=pointer;

asm("JMP 0");

close();

};

void back()

{

close();

pointer=0;

};

void animate()

{

};

void actions()

{

oled.OLED\_Clear\_Bufer\_part(0,y-1,16,8);

oled.OLED\_Write\_To\_Bufer(x[2]+0,y-1,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[2]+8,y-1,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[2]+16,y-1,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[2]+32,y-1,1,8,B);

oled.OLED\_Write\_To\_Bufer(x[2]+40,y-1,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[2]+48,y-1,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[2]+56,y-1,1,8,G);

oled.OLED\_Write\_To\_Bufer(x[2]+64,y-1,1,8,H);

oled.OLED\_Write\_To\_Bufer(x[2]+72,y-1,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[2]+80,y-1,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[2]+88,y-1,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[2]+96,y-1,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[2]+104,y-1,1,8,S);

if(Brightness\_Status){

switch(pointer)

{

case 0:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

}

case 1:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

break;

}

case 2:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

break;

}

case 3:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

break;

}

case 4:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

break;

}

case 5:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+1,1,16,dfull);

break;

}

case 6:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+40,y+1,1,16,dfull);

break;

}

case 7:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+40,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+48,y+1,1,16,dfull);

break;

}

case 8:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+40,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+48,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+56,y+1,1,16,dfull);

break;

}

case 9:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+40,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+48,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+56,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+64,y+1,1,16,dfull);

break;

}

case 10:

{

oled.OLED\_Clear\_Bufer\_part(x[3],y+1,10,16);

oled.OLED\_Write\_To\_Bufer(x[3]+0,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+8,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+16,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+24,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+32,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+40,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+48,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+56,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+64,y+1,1,16,dfull);

oled.OLED\_Write\_To\_Bufer(x[3]+72,y+1,1,16,dfull);

break;

}

}

}

};

};

#define DR\_POINER\_COUNT 0

#define DR\_IMG\_X\_0 12

#define DR\_IMG\_X\_1 0

#define DR\_IMG\_X\_2 104

#define DR\_IMG\_Y 2

class Device\_reset

{

protected:

int x[3]={ DR\_IMG\_X\_0, DR\_IMG\_X\_1,DR\_IMG\_X\_2};

int y= DR\_IMG\_Y;

public:

Device\_reset()

{

};

void Default()

{

oled.OLED\_Write\_To\_Bufer(x[0]+0,y,1,8,R);

oled.OLED\_Write\_To\_Bufer(x[0]+8,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[0]+16,y,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[0]+24,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[0]+32,y,1,8,T);

oled.OLED\_Write\_To\_Bufer(x[0]+48,y,1,8,D);

oled.OLED\_Write\_To\_Bufer(x[0]+56,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[0]+64,y,1,8,V);

oled.OLED\_Write\_To\_Bufer(x[0]+72,y,1,8,I);

oled.OLED\_Write\_To\_Bufer(x[0]+80,y,1,8,C);

oled.OLED\_Write\_To\_Bufer(x[0]+88,y,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[0]+96,y,1,8,what);

oled.OLED\_Write\_To\_Bufer(x[2]+0,y+2,1,8,Y);

oled.OLED\_Write\_To\_Bufer(x[2]+8,y+2,1,8,E);

oled.OLED\_Write\_To\_Bufer(x[2]+16,y+2,1,8,S);

oled.OLED\_Write\_To\_Bufer(x[1]+0,y+2,1,8,N);

oled.OLED\_Write\_To\_Bufer(x[1]+8,y+2,1,8,O);

};

void refresh()

{

Default();

//actions();

};

void close()

{

Reset\_Status=false;

Settings\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

newdevice=true;

Reset\_Status=false;

asm("JMP 0");

//oled.OLED\_Bufer\_Clear();

};

void previous()

{

close();

};

void choise()

{

};

void back()

{

};

void animate()

{

};

void actions()

{

};

};

#define Add\_Owner\_MENU\_POINER\_COUNT 10

#define Add\_Owner\_MENU\_SIMVOLS\_COUNT 10

#define Add\_Owner\_MENU\_IMG\_X\_0 5

#define Add\_Owner\_MENU\_IMG\_X\_1 16

#define Add\_Owner\_MENU\_IMG\_X\_2 27

#define Add\_Owner\_MENU\_IMG\_X\_3 38

#define Add\_Owner\_MENU\_IMG\_X\_4 49

#define Add\_Owner\_MENU\_IMG\_X\_5 60

#define Add\_Owner\_MENU\_IMG\_X\_6 71

#define Add\_Owner\_MENU\_IMG\_X\_7 82

#define Add\_Owner\_MENU\_IMG\_X\_8 93

#define Add\_Owner\_MENU\_IMG\_X\_9 104

#define Add\_Owner\_MENU\_IMG\_X\_10 115

#define Add\_Owner\_MENU\_IMG\_Y 3

class Add\_Owner

{

protected:

int pointer=0;

int symbol\_pointer=0;

int x[11]={Add\_Owner\_MENU\_IMG\_X\_0,Add\_Owner\_MENU\_IMG\_X\_1,Add\_Owner\_MENU\_IMG\_X\_2,Add\_Owner\_MENU\_IMG\_X\_3,Add\_Owner\_MENU\_IMG\_X\_4,Add\_Owner\_MENU\_IMG\_X\_5,Add\_Owner\_MENU\_IMG\_X\_6,Add\_Owner\_MENU\_IMG\_X\_7,Add\_Owner\_MENU\_IMG\_X\_8,Add\_Owner\_MENU\_IMG\_X\_9,Add\_Owner\_MENU\_IMG\_X\_10};

int y=PASSWORD\_MENU\_IMG\_Y;

public:

Add\_Owner()

{

};

void Default()

{

oled.OLED\_Write\_To\_Bufer(x[0],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[1],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[2],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[3],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[4],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[5],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[6],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[7],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[8],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[9],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[10],y+1,1,8,line);

oled.OLED\_Write\_To\_Bufer(x[11],y+1,1,8,line);

};

void refresh()

{

//oled.OLED\_Bufer\_Clear();

Default();

actions();

};

void close()

{

pointer=0;

symbol\_pointer=0;

User\_Add\_Status=false;

Settings\_Menu\_Status=true;

oled.OLED\_Bufer\_Clear();

};

void next()

{

symbol\_pointer++;

if(symbol\_pointer>Add\_Owner\_MENU\_SIMVOLS\_COUNT) symbol\_pointer=0;

actions();

};

void previous()

{

symbol\_pointer--;

if(symbol\_pointer<0) symbol\_pointer=Add\_Owner\_MENU\_SIMVOLS\_COUNT;

actions();

};

void choise()

{

switch(symbol\_pointer)

{

case 0:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 1:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 2:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 3:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 4:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 5:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 6:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 7:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 8:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 9:

{

pointer++;

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

{

oled.OLED\_Clear\_Bufer\_part(x[i],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[i],y-1,1,8);

}

break;

}

case 10:

{

back();

break;

}

}

if(pointer>Add\_Owner\_MENU\_POINER\_COUNT) {

User\_Add\_Status=false;

Settings\_Menu\_Status=true;

close();

}

else

{

symbol\_pointer=0;

actions();

}

};

void back()

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Clear\_Bufer\_part(x[pointer],y+2,1,8);

oled.OLED\_Clear\_Bufer\_part(x[pointer],y-1,1,8);

pointer--;

if(pointer<0)

{

User\_Add\_Status=false;

Settings\_Menu\_Status=true;

close();

pointer=0;

}

else actions();

};

void animate()

{

};

void actions()

{

if(User\_Add\_Status){

switch(symbol\_pointer)

{

case 0:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,zero);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 1:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,one);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 2:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,two);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 3:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,three);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 4:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,four);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 5:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,five);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 6:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,six);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 7:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,seven);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 8:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,eight);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 9:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,nine);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

case 10:

{

oled.OLED\_Clear\_Bufer\_part(x[pointer],y,1,8);

oled.OLED\_Write\_To\_Bufer(x[pointer],y,1,8,arrow);

oled.OLED\_Write\_To\_Bufer(x[pointer],y+2,1,8,downn);

oled.OLED\_Write\_To\_Bufer(x[pointer],y-1,1,8,upp);

break;

}

}

}

};

};

Info\_menu inf\_menu;

Settings\_Menu sett\_menu;

Password\_Menu passw\_menu;

Main\_Menu menu;

User\_Info us\_inf;

Device\_info dev\_inf;

Bright\_set brgtns;

Device\_reset Dreset;

Add\_Owner adown;

bool btn0=false;

bool btn1=false;

bool btn2=false;

bool btn3=false;

bool btn03=false;

bool btn12=false;

ISR(PCINT0\_vect)

{

//getup();

//cli();

//PORTD=0b10000000;

if(!(PINA&0b00000100)&!btn2 )

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,three);

btn2=true;

}

if(!(PINA&0b00001000)&!btn3 )

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,zero);

btn3=true;

}

if(!(PINA&0b00010000)&!btn0 )

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,two);

btn0=true;

}

if(!(PINA&0b00100000)&!btn1 )

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,one);

btn1=true;

}

//\_delay\_ms(100);

//sei();

//PORTD=0b00000000;

}

void check()

{

*\_delay\_ms*(1000);

cli();

if(btn0 & !btn1 & !btn2 & btn3) btn03=true;

if(!btn0 & btn1 & btn2 & !btn3) btn12=true;

if(btn0 & !btn1 & !btn2 & !btn3 & !btn12 & !btn03)

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,zero);

if (Password\_Menu\_Status)

{

passw\_menu.previous();

//passw\_menu.refresh();

}

else if (Main\_Menu\_Status)

{

menu.previous();

//menu.refresh();

}

else if (Settings\_Menu\_Status)

{

sett\_menu.previous();

//sett\_menu.refresh();

}

else if (Info\_Menu\_Status)

{

inf\_menu.previous();

//sett\_menu.refresh();

}

else if (User\_Info\_Status)

{

us\_inf.close();

}

else if (Device\_Info\_Status)

{

dev\_inf.close();

}

else if (User\_Add\_Status)

{

adown.previous();

}

else if (Reset\_Status)

{

Dreset.previous();

}

else if (Brightness\_Status)

{

brgtns.previous();

}

else if (Waiting\_Status)

{

// .refresh();

}

dreamreset=true;

if(newdevice)

{

newdevice=!newdevice;

Main\_Menu\_Status=true;

}

}

else if(!btn0 & btn1 & !btn2 & !btn3 & !btn12 & !btn03)

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,one);

if (Password\_Menu\_Status)

{

}

else if (Main\_Menu\_Status)

{

}

else if (Settings\_Menu\_Status)

{

}

else if (Info\_Menu\_Status)

{

}

else if (User\_Info\_Status)

{

us\_inf.close();

}

else if (Device\_Info\_Status)

{

dev\_inf.close();

}

else if (User\_Add\_Status)

{

// .refresh();

}

else if (Reset\_Status)

{

// .refresh();

}

else if (Brightness\_Status)

{

// .refresh();

}

else if (Waiting\_Status)

{

// .refresh();

}

if(!locked)

{

locktimer=true;

}

dreamreset=true;

if(newdevice)

{

newdevice=!newdevice;

Main\_Menu\_Status=true;

}

}

else if(!btn0 & !btn1 & btn2 & !btn3 & !btn12 & !btn03)

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,two);

if (Password\_Menu\_Status)

{

}

else if (Main\_Menu\_Status)

{

}

else if (Settings\_Menu\_Status)

{

}

else if (Info\_Menu\_Status)

{

}

else if (User\_Info\_Status)

{

us\_inf.close();

}

else if (Device\_Info\_Status)

{

dev\_inf.close();

}

else if (User\_Add\_Status)

{

// .refresh();

}

else if (Reset\_Status)

{

// .refresh();

}

else if (Brightness\_Status)

{

// .refresh();

}

else if (Waiting\_Status)

{

// .refresh();

}

if(!locked)

{

locktimer=true;

}

dreamreset=true;

if(newdevice)

{

newdevice=!newdevice;

Main\_Menu\_Status=true;

}

}

else if(!btn0 & !btn1 & !btn2 & btn3 & !btn12 & !btn03)

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

//oled.OLED\_Write\_To\_Bufer(0,0,1,8,three);

if (Password\_Menu\_Status)

{

passw\_menu.next();

//passw\_menu.refresh();

}

else if (Main\_Menu\_Status)

{

menu.next();

//menu.refresh();

}

else if (Settings\_Menu\_Status)

{

sett\_menu.next();

//sett\_menu.refresh();

}

else if (Info\_Menu\_Status)

{

inf\_menu.next();

}

else if (User\_Info\_Status)

{

us\_inf.close();

}

else if (Device\_Info\_Status)

{

dev\_inf.close();

}

else if (User\_Add\_Status)

{

adown.next();

}

else if (Reset\_Status)

{

Dreset.next();

}

else if (Brightness\_Status)

{

brgtns.next();

}

else if (Waiting\_Status)

{

// .refresh();

}

dreamreset=true;

if(newdevice)

{

newdevice=!newdevice;

Main\_Menu\_Status=true;

}

}

else if(!btn0 & !btn3 & btn12 & !btn03)

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,four);

if (Password\_Menu\_Status)

{

//passw\_menu.choise();

//passw\_menu.refresh();

}

else if (Main\_Menu\_Status)

{

//menu.choise();

//menu.refresh();

}

else if (Settings\_Menu\_Status)

{

//sett\_menu.choise();

//sett\_menu.refresh();

}

else if (Info\_Menu\_Status)

{

}

else if (User\_Info\_Status)

{

us\_inf.close();

}

else if (Device\_Info\_Status)

{

dev\_inf.close();

}

else if (User\_Add\_Status)

{

// .refresh();

}

else if (Reset\_Status)

{

// .refresh();

}

else if (Brightness\_Status)

{

// .refresh();

}

else if (Waiting\_Status)

{

// .refresh();

}

if(!locked)

{

locktimer=true;

}

dreamreset=true;

if(newdevice)

{

newdevice=!newdevice;

Main\_Menu\_Status=true;

}

}

else if(!btn1 & !btn2 & !btn12 & btn03)

{

// oled.OLED\_Clear\_Bufer\_part(0,0,1,8);

// oled.OLED\_Write\_To\_Bufer(0,0,1,8,five);

if (Password\_Menu\_Status)

{

passw\_menu.choise();

//passw\_menu.refresh();

}

else if (Main\_Menu\_Status)

{

menu.choise();

//menu.refresh();

}

else if (Settings\_Menu\_Status)

{

sett\_menu.choise();

//sett\_menu.refresh();

}

else if (Info\_Menu\_Status)

{

inf\_menu.choise();

//sett\_menu.refresh();

}

else if (User\_Info\_Status)

{

us\_inf.close();

}

else if (Device\_Info\_Status)

{

dev\_inf.close();

}

else if (User\_Add\_Status)

{

adown.choise();

}

else if (Reset\_Status)

{

// .refresh();

}

else if (Brightness\_Status)

{

brgtns.choise();

}

else if (Waiting\_Status)

{

// .refresh();

}

dreamreset=true;

if(newdevice)

{

newdevice=!newdevice;

Main\_Menu\_Status=true;

}

}

//\_delay\_ms(100);

btn0=false;

btn1=false;

btn2=false;

btn3=false;

btn03=false;

btn12=false;

sei();

}

class Batary

{

private:

int curlvl;

public:

Batary()

{

DDRB=0x00;

PORTB=0xFF;

refreshlvl();

// printlvltoOLEDbufer();

}

void refreshlvl()

{

int kf=0;

if((PINB>>0)&(0b00000001==1)) kf++;

if((PINB>>1)&(0b00000001==1)) kf++;

if((PINB>>2)&(0b00000001==1)) kf++;

if((PINB>>3)&(0b00000001==1)) kf++;

curlvl=kf;

}

void printlvltoOLEDbufer()

{

if(curlvl==0) oled.OLED\_Write\_To\_Bufer(96,0,4,8,batary5);

if(curlvl==1) oled.OLED\_Write\_To\_Bufer(96,0,4,8,batary4);

if(curlvl==2) oled.OLED\_Write\_To\_Bufer(96,0,4,8,batary3);

if(curlvl==3) oled.OLED\_Write\_To\_Bufer(96,0,4,8,batary2);

if(curlvl==4) oled.OLED\_Write\_To\_Bufer(96,0,4,8,batary1);

}

void refresh()

{

refreshlvl();

printlvltoOLEDbufer();

}

};

Batary batary;

int j=0;

int i=0;

void asciitoimg(int x,int y,int w,int h,*uint8\_t* data)

{

if (data==0x41)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,A);

else if (data==0x42)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,B);

else if (data==0x43)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,C);

else if (data==0x44)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,D);

else if (data==0x45)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,E);

else if (data==0x46)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,F);

else if (data==0x47)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,G);

else if (data==0x48)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,H);

else if (data==0x49)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,I);

else if (data==0x4A)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,J);

else if (data==0x4B)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,K);

else if (data==0x4C)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,L);

else if (data==0x4D)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,M);

else if (data==0x4E)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,N);

else if (data==0x4F)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,O);

else if (data==0x50)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,P);

else if (data==0x51)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,Q);

else if (data==0x52)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,R);

else if (data==0x53)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,S);

else if (data==0x54)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,T);

else if (data==0x55)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,U);

else if (data==0x56)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,V);

else if (data==0x57)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,W);

else if (data==0x58)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,X);

else if (data==0x59)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,Y);

else if (data==0x5A)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,Z);

else if (data==0x30)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,zero);

else if (data==0x31)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,one);

else if (data==0x32)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,two);

else if (data==0x33)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,three);

else if (data==0x34)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,four);

else if (data==0x35)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,five);

else if (data==0x36)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,six);

else if (data==0x37)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,seven);

else if (data==0x38)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,eight);

else if (data==0x39)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,nine);

else if (data>>4==0)

oled.OLED\_Write\_To\_Bufer(x,y,w,h,NL);

else oled.OLED\_Write\_To\_Bufer(x,y,w,h,what);

}

void send\_Uart(const char c)

{

while(!(UCSR0A&(1<<UDRE0)))

{}

UDR0 = c;

}

void send\_Uart\_str(const char \*s)

{

while (\*s != 0) send\_Uart(\*s++);

}

void USART\_Init()

{

UBRR0L = 19;

UBRR0H = 19 >> 8;

UCSR0B |=(1<<RXEN0)|(1<<RXCIE0)|(1<<TXEN0);

UCSR0C = (0<<USBS0) | (1<<UCSZ00)| (1<<UCSZ01)| (0<<UCSZ02);

*\_delay\_ms*(100);

send\_Uart\_str("AT");

send\_Uart((char)13);

*\_delay\_ms*(100);

send\_Uart\_str("ATE1");

send\_Uart((char)13);

*\_delay\_ms*(100);

send\_Uart\_str("ATV1");

send\_Uart((char)13);

*\_delay\_ms*(100);

}

*uint8\_t* data\_in[100];

bool read=false;

int count=0;

ISR (USART0\_RX\_vect)

{

while(!(UCSR0A&(1<<RXC0))) {};

UCSR0B |=(0<<RXEN0)|(0<<RXCIE0);

data\_in[count] = UDR0;

count++;

/\*asciitoimg((j++)\*8,1,1,8,data\_in[count-1]);\*/

UCSR0B |=(1<<RXEN0)|(1<<RXCIE0);

}

void readfirst()

{

send\_Uart\_str("AT+CMGF=1");

send\_Uart((char)13);

*\_delay\_ms*(1000);

send\_Uart\_str("AT+CMGR=1,1");

send\_Uart((char)13);

}

void deleteall()

{

send\_Uart\_str("AT+CMGDA=\"DEL ALL\"");

send\_Uart((char)13);

*\_delay\_ms*(1000);

send\_Uart\_str("AT+CMGF=1");

send\_Uart((char)13);

*\_delay\_ms*(1000);

}

bool isnum(*uint8\_t* sign)

{

if(sign==0x30||sign==0x31||sign==0x32||sign==0x33||sign==0x34||sign==0x35||sign==0x36||sign==0x37||sign==0x38||sign==0x39) return true; else return false;

}

bool corect\_sender=false;

bool unlock\_comand=false;

bool readinginprocess=false;

bool test=false;

int z=0;

int k=0;

ISR (TIMER1\_COMPA\_vect)

{

if(locktimer)

if(z<100) z++;

else{

if(test){

oled.OLED\_Clear\_Bufer\_part(8,0,1,8);

// oled.OLED\_Write\_To\_Bufer(8,0,1,8,zero);

PORTD=0b00000000;

locktimer=false;

}

else

{

oled.OLED\_Clear\_Bufer\_part(8,0,1,8);

// oled.OLED\_Write\_To\_Bufer(8,0,1,8,one);

PORTD=0b10000000;

}

test=!test;

z=0;

}

else {z=100;}

if(!dream)

{

if(k<50)

{

k++;

oled.OLED\_Clear\_Bufer\_part(16,0,1,8);

//oled.OLED\_Write\_To\_Bufer(16,0,1,8,zero);

}

else{

oled.OLED\_Clear\_Bufer\_part(16,0,1,8);

// oled.OLED\_Write\_To\_Bufer(16,0,1,8,two);

dream=true;

}

}

if(dreamreset)

{

k=0;

dreamreset=false;

dream=false;

}

}

void sleepc(void)

{

cli();

set\_sleep\_mode(SLEEP\_MODE\_PWR\_DOWN);

sei();

sleep\_mode();

cli();

}

void oledwritenum(int num,int x,int y)

{

if(num==0)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,zero);

}

else if(num==1)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,one);

}

else if(num==2)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,two);

}

else if(num==3)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,three);

}

else if(num==4)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,four);

}

else if(num==5)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,five);

}

else if(num==6)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,six);

}

else if(num==7)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,seven);

}

else if(num==8)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,eight);

}

else if(num==9)

{

oled.OLED\_Clear\_Bufer\_part(x,y,1,8);

oled.OLED\_Write\_To\_Bufer(x,y,1,8,nine);

}

}

void send\_SMS(char \*text,char \*number)

{

send\_Uart\_str("AT+CMGF=1");

send\_Uart((char)13);

*\_delay\_ms*(250);

send\_Uart\_str("AT+CMGS=\"");

send\_Uart\_str(number);

send\_Uart\_str("\"");

send\_Uart((char)13);

*\_delay\_ms*(250);

send\_Uart\_str(text);

send\_Uart((char)26);

*\_delay\_ms*(250);

}

void SMStranslator(int j)

{

if (count>j) {

if(count==j)

{

j=0;

count=0;

readinginprocess=false;

}

if(!readinginprocess)

{

if(count-j>3)

{

if(data\_in[j]=='C'&data\_in[j+1]=='M'&data\_in[j+2]=='T'&data\_in[j+3]=='I')

{

readinginprocess=true;

oled.OLED\_Write\_To\_Bufer((i++)\*8,1,1,8,C);

readfirst();

deleteall();

}

}

}

else

{

if(count>6)

{

if(data\_in[j]=='0'&data\_in[j+1]=='7'&data\_in[j+2]=='5'&data\_in[j+3]=='5'&data\_in[j+4]=='7'&data\_in[j+5]=='2'&data\_in[j+6]=='5')

{

oled.OLED\_Write\_To\_Bufer((i++)\*8,1,1,8,N);

corect\_sender=true;

j+=6;

}

}

if(corect\_sender)

{

if(j>5)

{

if(data\_in[j]=='U'&data\_in[j+1]=='N'&data\_in[j+2]=='L'&data\_in[j+3]=='O'\*data\_in[j+4]=='C'&data\_in[j+5]=='K')

{

oled.OLED\_Write\_To\_Bufer((i++)\*8,1,1,8,U);

unlock\_comand=true;

j+=5;

}

}

}

if(unlock\_comand)

{

if(j>4)

{

if(isnum(data\_in[j])&isnum(data\_in[j+1])&isnum(data\_in[j+2])&isnum(data\_in[j+3])&isnum(data\_in[j+4]))

{

if(data\_in[j]==password[0]&data\_in[j+1]==password[1]&data\_in[j+2]==password[2]&data\_in[j+3]==password[3]&data\_in[j+4]==password[4]){

oled.OLED\_Write\_To\_Bufer((i++)\*8,1,1,8,U);

locked=false;

PORTD=0b10000000;

readinginprocess=false;

j+=4;

}

}

}

}

}

j++;

}

}

int main(void){

sei();

oled.OLED\_Write\_To\_Bufer(48,2,4,16,SL);

oled.OLED\_Write\_To\_Bufer(24,6,1,8,L);

oled.OLED\_Write\_To\_Bufer(32,6,1,8,O);

oled.OLED\_Write\_To\_Bufer(40,6,1,8,A);

oled.OLED\_Write\_To\_Bufer(48,6,1,8,D);

oled.OLED\_Write\_To\_Bufer(56,6,1,8,I);

oled.OLED\_Write\_To\_Bufer(64,6,1,8,N);

oled.OLED\_Write\_To\_Bufer(72,6,1,8,G);

oled.OLED\_Write\_To\_Bufer(80,6,1,8,dot);

oled.OLED\_Write\_To\_Bufer(88,6,1,8,dot);

oled.OLED\_Write\_To\_Bufer(96,6,1,8,dot);

oled.OLED\_Write\_Bufer();

oled.OLED\_Bufer\_Clear();

DDRD = 0b10000000;

PCICR|=0b00000001;

PCMSK0=0b00111100;

TCCR1A = 0;

TCCR1B = (1<<WGM12)|(5<<CS10);

OCR1A = 0b11111111;

TIMSK1 |= (1<<OCIE1A);

TIFR1 = (1<<OCF1A);

*\_delay\_ms*(30000);

//\_delay\_ms(1000);

USART\_Init();

*\_delay\_ms*(250);

//deleteall();

//\_delay\_ms(250);

//send\_Uart\_str("AT");

//send\_Uart((char)13);

//\_delay\_ms(250);

while(1)

{

/\*

oledwritenum(password[0],8,1);

oledwritenum(password[1],16,1);

oledwritenum(password[2],24,1);

oledwritenum(password[3],32,1);

oledwritenum(password[4],40,1);

oledwritenum(passwin[0],56,1);

oledwritenum(passwin[1],64,1);

oledwritenum(passwin[2],72,1);

oledwritenum(passwin[3],80,1);

oledwritenum(passwin[4],88,1);\*/

/\*

if(dream)

{

cli();

set\_sleep\_mode(SLEEP\_MODE\_PWR\_DOWN);

sei();

sleep\_mode();

cli();

}\*/

sei();

if ((count>0)&(!(UCSR0A &(1<<RXC0)))) {

//oled.OLED\_Clear\_Bufer\_part((j)\*8,1,1,8);

//asciitoimg((j)\*8,1,1,8,data\_in[j]);

if(count-j>3)

{

if((data\_in[j]=='C') & (data\_in[j+1]=='M') & (data\_in[j+2]=='T') & (data\_in[j+3]=='I'))

{

readinginprocess=true;

//oled.OLED\_Write\_To\_Bufer((i++)\*8,1,1,8,C);

//readfirst();

count=1;

j=-1;

// deleteall();

// send\_SMS("Device status changed",owner\_number);

//PORTD=0b10000000;

locked=!locked;

}

}

j++;

count--;

}

if(count==0){

j=0;

}

check();

if (Password\_Menu\_Status)

{

passw\_menu.refresh();

}

else if (Main\_Menu\_Status)

{

menu.refresh();

}

else if (Settings\_Menu\_Status)

{

sett\_menu.refresh();

}

else if (Info\_Menu\_Status)

{

inf\_menu.refresh();

}

else if (User\_Info\_Status)

{

us\_inf.refresh();

}

else if (Device\_Info\_Status)

{

dev\_inf.refresh();

}

else if (User\_Add\_Status)

{

adown.refresh();

}

else if (Reset\_Status)

{

Dreset.refresh();

}

else if (Brightness\_Status)

{

brgtns.refresh();

}

else if (Waiting\_Status)

{

// .refresh();

}

else if (newdevice)

{

int x=8;

int q=28;

oled.OLED\_Write\_To\_Bufer(x+0,3,1,8,Y);

oled.OLED\_Write\_To\_Bufer(x+8,3,1,8,O);

oled.OLED\_Write\_To\_Bufer(x+16,3,1,8,U);

oled.OLED\_Write\_To\_Bufer(x+24,3,1,8,R);

oled.OLED\_Write\_To\_Bufer(x+40,3,1,8,I);

oled.OLED\_Write\_To\_Bufer(x+48,3,1,8,N);

oled.OLED\_Write\_To\_Bufer(x+56,3,1,8,I);

oled.OLED\_Write\_To\_Bufer(x+64,3,1,8,T);

oled.OLED\_Write\_To\_Bufer(x+80,3,1,8,C);

oled.OLED\_Write\_To\_Bufer(x+88,3,1,8,O);

oled.OLED\_Write\_To\_Bufer(x+96,3,1,8,D);

oled.OLED\_Write\_To\_Bufer(x+104,3,1,8,E);

oled.OLED\_Write\_To\_Bufer(q+0,4,1,8,two);

oled.OLED\_Write\_To\_Bufer(q+16,4,1,8,nine);

oled.OLED\_Write\_To\_Bufer(q+32,4,1,8,six);

oled.OLED\_Write\_To\_Bufer(q+48,4,1,8,zero);

oled.OLED\_Write\_To\_Bufer(q+64,4,1,8,eight);

}

batary.refresh();

oled.OLED\_Write\_To\_Bufer(80,0,2,8,Net3);

oled.OLED\_Write\_Bufer();

// PORTD=0b10000000;

// PORTD=0b00000000;

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Изм. | Лист | № докум. | Подп. | Дата |
|  |  |  |  |  |
| Инв. № подл. | Подп. И дата | Взам. Инв. № | Инв. № дубл. | Подп. и дата |
| RU.17701729.01.01-01 12 01-1 |  |  |  |  |

**3. Текст программы модели корпуса**

**3.1. Файл bot.scad**

union(){

translate([90, -210, 80]) rotate([-90, 0, 0])

difference() {

union(){

difference() {

union(){

rotate([0, 0, 0]) translate([10, 10, 0]) cube([60,60,200]);

rotate([0, 0, 0]) translate([10, 0, 0]) cube([60,10,200]);

rotate([0, 0, 0]) translate([0, 10, 0]) cube([10,60,200]);

rotate([0, 0, 0]) translate([10, 70, 0]) cube([60,10,200]);

rotate([0, 0, 0]) translate([70, 10, 0]) cube([10,60,200]);

rotate([0, 0, 0]) translate([10, 10, 0]) cylinder(200,10,10,$fn=50);

rotate([0, 0, 0]) translate([10, 70, 0]) cylinder(200,10,10,$fn=50);

rotate([0, 0, 0]) translate([70, 10, 0]) cylinder(200,10,10,$fn=50);

rotate([0, 0, 0]) translate([70, 70, 0]) cylinder(200,10,10,$fn=50);

}

rotate([0, 0, 0]) translate([15, 15, -1]) cube([50,50,202]);

rotate([0, 0, 0]) translate([-1, -35, -1]) cube([82,80,202]);

rotate([0, 0, 0]) translate([40, 81, 42.5])

rotate([90, 0, 0]) cylinder(17,8.5,8.5,$fn=50);

rotate([0, 0, 0]) translate([40, 81, 157.5])

rotate([90, 0, 0]) cylinder(17,8.5,8.5,$fn=50);

//

// rotate([0, 0, 0]) translate([9, 5,-1]) cube([12,60,200]);

//rotate([0, 0, 0]) translate([59, 5,-1]) cube([12,60,200]);

//

}

rotate([0, 0, 0]) translate([15, 40, 177]) cube([50,25,20]);

rotate([0, 0, 0]) translate([65, 36, 0]) cube([15,10,200]);

rotate([0, 0, 0]) translate([0, 36, 0]) cube([15,10,200]);

rotate([0, 0, 0]) translate([15, 55, 18]) cube([50,10,5]);

}

rotate([0, 0, 0]) translate([15, 65, -1]) cylinder(202,6,6,$fn=50);

rotate([0, 0, 0]) translate([65, 65, -1]) cylinder(202,6,6,$fn=50);

rotate([0, 0, 0]) translate([19, 72.5, 190])

rotate([90, 0, 0]) cylinder(65,10,10,$fn=50);

rotate([0, 0, 0]) translate([40, 72.5, 190])

rotate([90, 0, 0]) cylinder(65,10,10,$fn=50);

rotate([0, 0, 0]) translate([61, 72.5, 190])

rotate([90, 0, 0]) cylinder(65,10,10,$fn=50);

rotate([0, 0, 0]) translate([9, 72.5, 190])

rotate([90, 0, 0]) cube([62,60,40]);

rotate([0, 0, 0]) translate([64, 35, 23]) cube([7,30,154]);

rotate([0, 0, 0]) translate([-1, 34, -1]) cube([6,11,202]);

rotate([0, 0, 0]) translate([75, 34, -1]) cube([6,11,202]);

rotate([0, 0, 0]) translate([9, 35, 23]) cube([7,30,154]);

rotate([0, 0, 0]) translate([9, 34, -1]) cube([7,21,30]);

rotate([0, 0, 0]) translate([64, 34, -1]) cube([7,21,30]);

rotate([0, 0, 0]) translate([9, 35, -1]) cube([7,30,19]);

rotate([0, 0, 0]) translate([64, 35, -1]) cube([7,30,19]);

rotate([0, 0, 0]) translate([9, 10, 170]) cube([7,30,19]);

rotate([0, 0, 0]) translate([64, 10, 170]) cube([7,30,19]);

rotate([0, 0, 0]) translate([9, 72.5, 183]) cube([62,2,20]);

rotate([0, 0, 0]) translate([20, 7.5, 183]) cube([40,65,10]);

//

rotate([0, 0, 0]) translate([29.5, 55, 170])

rotate([0, 0, 0]) cylinder(30,2.5,2.5,$fn=50);

rotate([0, 0, 0]) translate([50.5, 55, 170])

rotate([0, 0, 0]) cylinder(30,2.5,2.5,$fn=50);

rotate([0, 0, 0]) translate([74, 74, 185])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([6, 74, 185])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([40, 77.25, 185])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([40, 77.25, -1])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([74, 74, -1])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([6, 74, -1])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([19, 9,-1]) cube([42,62,19]);

rotate([0, 0, 0]) translate([15, 41,-1]) cube([50,30,19]);

rotate([0, 0, 0]) translate([15, 11,23]) cube([50,60,154]);

rotate([0, 0, 0]) translate([0, 35, -1]) cube([80,10,12]);

//

}

}

**3.2. Файл buttons.scad**

translate([10,10, 0]) rotate([0, 0, 0])

union()

{

difference() {

union()

{

rotate([0, 0, 0]) translate([0,0, 0]) cylinder(25,4,4,$fn=50);

rotate([0, 0, 0]) translate([0,0, 0]) cylinder(3,6,6,$fn=50);

rotate([0, 0, 0]) translate([0,0, 25]) sphere(4,$fn=50);

}

rotate([0, 0, 0]) translate([0,0, -1]) cylinder(8,2.5,2.5,$fn=50);

}

}

union()

{

translate([30,10, 0]) rotate([0, 0, 0])

difference() {

union()

{

rotate([0, 0, 0]) translate([0,0, 0]) cylinder(25,4,4,$fn=50);

rotate([0, 0, 0]) translate([0,0, 0]) cylinder(3,6,6,$fn=50);

rotate([0, 0, 0]) translate([0,0, 25]) sphere(4,$fn=50);

}

rotate([0, 0, 0]) translate([0,0, -1]) cylinder(8,2.5,2.5,$fn=50);

}

}

**3.3. Файл left.scad**

//bottom part

union(){

rotate([0, 0, 0]) translate([20, -100, 0])

difference() {

union(){

rotate([0, 0, 0]) translate([10, 10, 0]) cube([60,60,3]);

rotate([0, 0, 0]) translate([10, 0, 0]) cube([60,10,3]);

rotate([0, 0, 0]) translate([0, 10, 0]) cube([10,60,3]);

rotate([0, 0, 0]) translate([10, 70, 0]) cube([60,10,3]);

rotate([0, 0, 0]) translate([70, 10, 0]) cube([10,60,3]);

rotate([0, 0, 0]) translate([10, 10, 0]) cylinder(3,10,10,$fn=100);

rotate([0, 0, 0]) translate([10, 70, 0]) cylinder(3,10,10,$fn=100);

rotate([0, 0, 0]) translate([70, 10, 0]) cylinder(3,10,10,$fn=100);

rotate([0, 0, 0]) translate([70, 70, 0]) cylinder(3,10,10,$fn=100);

}

rotate([0, 0, 0]) translate([6, 6, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([74, 6, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([40, 2.75, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([6, 74, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([74, 74, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([40, 77.25, -1]) cylinder(7,1.5,1.5,$fn=100);

}}

**3.4. Файл right.scad**

//bottom part

union(){

rotate([0, 0, 0]) translate([50, -100, 0])

difference() {

union(){

rotate([0, 0, 0]) translate([10, 10, 0]) cube([60,60,3]);

rotate([0, 0, 0]) translate([10, 0, 0]) cube([60,10,3]);

rotate([0, 0, 0]) translate([0, 10, 0]) cube([10,60,3]);

rotate([0, 0, 0]) translate([10, 70, 0]) cube([60,10,3]);

rotate([0, 0, 0]) translate([70, 10, 0]) cube([10,60,3]);

rotate([0, 0, 0]) translate([10, 10, 0]) cylinder(3,10,10,$fn=100);

rotate([0, 0, 0]) translate([10, 70, 0]) cylinder(3,10,10,$fn=100);

rotate([0, 0, 0]) translate([70, 10, 0]) cylinder(3,10,10,$fn=100);

rotate([0, 0, 0]) translate([70, 70, 0]) cylinder(3,10,10,$fn=100);

}

rotate([90, 0, 0]) translate([35.5,-3, -80+38]) cube([9,11,4]);

rotate([0, 0, 0]) translate([6, 6, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([74, 6, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([40, 2.75, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([6, 74, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([74, 74, -1]) cylinder(7,1.5,1.5,$fn=100);

rotate([0, 0, 0]) translate([40, 77.25, -1]) cylinder(7,1.5,1.5,$fn=100);

}

}

**3.5. Файл top.scad**

union(){

translate([80, -210, 0]) rotate([90, 0, 180])

difference() {

union(){

difference() {

union()

{

rotate([0, 0, 0]) translate([10, 10, 0]) cube([60,60,200]);

rotate([0, 0, 0]) translate([10, 0, 0]) cube([60,10,200]);

rotate([0, 0, 0]) translate([0, 10, 0]) cube([10,60,200]);

rotate([0, 0, 0]) translate([10, 70, 0]) cube([60,10,200]);

rotate([0, 0, 0]) translate([70, 10, 0]) cube([10,60,200]);

rotate([0, 0, 0]) translate([10, 10, 0]) cylinder(200,10,10,$fn=50);

rotate([0, 0, 0]) translate([10, 70, 0]) cylinder(200,10,10,$fn=50);

rotate([0, 0, 0]) translate([70, 10, 0]) cylinder(200,10,10,$fn=50);

rotate([0, 0, 0]) translate([70, 70, 0]) cylinder(200,10,10,$fn=50);

}

rotate([0, 0, 0]) translate([15, 15, -1]) cube([50,50,202]);

rotate([0, 0, 0]) translate([-1, 35, -1]) cube([82,80,202]);

rotate([0, 0, 0]) translate([22.5, -1, 82.5]) cube([35,17,35]);

rotate([0, 0, 0]) translate([20, -1, 80]) cube([40,3.5,40]);

rotate([0, 0, 0]) translate([40, 16, 70])

rotate([90, 0, 0]) cylinder(17,5,5,$fn=100);

rotate([0, 0, 0]) translate([40, 16, 130])

rotate([90, 0, 0]) cylinder(17,5,5,$fn=100);

}

rotate([0, 0, 0]) translate([15, 15, 177]) cube([50,25,20]);

rotate([0, 0, 0]) translate([65, 35, 0]) cube([15,10,200]);

rotate([0, 0, 0]) translate([0, 35, 0]) cube([15,10,200]);

rotate([0, 0, 0]) translate([15, 15, 18]) cube([50,10,5]);

}

rotate([0, 0, 0]) translate([15, 15, -1]) cylinder(202,6,6,$fn=50);

rotate([0, 0, 0]) translate([65, 15, -1]) cylinder(202,6,6,$fn=50);

rotate([0, 0, 0]) translate([19, 72.5, 190])

rotate([90, 0, 0]) cylinder(65,10,10,$fn=50);

rotate([0, 0, 0]) translate([40, 72.5, 190])

rotate([90, 0, 0]) cylinder(65,10,10,$fn=50);

rotate([0, 0, 0]) translate([61, 72.5, 190])

rotate([90, 0, 0]) cylinder(65,10,10,$fn=50);

rotate([0, 0, 0]) translate([9, 47.5, 190])

rotate([90, 0, 0]) cube([62,15,40]);

rotate([0, 0, 0]) translate([71, 36, -1]) cube([4.75,11,202]);

rotate([0, 0, 0]) translate([4.25, 36, -1]) cube([4.75,11,202]);//

rotate([0, 0, 0]) translate([9, 5.5, 183]) cube([62,2,20]);

rotate([0, 0, 0]) translate([20, 7.5, 183]) cube([40,65,10]);

rotate([0, 0, 0]) translate([9, 15, 23]) cube([7,35,154]);

rotate([0, 0, 0]) translate([64, 15, 23]) cube([7,35,154]);

rotate([0, 0, 0]) translate([9, 15, -1]) cube([7,40,19]);

rotate([0, 0, 0]) translate([64, 15, -1]) cube([7,40,19]);

rotate([0, 0, 0]) translate([9, 25, -1]) cube([7,21,25]);

rotate([0, 0, 0]) translate([64, 25, -1]) cube([7,21,25]);

rotate([0, 0, 0]) translate([9, 40, 170]) cube([7,21,25]);

rotate([0, 0, 0]) translate([64, 40, 170]) cube([7,21,25]);

rotate([0, 0, 0]) translate([30, 25, 170])

rotate([0, 0, 0]) cylinder(30,2.5,2.5,$fn=50);

rotate([0, 0, 0]) translate([50, 25, 170])

rotate([0, 0, 0]) cylinder(30,2.5,2.5,$fn=50);

rotate([0, 0, 0]) translate([74, 6, 185])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([6, 6, 185])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([40, 2.75, 185])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([40, 2.75, -1])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([74, 6, -1])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([6, 6, -1])

rotate([0, 0, 0]) cylinder(16,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([22.5, 16, 27.5])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([(80-22.5), 16, 27.5])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([22.5, 16, (200-27.5)])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([(80-22.5), 16, (200-27.5)])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([22.5, 16, 27.5+25])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([(80-22.5), 16, 27.5+25])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([22.5, 16, (200-27.5-25)])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([(80-22.5), 16, (200-27.5-25)])

rotate([90, 0, 0]) cylinder(8,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([19, 9,-1]) cube([42,62,19]);

rotate([0, 0, 0]) translate([15, 9,-1]) cube([50,30,19]);

//

rotate([0, 0, 0]) translate([6.75, 36, 5])

rotate([90, 0, 0]) cylinder(6,1.5,1.5,$fn=50);

rotate([0, 0, 0]) translate([80-6.75, 36, 5])

rotate([90, 0, 0]) cylinder(6,1.5,1.5,$fn=50);

}

}

**3.6. Файл U.scad**

translate([100, -30, 10])rotate([0, 0, 0])

union(){

difference() {

union()

{

translate([-2.5, 0, 0])rotate([0, 0, 0])

union()

{

rotate([180, 0, 0]) translate([10, 0, 0]) difference() {

rotate\_extrude(angle=360, convexity=15) translate([20, 0]) circle(7.5);

translate([-50, 0, -10]) cube([200,50,20]);

translate([-50, -50, -10]) cube([50,60,20]);

}

}

translate([-57.5, 0, 0])rotate([0, 180, 0])

union()

{

rotate([180, 0, 0]) translate([10, 0, 0]) difference() {

rotate\_extrude(angle=360, convexity=15) translate([20, 0]) circle(7.5);

translate([-50, 0, -10]) cube([200,50,20]);

translate([-50, -50, -10]) cube([50,60,20]);

}

}

rotate([90, 0, 0]) translate([27.5, 0, 0]) cylinder(85,7.5,7.5,$fn=100);

rotate([90, 0, 0]) translate([-87.5, 0, 0]) cylinder(85,7.5,7.5,$fn=100);

rotate([0, 90, 0]) translate([0, 20, -67.5]) cylinder(75,7.5,7.5,$fn=100);

}

translate([-100, -70, -3.5]) cube([200,12,7]);

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Изм. | Лист | № докум. | Подп. | Дата |
|  |  |  |  |  |
| Инв. № подл. | Подп. И дата | Взам. Инв. № | Инв. № дубл. | Подп. и дата |
| RU.17701729.01.01-01 12 01-1 |  |  |  |  |

**Лист регистрации изменений**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в докум. | № докум. | Входящий № сопроводительного докум. и дата | Подпись | Дата |
| Измененных | Замененных | Новых | Анулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Изм. | Лист | № докум. | Подп. | Дата |
|  |  |  |  |  |
| Инв. № подл. | Подп. И дата | Взам. Инв. № | Инв. № дубл. | Подп. и дата |
| RU.17701729.01.01-01 12 01-1 |  |  |  |  |