

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.

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

Москва 2019

ПРОГРАММНО-АППАРАТНЫЙ КОМПЛЕКС УПРАВЛЕНИЯ УМНЫМ ВЕЛОСИПЕДНЫМ ЗАМКОМ

Текст программы
RU.17701729.01.01-01 12 01-1

Листов 67

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

Содержание

1. Основные термины и определения	3
2. Текст программы комплекса	4
2.1. Файл main.cpp	4
2.2. Файл Defines.h	35
2.3. Файл OLED.h	35
2.4. Файл OLED.cpp	35
2.5. Файл TWI.h	37
2.6. Файл TWI.cpp	37
2.7. Файл Images.h	38
3. Текст программы модели корпуса	50
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.scad	57
Приложение 1. Текст прототипа программы для ОС Android “SmartLock”	59
Лист регистрации изменений	67

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 двуправленные 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.

SPI - последовательный синхронный стандарт передачи данных в режиме полного дуплекса, предназначенный для обеспечения простого и недорогого высокоскоростного сопряжения микроконтроллеров и периферии. SPI также иногда называют четырёхпроводным интерфейсом.

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 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_POINTER_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_POINTER_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= brightnesslvl;
    };

    void Default()
    {
        pointer= brightnesslvl;
        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();
        brightnesslvl=pointer;
    };

    void previous()
    {
        pointer--;
        if(pointer<1) pointer=1;
        actions();
        brightnesslvl=pointer;
    };

    void choise()
    {
        brightnesslvl=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("\n");
    send_Uart((char)13);
    _delay_ms(250);
    send_Uart_str(text);
    send_Uart((char)26);
    _delay_ms(250);
}

void SMStranlator(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)
            {
                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;
}
}

```

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 2000000UL
#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_SendAddress();
    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_SendAddress(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_SendAddress(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 Images.h

```
uint8_t six[]={
    0b00000000,
    0b00111100,
    0b00100000,
    0b00111100,
    0b00100100,
    0b00100100,
```

```
        0b00111100,  
        0b00000000  
};  
  
uint8_t seven[]={  
    0b00000000,  
    0b00111100,  
    0b00000100,  
    0b00001000,  
    0b00010000,  
    0b00100000,  
    0b00100000,  
    0b00100000,  
    0b00000000  
};  
  
uint8_t eight[]={  
    0b00000000,  
    0b00111100,  
    0b00100100,  
    0b00111100,  
    0b00100100,  
    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  
};
```


$\} ;$

[illegible]

```

0B11100000,
0B11110000,
0B11111000,
0B11111100,
0B11111110,
0B11111111,
0B11111111,
0B11111110,
0B11111100,
0B11111000,
0B11110000,
0B11100000,
0B11000000,
0B10000000,
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, 0B01000000,
    0B00000001, 0B11000000, 0B00000011, 0B10000000,
    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, 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,
    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
};

```

[illegible]

```

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,
    0B01100110,
    0B00000000
};

uint8_t B[] =
{
    0B00000000,
    0B01111100,
    0B01000010,
    0B01111100,
    0B01000010,
    0B01000010,

```

```

        0B01111100,
        0B00000000
};

uint8_t C[] =
{
    0B00000000,
    0B00111110,
    0B01100000,
    0B01000000,
    0B01000000,
    0B01000000,
    0B01100000,
    0B00111110,
    0B00000000
};

uint8_t D[] =
{
    0B00000000,
    0B01111000,
    0B01000100,
    0B01000010,
    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,
    0B00011000,
    0B00111100,

```

```

0B00000000
};

```

```

uint8_t J[] =
{
    0B00000000,
    0B00000110,
    0B00000110,
    0B00000110,
    0B00000110,
    0B00000110,
    0B01100110,
    0B00111100,
    0B00000000
};

```

```

uint8_t K[] =
{
    0B00000000,
    0B01000100,
    0B01001000,
    0B01110000,
    0B01110000,
    0B01110000,
    0B01001000,
    0B01000100,
    0B00000000
};

```

```

uint8_t L[] =
{
    0B00000000,
    0B01000000,
    0B01000000,
    0B01000000,
    0B01000000,
    0B01000000,
    0B01000000,
    0B01111110,
    0B00000000
};

```

```

uint8_t M[] =
{
    0B00000000,
    0B01000010,
    0B01100110,
    0B01011010,
    0B01000010,
    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,
    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,
    0B00011000,
    0B00000000
};

uint8_t U[] =
{
    0B00000000,
    0B01000010,
    0B01000010,
    0B01000010,
    0B01000010,
    0B01000010,
    0B00111100,
    0B00000000
};

uint8_t V[] =
{
    0B00000000,
    0B01000010,
    0B01000010,
    0B01000010,
    0B01000010,
    0B00100100,
    0B00100100,
    0B00011000,
    0B00000000
};

uint8_t W[] =
{
    0B00000000,
    0B01000010,
    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,
    0B00011000,
    0B00000000
};

uint8_t Z[] =
{
    0B00000000,
    0B01111110,
    0B00000100,
    0B00001000,
    0B00010000,
    0B00100000,
    0B01111110,
    0B00000000
};

uint8_t what[] =
{
    0B00000000,
    0B01111110,
    0B01000100,
    0B00011000,
    0B00011000,
    0B00000000,
    0B00000000,
    0B00011000,
    0B00000000
};

uint8_t NL[] =
{
    0B00000000,
    0B01111100,
    0B00001100,
    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,

```

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				

Приложение 1.

Текст прототипа программы для ОС Android “SmartLock”

MainActivity.java:

```
package com.example.smartlock;

import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart ()
    {
        super.onStart();
        SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
        if(myPreferences.getBoolean("NEW", true))
        {
            Intent intent = new Intent(MainActivity.this, StartRegActivity.class);
            startActivity(intent);
        }
        else
        {
            Intent intent = new Intent(MainActivity.this, MenuActivity.class);
            startActivity(intent);
        }
    }
}
```

MenuActivity.java:

```
package com.example.smartlock;

import android.app.Notification;
import android.app.NotificationManager;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.Vibrator;
import android.preference.PreferenceManager;
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.NotificationManagerCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.telephony.SmsMessage;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import java.nio.channels.Channel;

public class MenuActivity extends AppCompatActivity {

    @Override
```

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_menu);
    IntentFilter sms=new IntentFilter("android.provider.Telephony.SMS_RECEIVED");
    sms.setPriority(9999999);
    registerReceiver(SMSReceiverrr,sms);
}

public void sendSMS(String phoneNo, String msg) {
    try {
        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage(phoneNo, null, msg, null, null);
        Toast.makeText(getApplicationContext(), "Message Sent",
            Toast.LENGTH_LONG).show();
    } catch (Exception ex) {
        Toast.makeText(getApplicationContext(), ex.getMessage().toString(),
            Toast.LENGTH_LONG).show();
        ex.printStackTrace();
    }
}

public void unlock(android.view.View view) {
    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(MenuActivity.this);
    startService(new Intent(this, SMSService.class));
    Toast.makeText(getApplicationContext(), "unlocked", Toast.LENGTH_LONG).show();
    sendSMS(myPreferences.getString("DNUMBER", ""), "unlock");
}

public void lock(android.view.View view) {
    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(MenuActivity.this);
    startService(new Intent(this, SMSService.class));
    Toast.makeText(getApplicationContext(), "locked", Toast.LENGTH_LONG).show();
    sendSMS(myPreferences.getString("DNUMBER", ""), "lock");
}

public void opensettings(android.view.View view) {
    Intent intent = new Intent(MenuActivity.this, SettingsActivity.class);
    startActivity(intent);
}

public void sendmessage(android.view.View view) {
    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(MenuActivity.this);
    TextView txt = (TextView)findViewById(R.id.txt);
    sendSMS(myPreferences.getString("DNUMBER", ""), txt.getText().toString());
}

BroadcastReceiver SMSReceiverrr = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(MenuActivity.this);
        if (intent.getAction() == "android.provider.Telephony.SMS_RECEIVED") {
            Toast.makeText(getApplicationContext(), "tost by sms", Toast.LENGTH_LONG).show();
            TextView txt = (TextView)findViewById(R.id.textView5);
            Object[] pduArray = (Object[]) intent.getExtras().get("pdus");
            SmsMessage[] messages = new SmsMessage[pduArray.length];
            for (int i = 0; i < pduArray.length; i++) {
                messages[i] = SmsMessage.createFromPdu((byte[]) pduArray[i]);
            }
            String sms_from = (messages[0].getDisplayOriginatingAddress());
            String sms_fromD=myPreferences.getString("DNUMBER", "");
            sms_fromD="+7"+sms_fromD.substring(1, sms_fromD.length());
            if (sms_from.equalsIgnoreCase(sms_fromD)) {
                StringBuilder bodyText = new StringBuilder();
                for (int i = 0; i < messages.length; i++) {
                    bodyText.append(messages[i].getMessageBody());
                }
                String body = bodyText.toString();

                txt.setText(body);
            }
        }
    }
};
}

```

RegistrationActivity.java:

```
package com.example.smartlock;

import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class RegistrationActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration);

        SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(RegistrationActivity.this);

        TextView nm = (TextView) findViewById(R.id.editText5);
        TextView sn = (TextView) findViewById(R.id.editText6);
        TextView ad = (TextView) findViewById(R.id.editText3);
        TextView nu = (TextView) findViewById(R.id.editText4);
        TextView dn = (TextView) findViewById(R.id.editText2);
        TextView cd = (TextView) findViewById(R.id.editText7);

        nm.setText(myPreferences.getString("NAME", ""));
        sn.setText(myPreferences.getString("SYENAME", ""));
        ad.setText(myPreferences.getString("ADDRESS", ""));
        nu.setText(myPreferences.getString("NUMBER", ""));
        dn.setText(myPreferences.getString("DNUMBER", ""));
        cd.setText(myPreferences.getString("CODE", ""));

    }

    public void endregistration(android.view.View view)
    {
        TextView nm = (TextView) findViewById(R.id.editText5);
        TextView sn = (TextView) findViewById(R.id.editText6);
        TextView ad = (TextView) findViewById(R.id.editText3);
```

```

    TextView nu = (TextView) findViewById(R.id.editText4);
    TextView dn = (TextView) findViewById(R.id.editText2);
    TextView cd = (TextView) findViewById(R.id.editText7);

    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(RegistrationActivity.this);

    SharedPreferences.Editor myEditor = myPreferences.edit();

    myEditor.putString("NAME", nm.getText().toString());
    myEditor.putString("SYENAME", sn.getText().toString());
    myEditor.putString("ADDRESS", ad.getText().toString());
    myEditor.putString("NUMBER", nu.getText().toString());
    myEditor.putString("DNUMBER", dn.getText().toString());
    myEditor.putString("CODE", cd.getText().toString());
    myEditor.putBoolean("NEW", false);

    myEditor.commit();

    Intent intent = new Intent(RegistrationActivity.this, MenuActivity.class);
    startActivity(intent);

    finish();
}
}

```

SettingsActivity.java:

```

package com.example.smartlock;

import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

public class SettingsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_settings);

        SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(SettingsActivity.this);

        TextView nm = (TextView) findViewById(R.id.editText13);
        TextView sn = (TextView) findViewById(R.id.editText14);
        TextView ad = (TextView) findViewById(R.id.editText11);
        TextView nu = (TextView) findViewById(R.id.editText12);
    }
}

```

```

    TextView dn = (TextView) findViewById(R.id.textView7);

    nm.setText(myPreferences.getString("NAME", ""));

    sn.setText(myPreferences.getString("SYENAME", ""));

    ad.setText(myPreferences.getString("ADDRESS", ""));

    nu.setText(myPreferences.getString("NUMBER", ""));

    dn.setText(myPreferences.getString("DNUMBER", ""));

}

public void savechanges(android.view.View view) {

    TextView nm = (TextView) findViewById(R.id.editText13);

    TextView sn = (TextView) findViewById(R.id.editText14);

    TextView ad = (TextView) findViewById(R.id.editText11);

    TextView nu = (TextView) findViewById(R.id.editText12);

    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(SettingsActivity.this);

    SharedPreferences.Editor myEditor = myPreferences.edit();

    myEditor.putString("NAME", nm.getText().toString());

    myEditor.putString("SYENAME", sn.getText().toString() );

    myEditor.putString("ADDRESS", ad.getText().toString() );

    myEditor.putString("NUMBER", nu.getText().toString() );

    myEditor.commit();

    Toast.makeText(getApplicationContext(), "Changes saved!", Toast.LENGTH_SHORT).show();

}

public void reset(android.view.View view) {

    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(SettingsActivity.this);

    SharedPreferences.Editor myEditor = myPreferences.edit();

    myEditor.putString("NAME", "");

    myEditor.putString("SYENAME", "" );

    myEditor.putString("ADDRESS", "" );

    myEditor.putString("NUMBER", "" );

    myEditor.putString("DNUMBER", "");

    myEditor.putString("CODE", "");

    myEditor.putBoolean("NEW", true);

    myEditor.commit();

    Toast.makeText(getApplicationContext(), "Reset complete!", Toast.LENGTH_SHORT).show();

    Intent intent = new Intent(SettingsActivity.this, MainActivity.class);

    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

    startActivity(intent);

    finish();

}

```



```
}
```

SMSReciever.java:

```
package com.example.smartlock;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Vibrator;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        if (intent.getAction() == "android.provider.Telephony.SMS_RECEIVED") {

            long mills = 3000L;

            Vibrator vibrator = (Vibrator) context.getSystemService(Context.VIBRATOR_SERVICE);

            if (vibrator.hasVibrator()) {

                vibrator.vibrate(mills);

            }

            abortBroadcast();

        }

    }

}
```

SMSService.java:

```
package com.example.smartlock;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

public class SMSService extends Service {

    public SMSService() {

    }

}
```

```
public void onCreate() {  
    super.onCreate();  
}  
  
public int onStartCommand(Intent intent, int flags, int startId) {  
  
    someTask();  
    return super.onStartCommand(intent, flags, startId);  
}  
  
public void onDestroy() {  
    super.onDestroy();  
}  
  
public IBinder onBind(Intent intent) {  
    return null;  
}  
  
void someTask() {  
}  
}
```

StartRegActivity.java:

```
package com.example.smartlock;  
  
import android.content.Intent;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class StartRegActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_start_reg);  
    }  
  
    public void startregistration(android.view.View view)
```

```
{  
  
    Intent intent = new Intent(StartRegActivity.this, RegistrationActivity.class);  
  
    startActivity(intent);  
  
    finish();  
  
}  
}
```

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

[illegible]

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				