

Проектирование аппаратно-программного комплекса для безопасной передачи сообщения по оптическому каналу

Молганов А.А., И-19-2Р

Савельева В.В., ассоц.профессор, PhD

Факультет Информационных Технологий

Евразийский Технологический Университет

Алматы, 2023

Анализ предметной области и сферы применения

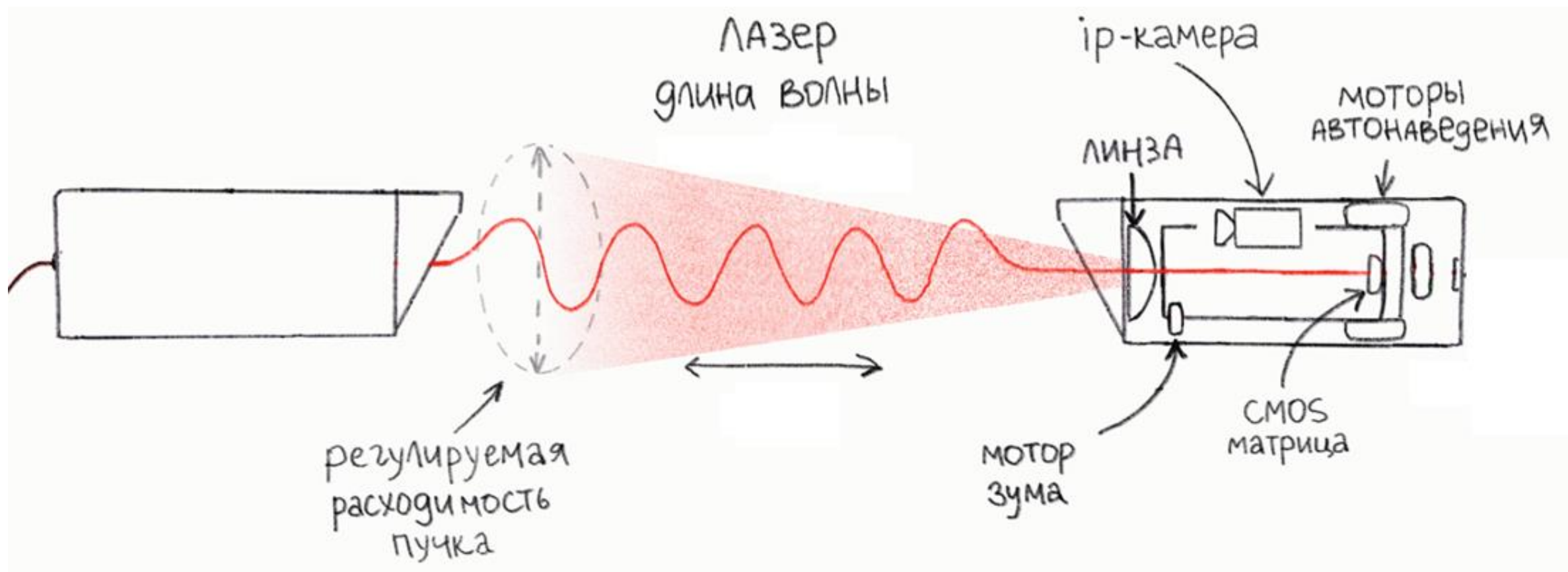
Анализ предметной области

Анализ предметной области сфокусирован на текущей ситуации телекоммуникационного рынка в области оборудования предназначенного для передачи и приема информации в открытом пространстве, иначе **FSO – Free-space optics**.

В качестве трансивера используется лазерный диод с длиной волны 940 нм, максимальным постоянным током не более 100 мА и импульсным током не более 1,5 А, мощность излучения до 40 мВт и углом горизонтальной развертки не более 34 градусов.



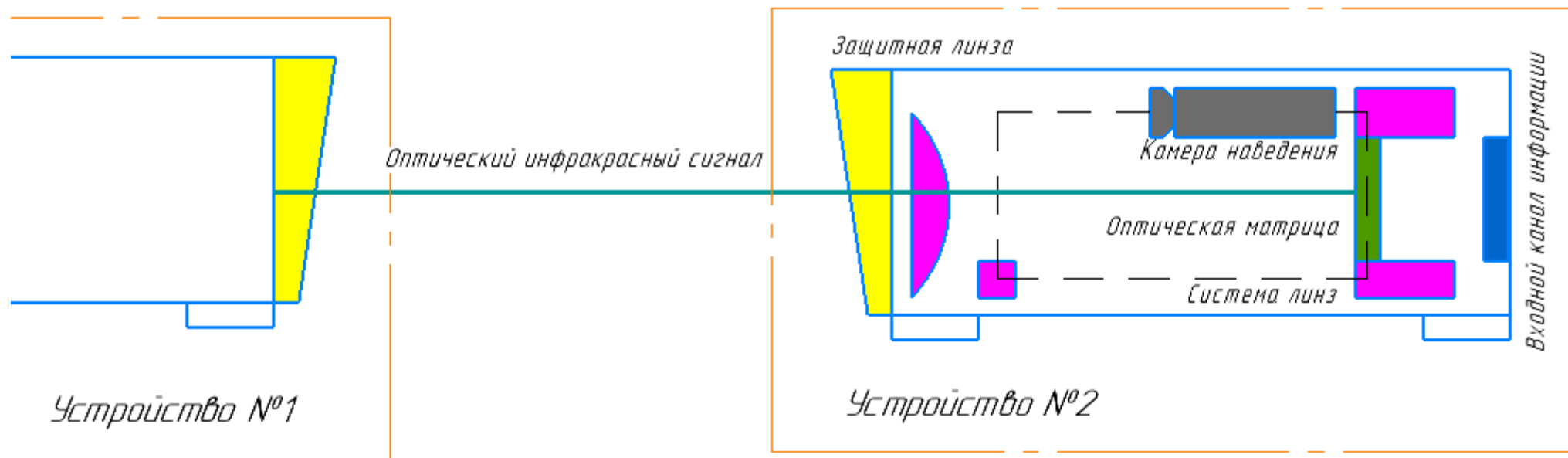
Анализ предметной области



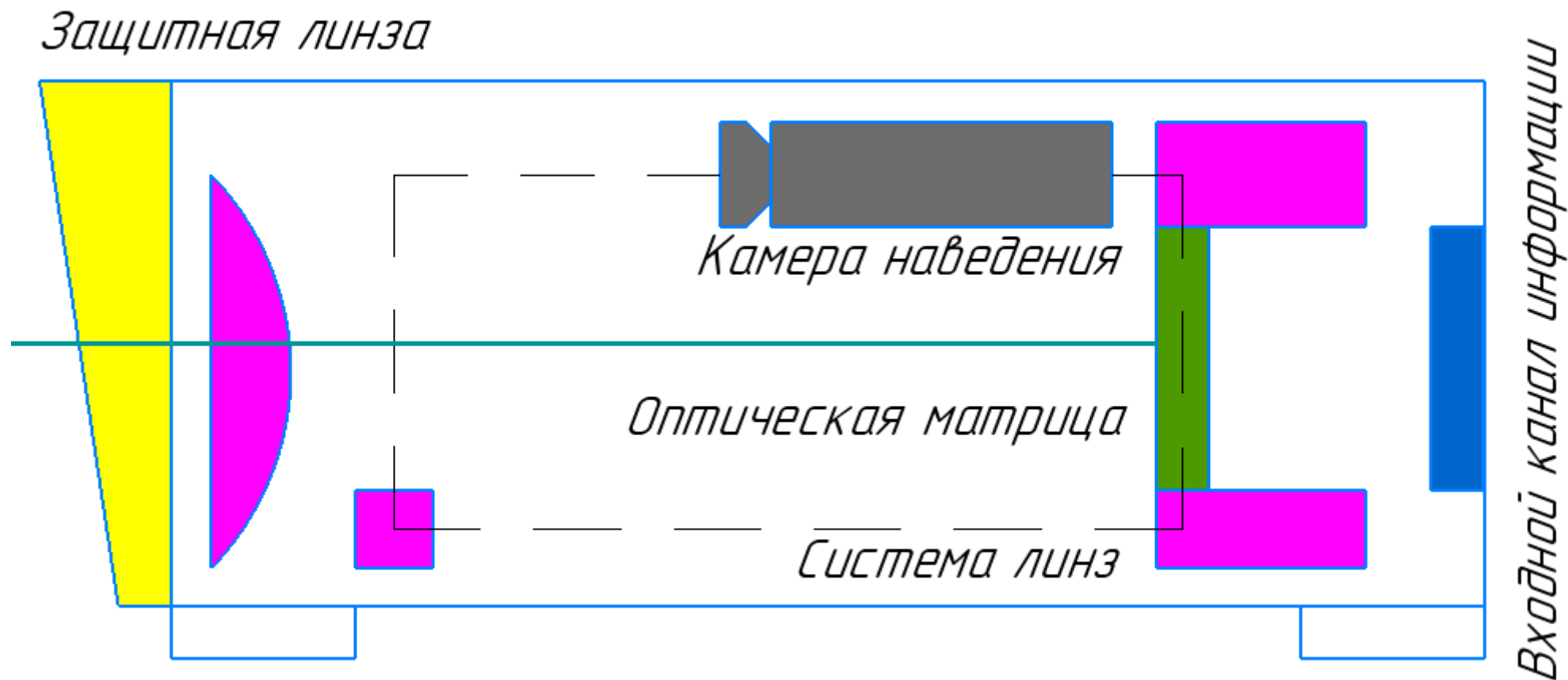


Проектирование аппаратно-программного комплекса

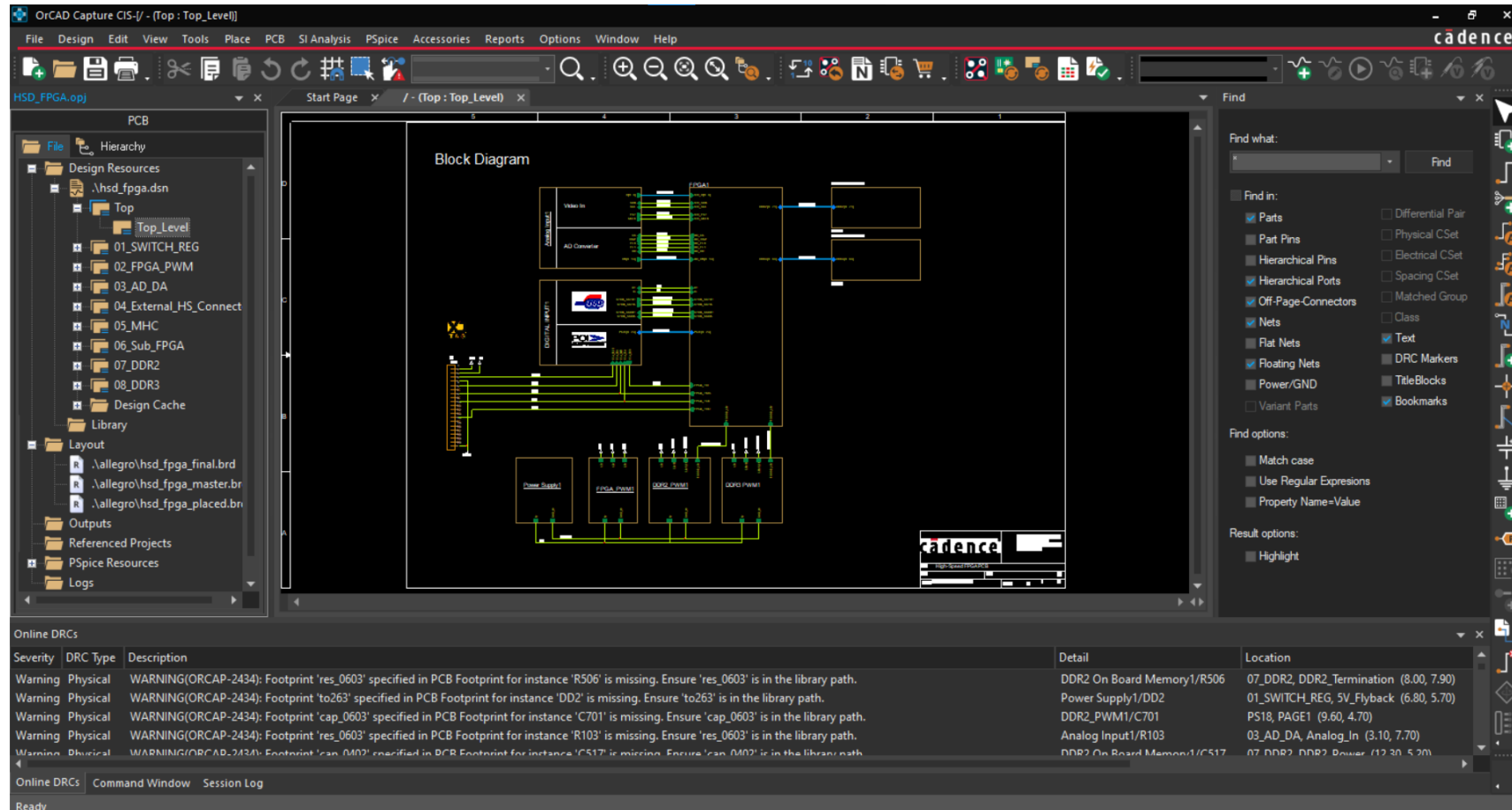
Проектирование аппаратного обеспечения комплекса



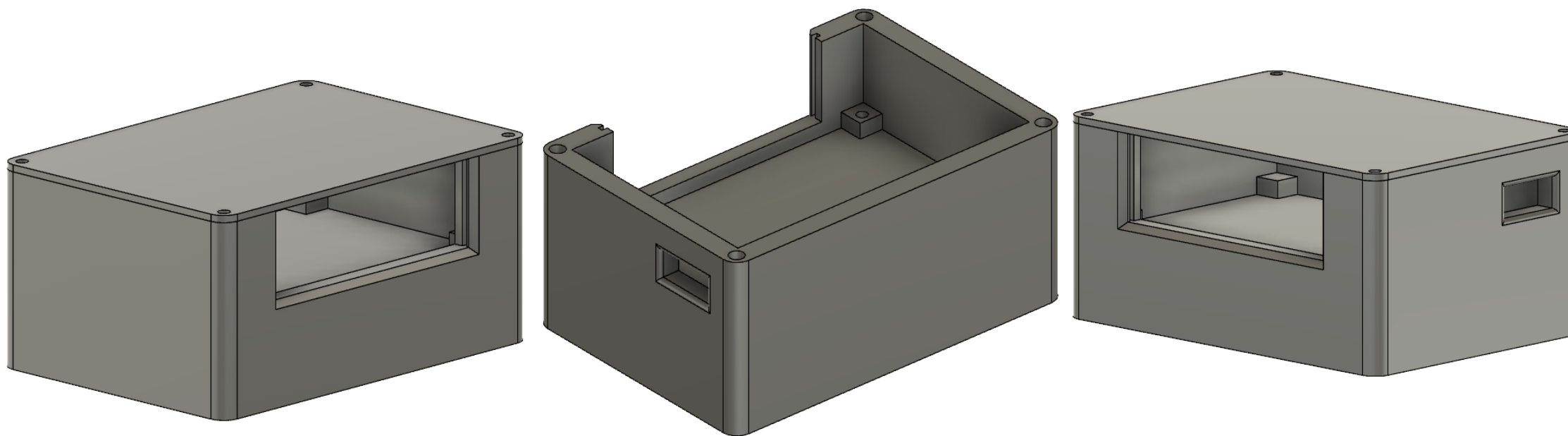
Проектирование аппаратного обеспечения комплекса



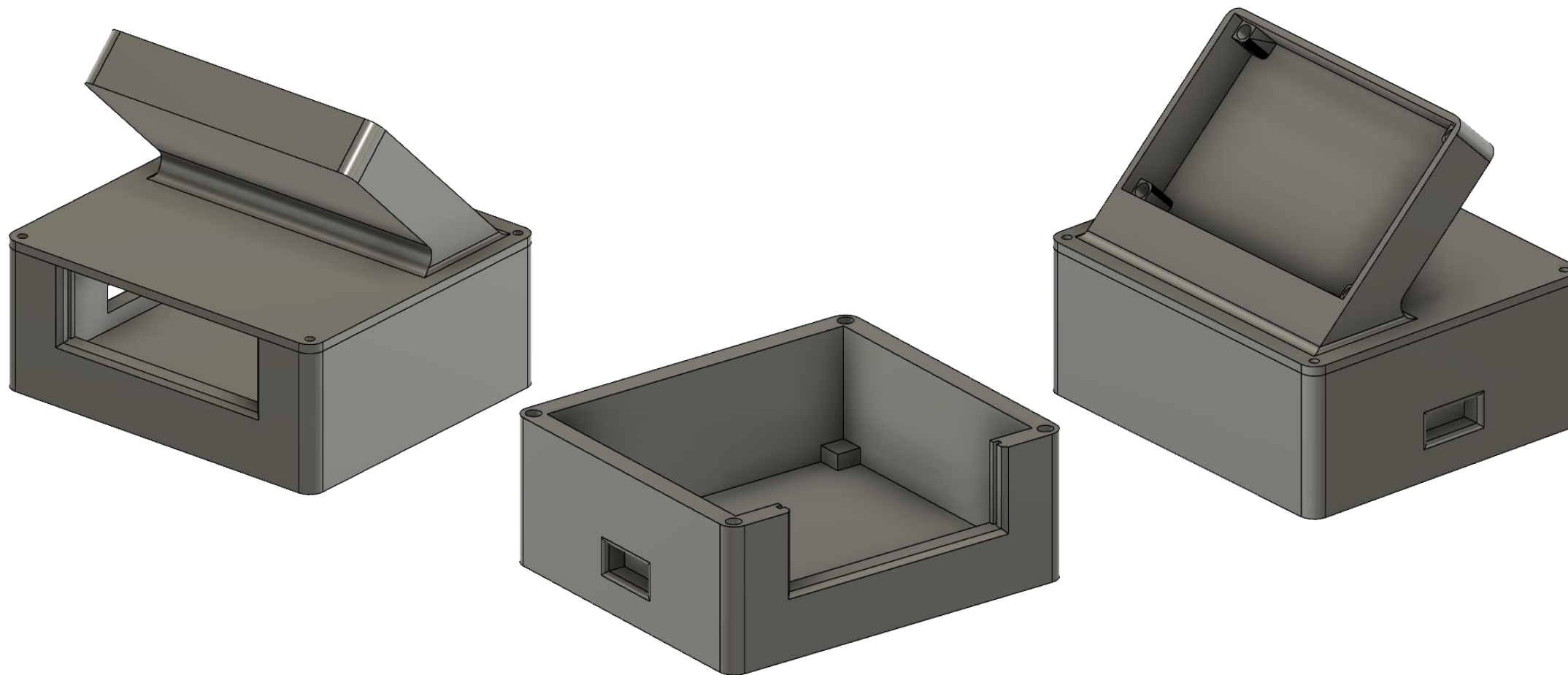
Проектирование аппаратного обеспечения комплекса



Проектирование аппаратного обеспечения комплекса



Проектирование аппаратного обеспечения комплекса



src - Device Configuration Tool - STM32CubeIDE

FileEditNavigateSearchProjectRunWindowHelp

Project Explorer

LAZER_TRANSFER

Binaries

Includes

Core

Drivers

Debug

LAZER_TRANSFER.ioc

LAZER_TRANSFER Debug

STM32H750VBTX_FLASH

STM32H750VBTX_RAM.Ic

SOUND_TRANSFER

Binaries

Includes

Core

Drivers

Debug

SOUND_TRANSFER.ioc

SOUND_TRANSFER Debug

STM32F411CEUX_FLASH

STM32F411CEUX_RAM.Ic

LAZER_TRANSFER.ioc

LAZER_TRANSFER.ioc - Pinout & Configuration

Pinout & Configuration

Clock Configuration

Project Manager

Tools

Software Packs

Pinout

CRYP Mode and Configuration

Mode

Activated

Configuration

Reset Configuration

NVIC Settings

DMA Settings

Parameter Settings

User Constants

Configure the below parameters :

Search (Ctrl+F)

Algorithm

Data encryption algorithm

AES ECB

Parameters

Data type

32b(no swapping)

Key size

128b

Encryption/Decryption key

00000000 00000000 00000000 000...

Pinout view

System view

STM32H750VBTx

LQFP100

CRYP Mode and Configuration

Mode

☒ Activated

Configuration

Reset Configuration

☒ NVIC Settings

☒ DMA Settings

☒ Parameter Settings

☒ User Constants

Configure the below parameters :





▼ Algorithm

Data encryption algorithm AES ECB

▼ Parameters

Data type 32b(no swapping)

Key size 128b

Encryption/Decryption key 00000000 00000000 00000000 000...

Проектирование программного обеспечения комплекса

The screenshot shows a GitHub repository interface for 'ProjectOptics' by user 'InnovusSollertia'. The repository is public. The left sidebar shows the file tree with folders like 'docs', 'src', '.metadata', 'LAZER_TRANSFER', 'SOUND_TRANSFER', and 'STM32_AES', along with 'LICENSE' and 'README.md'. The main area displays the 'src' directory contents, including a table of files and their commit history.

Repository Information:

- Repository: ProjectOptics / src
- User: InnovusSollertia
- Status: Public
- Actions: Pin, Unwatch (1), Fork (0), Star (0)

Navigation:

- Code (selected)
- Issues
- Pull requests
- Actions
- Projects
- Wiki
- Security
- Insights
- Settings

File Tree (Left Sidebar):

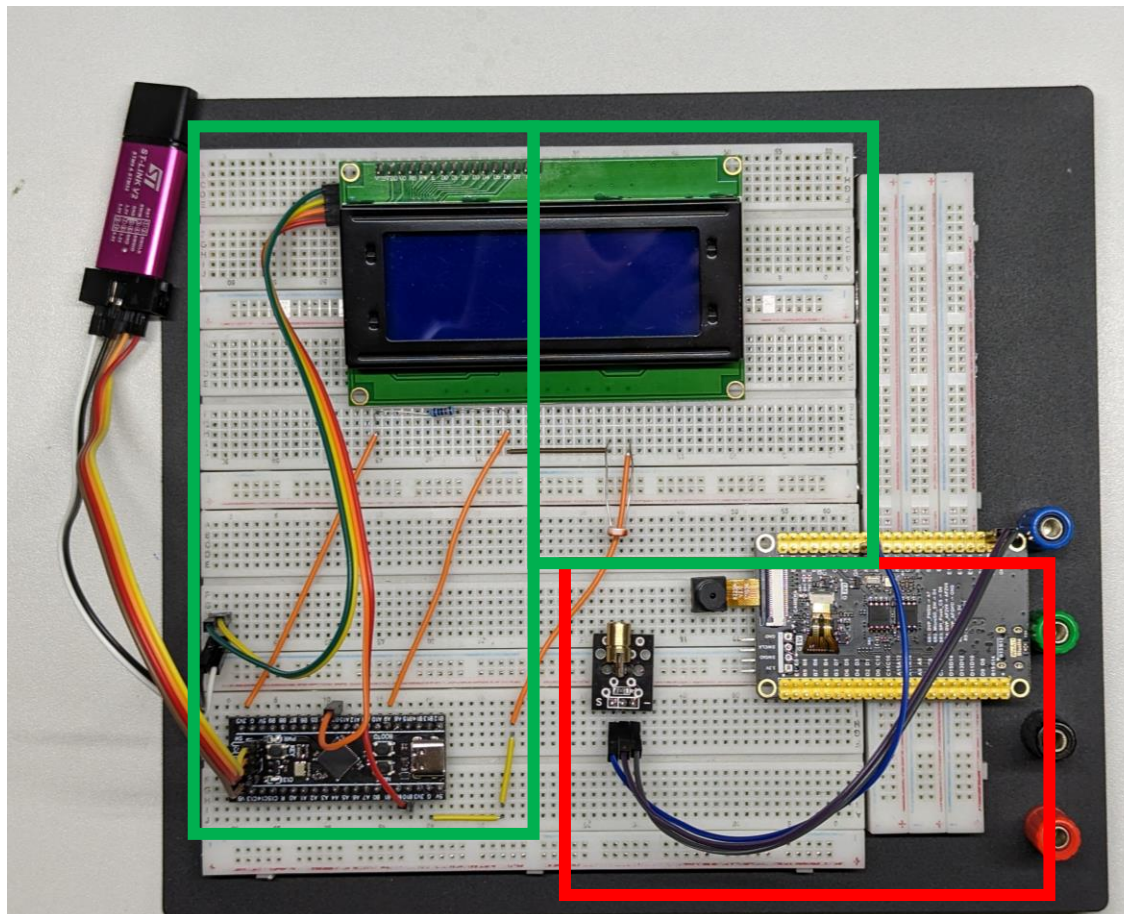
- docs
- src (selected)
- .metadata
- LAZER_TRANSFER
- SOUND_TRANSFER
- STM32_AES
- LICENSE
- README.md

src Directory Contents (Main Area):

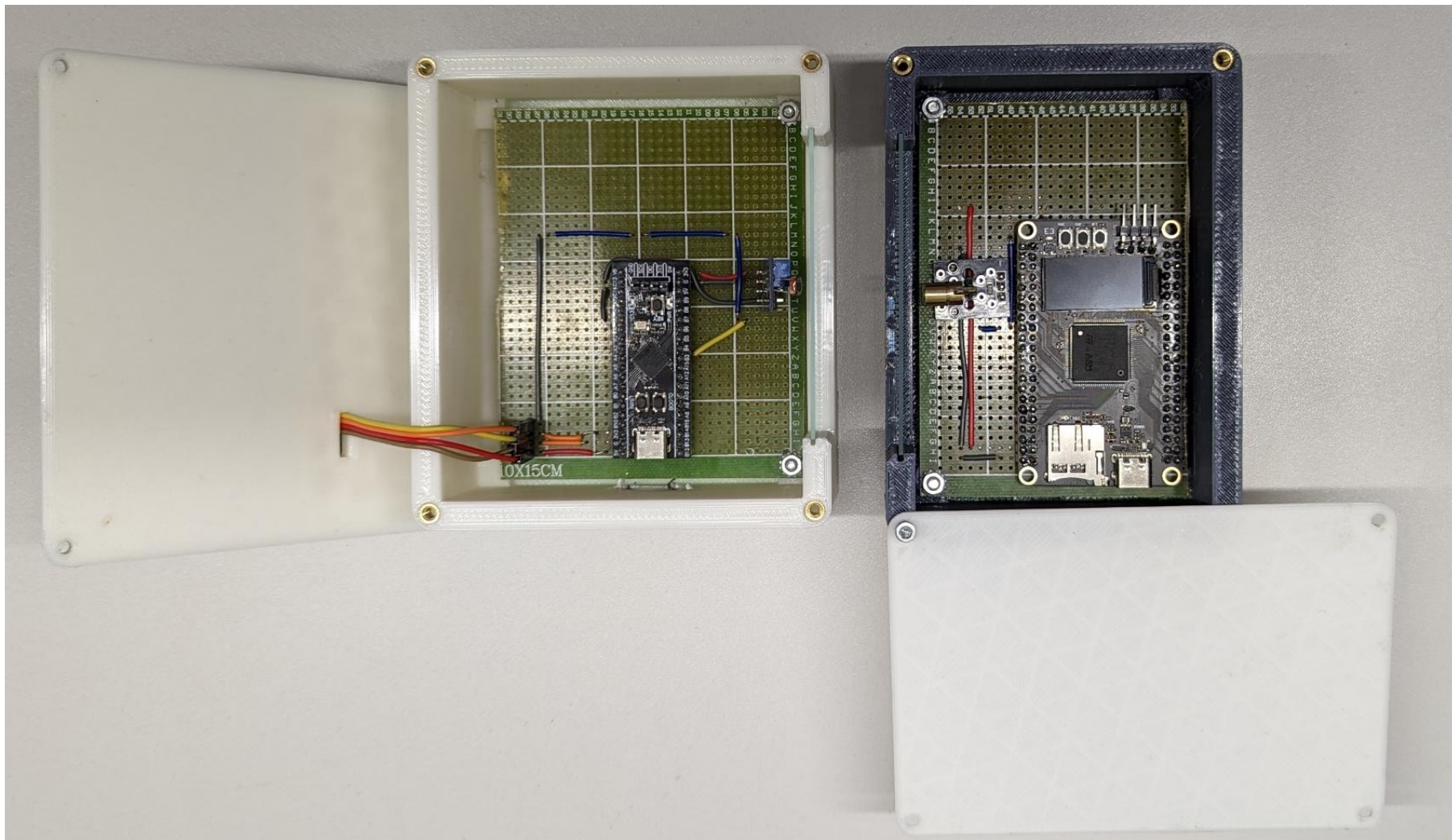
Name	Last commit message	Last commit date
..		
.metadata	20230123_145320	4 months ago
LAZER_TRANSFER	20230413_082240	last month
SOUND_TRANSFER	20230413_082240	last month
STM32_AES	20230413_082240	last month

Сборка тестового стенда

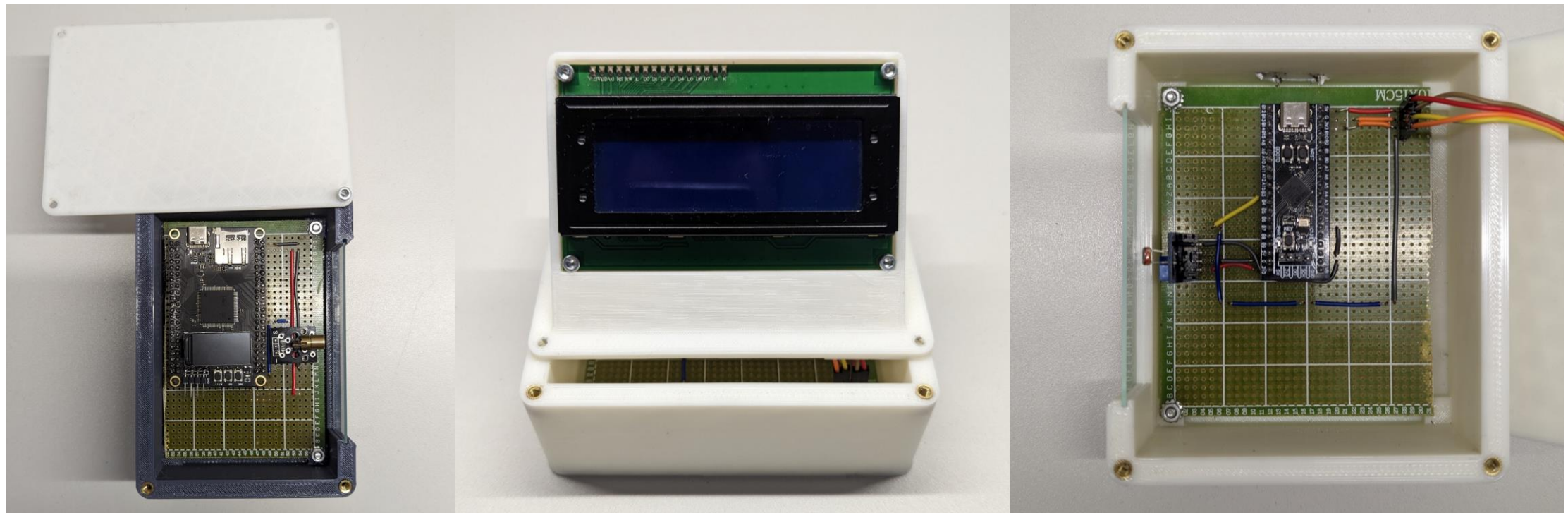
Сборка тестового стенда



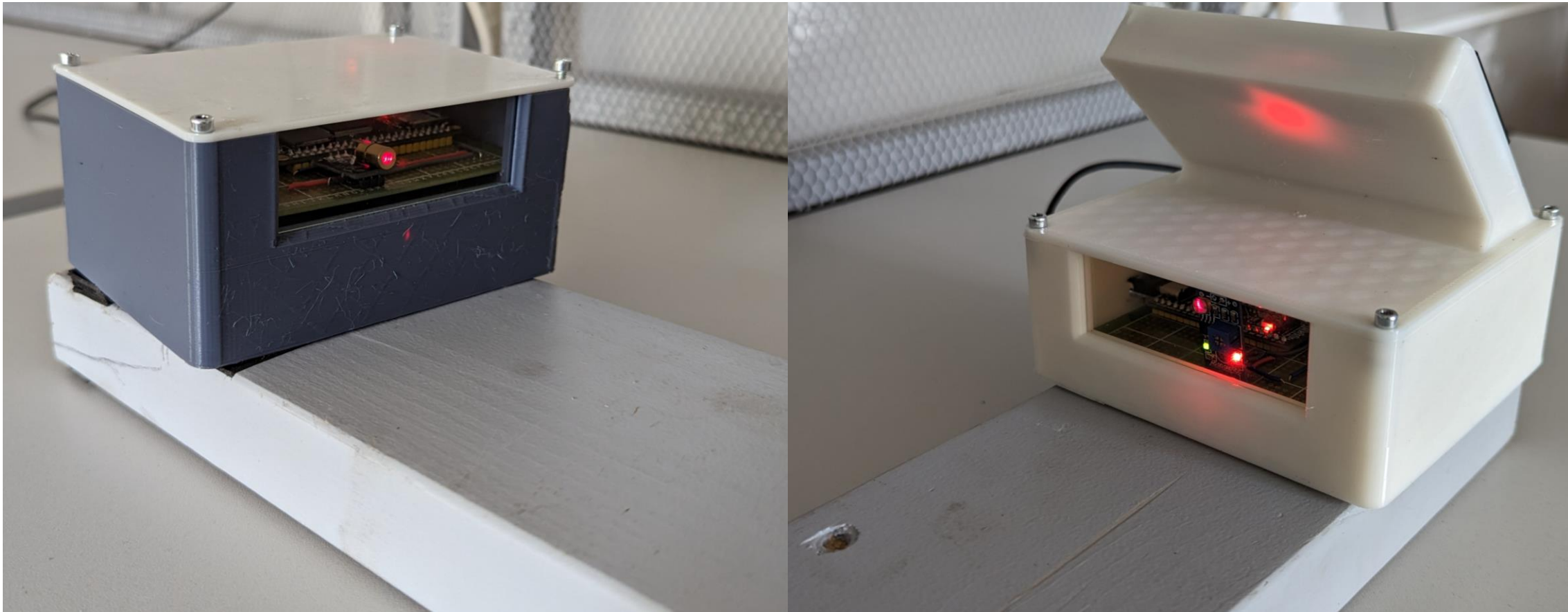
Сборка испытательного стенда



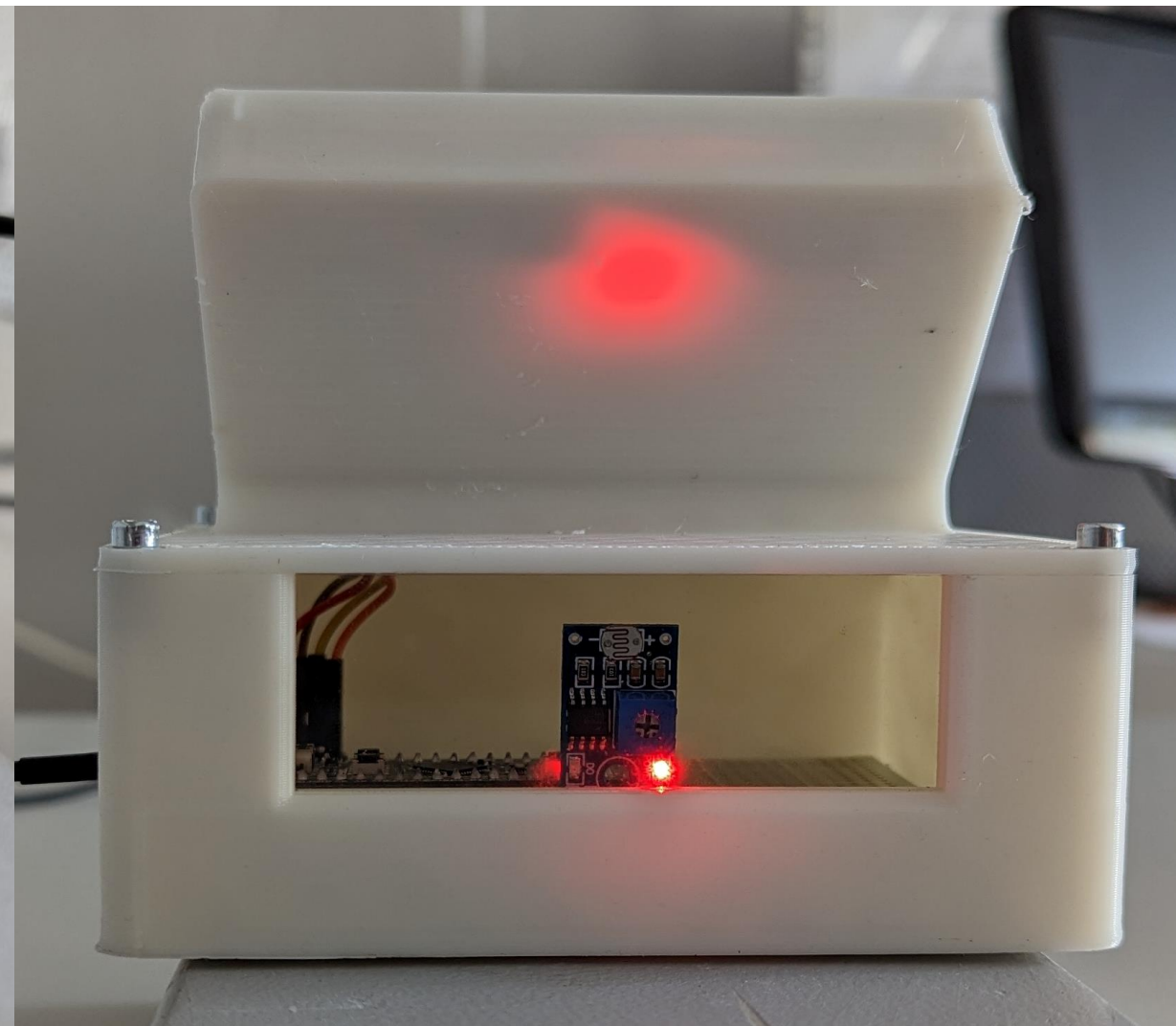
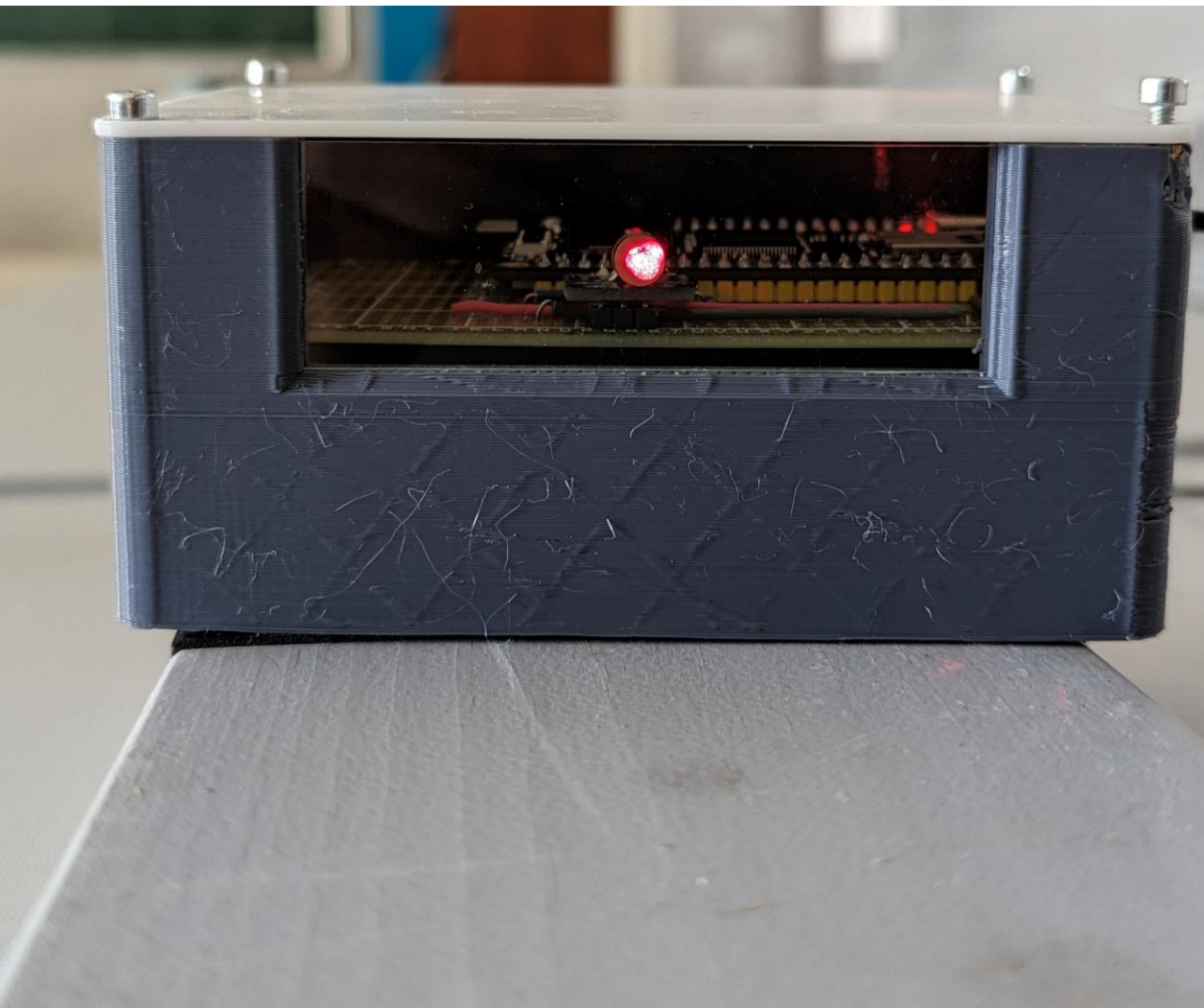
Сборка испытательного стенда



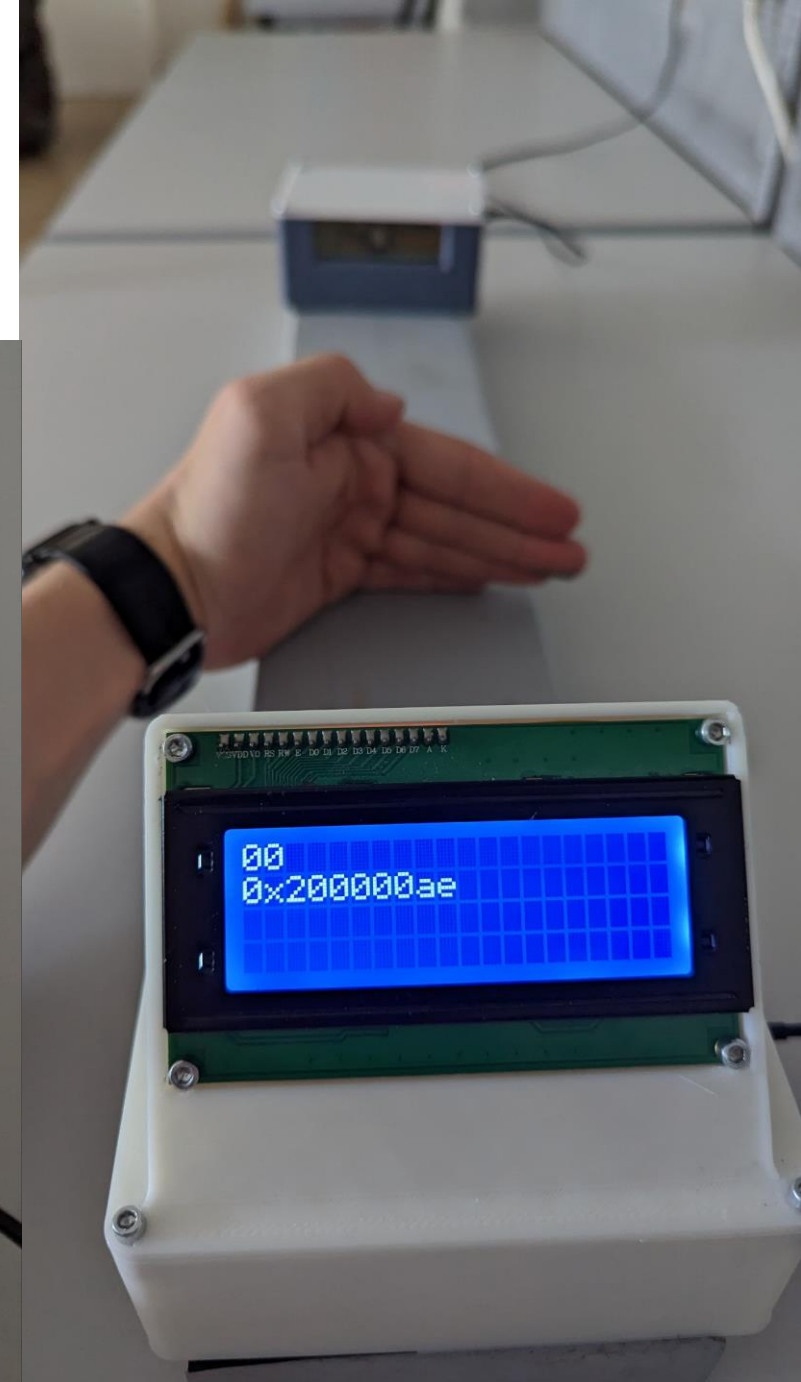
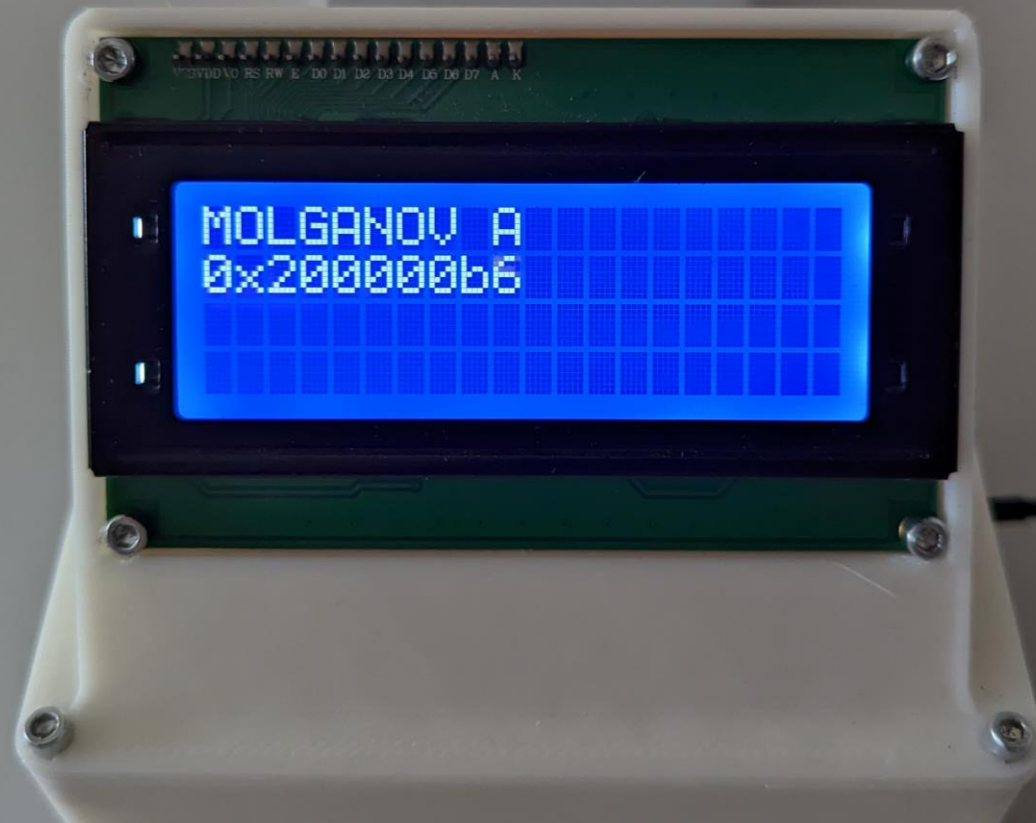
Отладка испытательного стенда



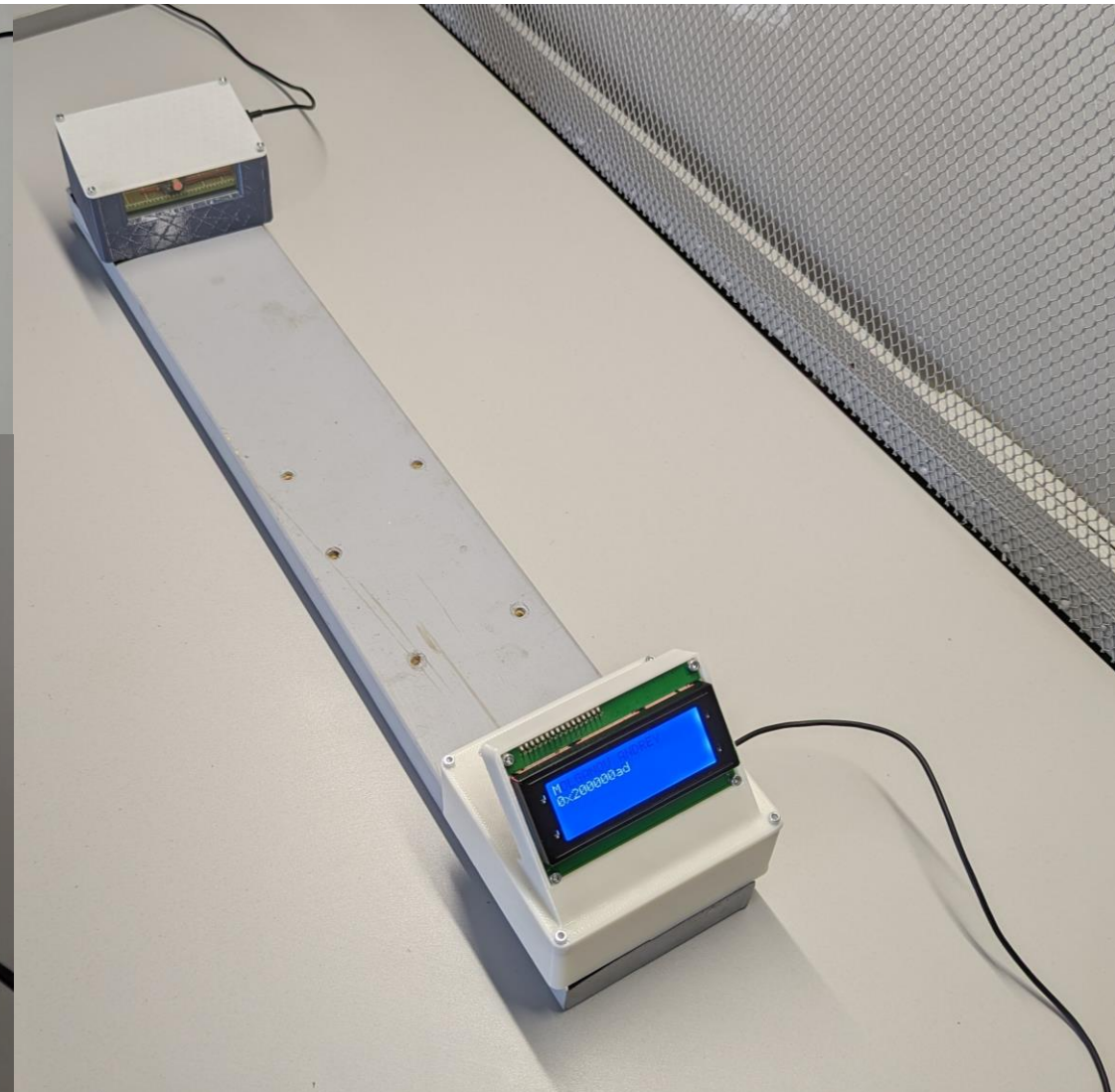
Отладка испытательного стенда



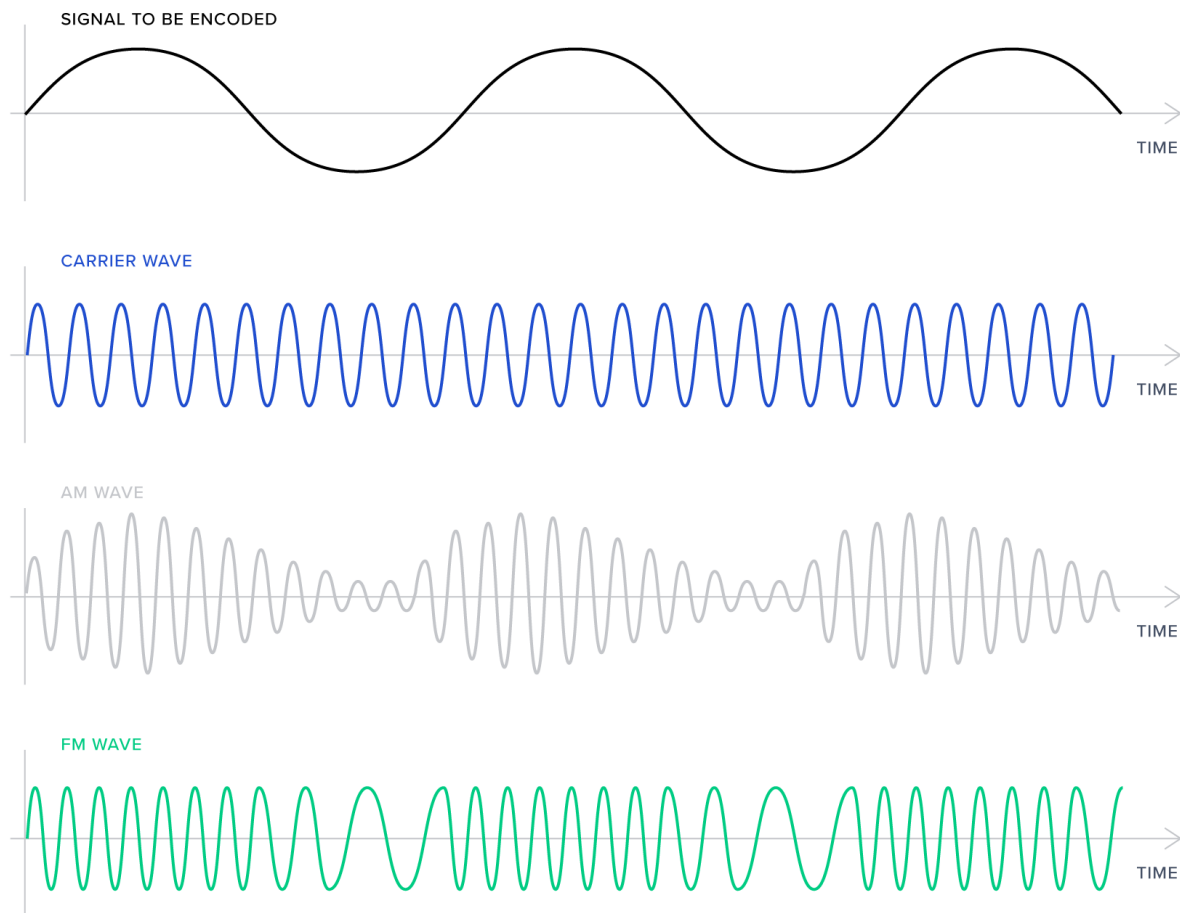
Отладка испытательного стенда



Отладка испытательного стенда



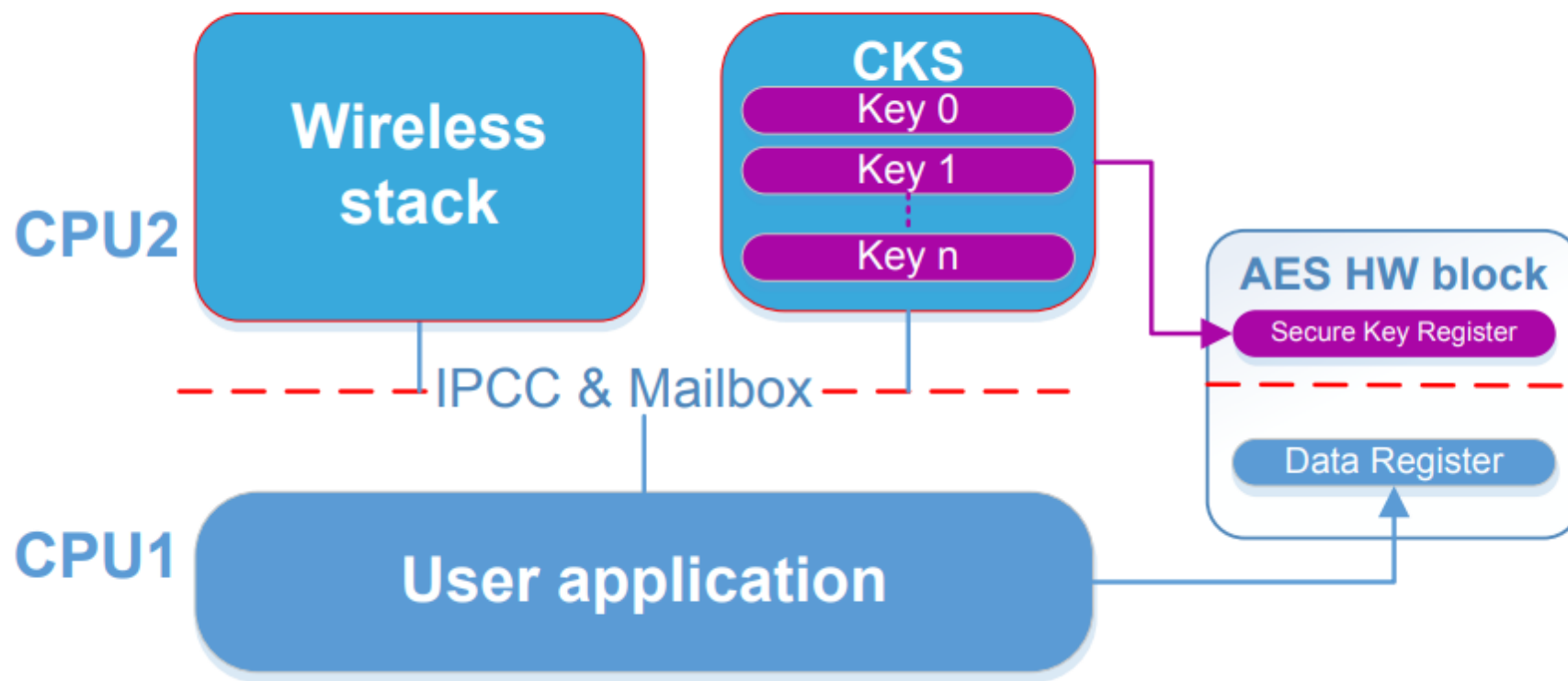
Отладка испытательного стенда



Оптимизация криптографического алгоритма

Оптимизация криптографического алгоритма AES-256

Figure 13. Dual-core architecture with CKS service



Оптимизация криптографического алгоритма AES-256

```
// выключаем прерывания
__disable_irq();

HAL_NVIC_SetPriority(TIM4_IRQn, 2, 0);
HAL_NVIC_EnableIRQ(TIM4_IRQn);
__HAL_TIM_ENABLE_IT(&tim4, TIM_IT_UPDATE);

// включаем таймеры

__HAL_TIM_ENABLE(&tim4);

// включаем прерывания
__enable_irq();

HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);

unsigned char openTextInput[16] = "abracadabra1234";
unsigned char openTextOutput[16] = { 0 };
unsigned char encryptedText[16] = { 0 };
unsigned char key[32] = "The quick brown fox jumps over";
```

Оптимизация криптографического алгоритма AES-256

```
// AES 256 *****

for(uint32_t i = 0; i < sizeof(encryptedText); encryptedText[i++] = 0);

start = millis;
Encrypt(openTextInput, key, encryptedText, 256);
timMeasures[measures::AES256_ENCRYPT] = millis - start;

for(uint32_t i = 0; i < sizeof(openTextOutput); openTextOutput[i++] = 0);

start = millis;
Decrypt(encryptedText, key, openTextOutput, 256);
timMeasures[measures::AES256_DECRYPT] = millis - start;
```

Оптимизация криптографического алгоритма AES-256

```
#define blockSize 4
// Макрос для нахождения произведения x ({02}) и аргумента по модулю {1b}
#define xtime(x) ((x<<1) ^ (((x>>7) & 1) * 0x1b))
// Макрос для умножения чисел в поле галуа (2^8)
#define Multiply(x,y) (((y & 1) * x) ^ ((y>>1 & 1) * xtime(x)) ^ ((y>>2 & 1) * xtime(xtime(x))) ^
#include "AES.h"

int rounds = 0;
int keyLength = 0;
//unsigned char plaintext[16];
//unsigned char encrypted[16];
//unsigned char state[4][4];
unsigned char roundKey[240];
unsigned char Key[32];
```


Оптимизация криптографического алгоритма AES-256

```
// Возвращает S-box значения
int get_SBox_Value(int num) {

    int sbox[256] = { 0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30,
        0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76, 0xca, 0x82, 0xc9, 0x7d,
        0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72,
        0xc0, 0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5,
        0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15, 0x04, 0xc7, 0x23, 0xc3, 0x18,
        0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
        0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6,
        0xb3, 0x29, 0xe3, 0x2f, 0x84, 0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc,
        0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf, 0xd0,
        0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f,
        0x50, 0x3c, 0x9f, 0xa8, 0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38,
        0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2, 0xcd, 0x0c,
        0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64,
        0x5d, 0x19, 0x73, 0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88,
        0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb, 0xe0, 0x32, 0x3a,
        0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95,
        0xe4, 0x79, 0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c,
        0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08, 0xba, 0x78, 0x25, 0x2e,
        0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
        0x8a, 0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35,
        0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e, 0xe1, 0xf8, 0x98, 0x11, 0x69,
        0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
        0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d,
        0x0f, 0xb0, 0x54, 0xbb, 0x16 };

    return sbox[num];
}
```

```
// Возвращает обратные S-box значения
int get_SBox_Inverse(int num) {

    int rsbox[256] = { 0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf,
        0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb, 0x7c, 0xe3, 0x39, 0x82,
        0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9,
        0xcb, 0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c,
        0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e, 0x08, 0x2e, 0xa1, 0x66, 0x28,
        0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
        0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c,
        0xcc, 0x5d, 0x65, 0xb6, 0x92, 0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed,
        0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84, 0x90,
        0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05,
        0xb8, 0xb3, 0x45, 0x06, 0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f,
        0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b, 0x3a, 0x91,
        0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0,
        0xb4, 0xe6, 0x73, 0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85,
        0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e, 0x47, 0xf1, 0x1a,
        0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18,
        0xbe, 0x1b, 0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a,
        0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4, 0x1f, 0xdd, 0xa8, 0x33,
        0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec,
        0x5f, 0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5,
        0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef, 0xa0, 0xe0, 0x3b, 0x4d, 0xae,
        0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
        0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14,
        0x63, 0x55, 0x21, 0x0c, 0x7d };

    return rsbox[num];
}
```

Оптимизация криптографического алгоритма AES-256

```
// Таблица поиска для раундового массива слов
// Содержит значения, полученные из x, путем возведения в степень (i-1) по модулю ({02}), в поле Галуа (2^8)
int Rcon[255] = { 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97,
0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4,
0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83,
0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc,
0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91,
0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33,
0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,
0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa,
0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25,
0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d,
0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4,
0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61,
0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74,
0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97,
0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4,
0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83,
0x1d, 0x3a, 0x74, 0xe8, 0xcb };
```


Оптимизация криптографического алгоритма AES-256

```
// Выделение раундовых ключей из основного ключа
void Expand_Keys() {
    int i, j;
    unsigned char temp[4], k;

    // Использовать основной ключ для первого раунда
    for (i = 0; i < keyLength; i++) {
        roundKey[i * 4] = Key[i * 4];
        roundKey[i * 4 + 1] = Key[i * 4 + 1];
        roundKey[i * 4 + 2] = Key[i * 4 + 2];
        roundKey[i * 4 + 3] = Key[i * 4 + 3];
    }

    // Каждый последующий раундовый ключ выводится из ранее полученных раундовых ключей
    while (i < (blockSize * (rounds + 1))) {
        for (j = 0; j < 4; j++) {
            temp[j] = roundKey[(i - 1) * 4 + j];
        }
    }
}
```


Оптимизация криптографического алгоритма AES-256

```
// Взять четырехбайтный ввод и применить S-box подстановку
{
    temp[0] = get_SBox_Value(temp[0]);
    temp[1] = get_SBox_Value(temp[1]);
    temp[2] = get_SBox_Value(temp[2]);
    temp[3] = get_SBox_Value(temp[3]);
}

temp[0] = temp[0] ^ Rcon[i / keyLength];
} else if (keyLength > 6 && i % keyLength == 4) {
{
    temp[0] = get_SBox_Value(temp[0]);
    temp[1] = get_SBox_Value(temp[1]);
    temp[2] = get_SBox_Value(temp[2]);
    temp[3] = get_SBox_Value(temp[3]);
}
}

roundKey[i * 4 + 0] = roundKey[(i - keyLength) * 4 + 0] ^ temp[0];
roundKey[i * 4 + 1] = roundKey[(i - keyLength) * 4 + 1] ^ temp[1];
roundKey[i * 4 + 2] = roundKey[(i - keyLength) * 4 + 2] ^ temp[2];
roundKey[i * 4 + 3] = roundKey[(i - keyLength) * 4 + 3] ^ temp[3];
i++;
}
```

Оптимизация криптографического алгоритма AES-256

```
// Добавить раундовый ключ к state с помощью XOR
void Add_Round_Key(int round, unsigned char state[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            state[j][i] ^= roundKey[round * blockSize * 4 + i * blockSize + j];
        }
    }
}

// Заменить значения матрицы состояний на соответственные значения S-box
void Sub_Bytes(unsigned char state[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            state[i][j] = get_SBox_Value(state[i][j]);
        }
    }
}

// То же что и в пред. функции, только с обратной S-Box
void Inv_Sub_Bytes(unsigned char state[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++) {
        for (j = 0; j < 4; j++) {
            state[i][j] = get_SBox_Inverse(state[i][j]);
        }
    }
}
```

```
// Сдвинуть строки в state влево на значение номера строки
void Shift_Rows(unsigned char state[4][4]) {
    unsigned char temp;

    // Первую строку на 1
    temp = state[1][0];
    state[1][0] = state[1][1];
    state[1][1] = state[1][2];
    state[1][2] = state[1][3];
    state[1][3] = temp;

    // Вторую строку на 2
    temp = state[2][0];
    state[2][0] = state[2][2];
    state[2][2] = temp;
    temp = state[2][1];
    state[2][1] = state[2][3];
    state[2][3] = temp;

    // Третью на 3
    temp = state[3][0];
    state[3][0] = state[3][3];
    state[3][3] = state[3][2];
    state[3][2] = state[3][1];
    state[3][1] = temp;
}
```

Оптимизация криптографического алгоритма AES-256

```
// То же что и в предыдущей функции но со сдвигом вправо
void Inv_Shift_Rows(unsigned char state[4][4]) {
    unsigned char temp;

    temp = state[1][3];
    state[1][3] = state[1][2];
    state[1][2] = state[1][1];
    state[1][1] = state[1][0];
    state[1][0] = temp;

    temp = state[2][0];
    state[2][0] = state[2][2];
    state[2][2] = temp;
    temp = state[2][1];
    state[2][1] = state[2][3];
    state[2][3] = temp;

    temp = state[3][0];
    state[3][0] = state[3][1];
    state[3][1] = state[3][2];
    state[3][2] = state[3][3];
    state[3][3] = temp;
}
```

```
// Перемешать столбцы в state
void Mix_Columns(unsigned char state[4][4]) {
    int i;
    unsigned char x1, x2, x3;
    for (i = 0; i < 4; i++) {
        x1 = state[0][i];
        x3 = state[0][i] ^ state[1][i] ^ state[2][i] ^ state[3][i];
        x2 = state[0][i] ^ state[1][i];
        x2 = xtime(x2);
        state[0][i] ^= x2 ^ x3;
        x2 = state[1][i] ^ state[2][i];
        x2 = xtime(x2);
        state[1][i] ^= x2 ^ x3;
        x2 = state[2][i] ^ state[3][i];
        x2 = xtime(x2);
        state[2][i] ^= x2 ^ x3;
        x2 = state[3][i] ^ x1;
        x2 = xtime(x2);
        state[3][i] ^= x2 ^ x3;
    }
}
```

Оптимизация криптографического алгоритма AES-256

```
void Encrypt(unsigned char plaintext[16], unsigned char Key[32], unsigned char encrypted[16], int rounds) {  
  
    int i, j, round = 0;  
    unsigned char state[4][4];  
  
    // rounds = keyLen;  
  
    // Вычислить значение длины ключа и количество раундов  
    keyLength = rounds / 32;  
    rounds = keyLength + 6;  
  
    // for (i = 0; i < keyLength * 4; i++) {  
    //     Key[i] = key[i];  
    // }  
  
    // for (i = 0; i < blockSize * 4; i++) {  
    //     plaintext[i] = text[i];  
    // }  
  
    // Сгенерировать раундовые ключи перед шифрованием  
    Expand_Keys();  
  
    // Копировать открытый текст в state массив  
    for (i = 0; i < 4; i++) {  
        for (j = 0; j < 4; j++) {  
            state[j][i] = plaintext[i * 4 + j];  
        }  
    }  
}
```

Оптимизация криптографического алгоритма AES-256

```
void Decrypt(unsigned char encrypted[16], unsigned char Key[32], unsigned char plaintext[16], int rounds)
```

```
    int i, j, round = 0;  
    unsigned char state[4][4];
```

```
    // rounds = keylen;
```

```
    // Вычислить значение длины ключа и количество раундов  
    keylength = rounds / 32;  
    rounds = keylength + 6;
```

```
    // for (i = 0; i < keylength * 4; i++) {  
    //     Key[i] = key[i];  
    // }
```

```
    // for (i = 0; i < blockSize * 4; i++) {  
    //     encrypted[i] = text[i];  
    // }
```

```
    Expand_Keys();
```

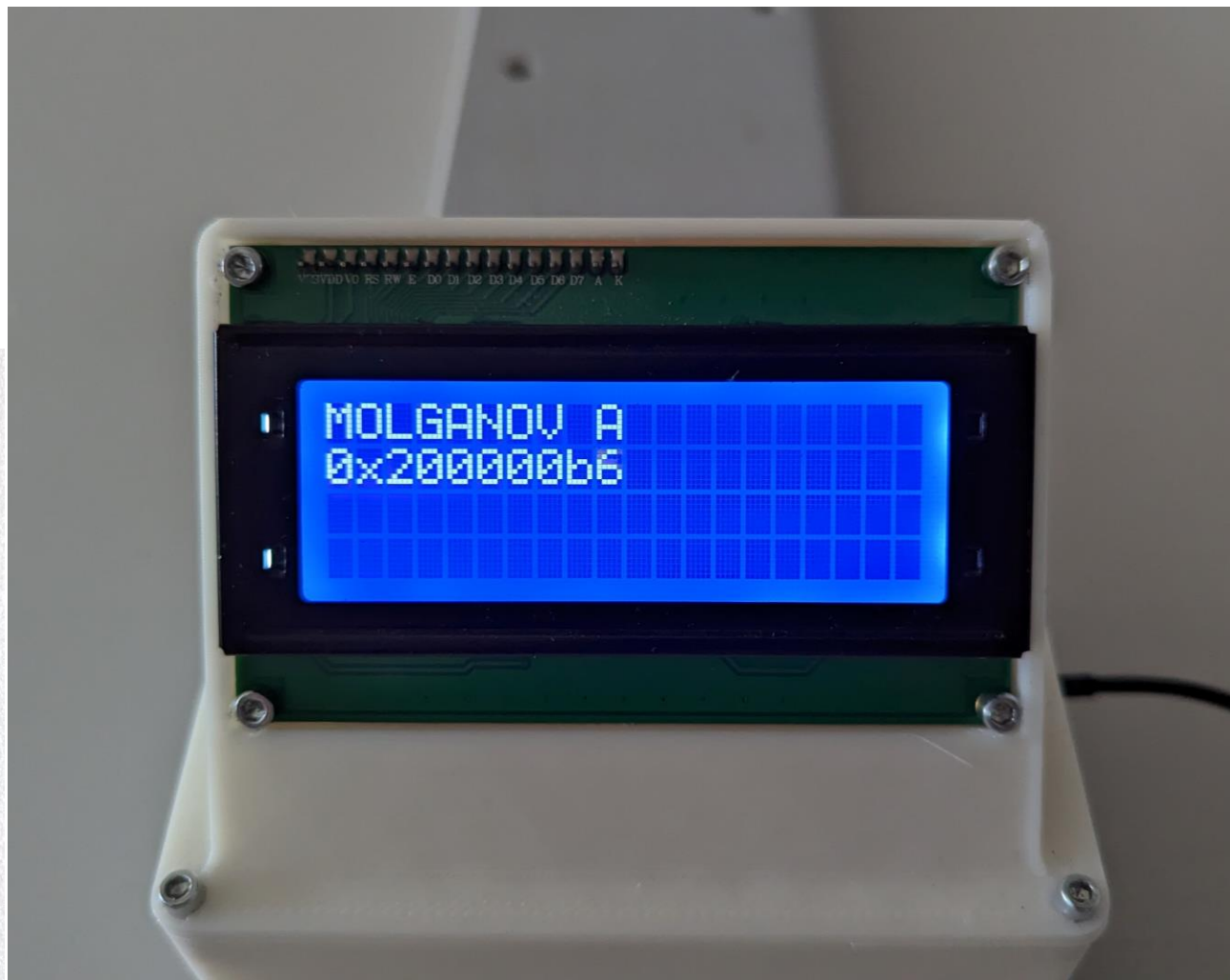
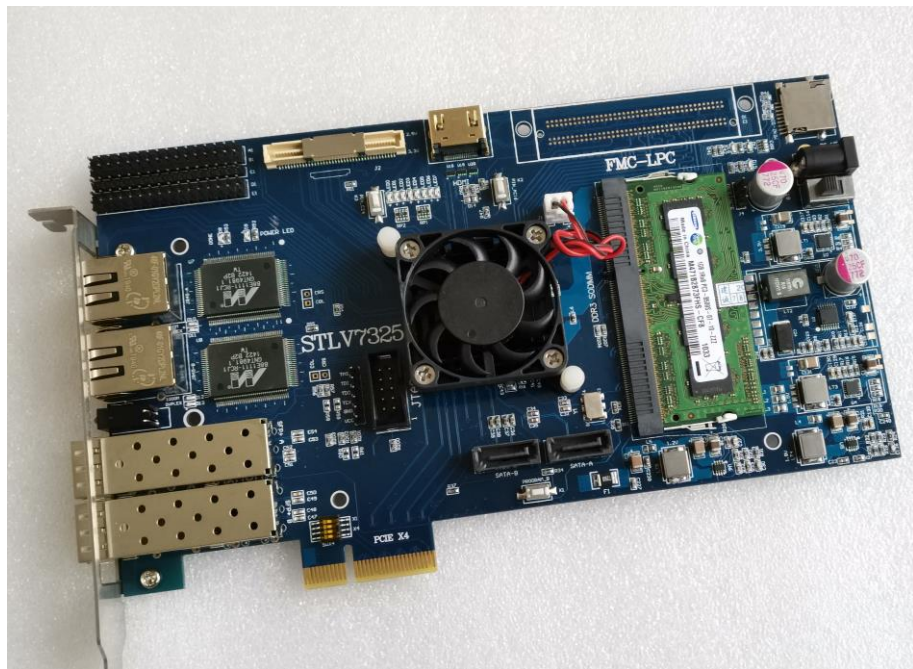
```
    // Копировать шифротекст в state
```

```
    for (i = 0; i < 4; i++) {  
        for (j = 0; j < 4; j++) {  
            state[j][i] = encrypted[i * 4 + j];  
        }  
    }
```

```
    // получаем открытый текст
```

```
    for (i = 0; i < 4; i++) {  
        for (j = 0; j < 4; j++) {  
            plaintext[i * 4 + j] = state[j][i];  
        }  
    }
```


Заклучение



Спасибо за внимание!