



Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Király Krisztina

ÖNÁLLÓ LABORATÓRIUM DOKUMENTÁCIÓ

KONZULENS: KOVÁCS VIKTOR

BUDAPEST, 2024

Az önálló laboratóriumi tárgy keretében egy STM32 mikrovezérlő alapú jelszókezelő hardver és az ehhez kapcsolódó kliens alkalmazás fejlesztésével foglalkoztam. A projekt célja egy olyan eszköz megvalósítása volt, amely a felhasználók jelszavait biztonságosan, titkosított formában tárolja az eszköz beépített flash memóriájában. További funkciója, hogy híd device-ként működjön, mellyel lehetővé teszi a felhasználó számára, hogy egyetlen gombnyomással az eszköz automatikusan begépelje a kiválasztott jelszót egy tetszőleges alkalmazásba vagy webes felületre. A hardver funkciók mellett fontos volt egy olyan kliens alkalmazás fejlesztése, amely felhasználóbaráttá teszi a jelszavak kezelését. Az alkalmazás segítségével a felhasználó új jelszavakat hozhat létre, meglévőket szerkesztheti vagy törölheti, illetve egyszerűen átláthatja a tárolt jelszavainak listáját. Ezzel az eszköz és a kliens alkalmazás együttesen biztosítja jelszavak biztonságos tárolását, kezelését és azok kényelmes felhasználását is.

A következőkben részletesen bemutatom a feladat megvalósításának módját, beleértve a hardver és szoftver komponensek tervezését és implementálását. Emellett kitérnék a fejlesztés során felmerült kihívásokra, valamint az ezekre adott megoldásokra is, amelyek meghatározó szerepet játszottak a projekt kifutásában.

A feladat a fentiek alapján három fő részre osztható:

- Hardver konfigurációja és a jelszavak hardver oldali kezelése – Az STM32 mikrovezérlő konfigurálása, a titkosításhoz használt algoritmusok megismerése és implementálása, valamint az adatok tárolásához és az automatikus jelszó-beíráshoz kapcsolódó funkciók megvalósítása.
- Kliens alkalmazás és felhasználói felület kialakítása – Felhasználóbarát szoftver létrehozása, amely lehetővé teszi felhasználók regisztrálását, bejelentkezését, valamint a jelszavak hozzáadását, törlését és szerkesztését.
- Hardver és kliens alkalmazás közötti kommunikáció – Stabil és biztonságos adatkapcsolat létrehozása az utasítások és az adatok kezeléséhez.

A fejlesztés az I-CUBE-USBD-Composite SDK megismerésével indult. Ez a könyvtár egy csomagoló osztályt biztosít, amely lehetővé teszi, hogy egy kapcsolattal egy eszköz egyszerre több USB device classként működjön. Az elérhető device classok között található például a Mass Storage (tömegtároló eszköz), mikrofon, kamera, egér, billentyűzet, valamint az ACM (Abstract Control Module), amely lehetőséget nyújt virtuális soros porton keresztüli kommunikációra a gazdarendszerrel.

A projekt megvalósítása során a billentyűzet és az ACM device classokat használtam. Ehhez először hozzá kellett adnom a könyvtárat az STM32CubeIDE fejlesztőkörnyezethez. Ezután az .ioc fájl middleware beállításai között engedélyeztem a szükséges osztályokat a library alatti menüben, így lehetővé téve azok használatát a projektben.

USBD_USE_CDC_ACM	<input checked="" type="checkbox"/>
USBD_CDC_ACM_COUNT	1
USBD_USE_CDC_RNDIS	<input type="checkbox"/>
USBD_USE_CDC_ECM	<input type="checkbox"/>
USBD_USE_HID_MOUSE	<input type="checkbox"/>
USBD_USE_HID_KEYBOARD	<input checked="" type="checkbox"/>

Ezt követően engedélyeztem USB_OTG_FS-t (USB On-The-Go Full Speed), amely a Connectivity szekció alatt található a device configuration tool nézetben. Ezáltal az eszköz már képes lesz usb-n keresztül csatlakozni a gazdarendszerrel. A konfigurációt követően a projekt main.c fájljában meg kellett hívni az MX_USB_Device_Init() függvényt a kommunikáció inicializálásához. Ez a függvény automatikusan elvégzi az USB interfész működéséhez szükséges paraméterek konfigurálását, például az eszközazonosítók, endpointok és protokollok beállítását.

A billentyűzet konfigurálásához definiált HID_KEYBOARD_ReportDesc tömb és a hozzá tartozó HID_KEYBOARD_REPORT_DESC_SIZE változó eredetileg nem tartalmazta a megfelelő beállításokat. Ez pedig nem várt hibát okozott az eszköz csatlakoztatásakor. Ugyanis ezek nélkül a Windows operációs rendszer nem tudta megfelelően felismerni az eszközt, ami nélkül pedig nem tudott megfelelően funkcionálni az eszköz. A hiba forrásának azonosítása és a megfelelő konfigurációk megtalálása után a két változó az alábbi szerkezetet kapta:

```
#define HID_KEYBOARD_REPORT_DESC_SIZE 63U

__ALIGN_BEGIN static uint8_t
HID_KEYBOARD_ReportDesc[HID_KEYBOARD_REPORT_DESC_SIZE] __ALIGN_END =
{
    0x05, 0x01, // USAGE_PAGE (Generic Desktop)
    0x09, 0x06, // USAGE (Keyboard)
    0xa1, 0x01, // COLLECTION (Application)
    0x05, 0x07, // USAGE_PAGE (Keyboard)
    0x19, 0xe0, // USAGE_MINIMUM (Keyboard LeftControl)
    0x29, 0xe7, // USAGE_MAXIMUM (Keyboard Right GUI)
    0x15, 0x00, // LOGICAL_MINIMUM (0)
    0x25, 0x01, // LOGICAL_MAXIMUM (1)
    0x75, 0x01, // REPORT_SIZE (1)
    0x95, 0x08, // REPORT_COUNT (8)
    0x81, 0x02, // INPUT (Data,Var,Abs)
    0x95, 0x01, // REPORT_COUNT (1)
    0x75, 0x08, // REPORT_SIZE (8)
```

```

0x81, 0x03, // INPUT (Cnst,Var,Abs)
0x95, 0x05, // REPORT_COUNT (5)
0x75, 0x01, // REPORT_SIZE (1)
0x05, 0x08, // USAGE_PAGE (LEDs)
0x19, 0x01, // USAGE_MINIMUM (Num Lock)
0x29, 0x05, // USAGE_MAXIMUM (Kana)
0x91, 0x02, // OUTPUT (Data,Var,Abs)
0x95, 0x01, // REPORT_COUNT (1)
0x75, 0x03, // REPORT_SIZE (3)
0x91, 0x03, // OUTPUT (Cnst,Var,Abs)
0x95, 0x06, // REPORT_COUNT (6)
0x75, 0x08, // REPORT_SIZE (8)
0x15, 0x00, // LOGICAL_MINIMUM (0)
0x25, 0x65, // LOGICAL_MAXIMUM (101)
0x05, 0x07, // USAGE_PAGE (Keyboard)
0x19, 0x00, // USAGE_MINIMUM (Reserved (no event indicated))
0x29, 0x65, // USAGE_MAXIMUM (Keyboard Application)
0x81, 0x00, // INPUT (Data,Ary,Abs)
0xc0 /* END_COLLECTION */
};

```

Miután megoldódott ez a probléma az string to key input konverter megírása következett. Ezek angol és magyar elosztáshoz is elkészítettem. És az alábbi módon néz ki:

```

int* convert_message_hun(wchar_t* message, int *t, int* is_special, int
length)
{
    for(int i=0; i<length; ++i)
    {
        is_special[i] = 0x00;
        switch (message[i])
        {
            case 'A': is_special[i] = 0x02;
            case 'a': t[i] = 0x04; break;
            case 'B': is_special[i] = 0x02;
            case 'b': t[i] = 0x05; break;
            // ... till
            case 'X': is_special[i] = 0x02;
            case 'x': t[i] = 0x1b; break;
            case 'Y': is_special[i] = 0x02;
            case 'y': t[i] = 0x1d; break;
            case 'Z': is_special[i] = 0x02;
            case 'z': t[i] = 0x1c; break;

            case L'Ö': is_special[i] = 0x20;
            case L'ö': t[i] = 0x27; break;
            case L'Ü': is_special[i] = 0x20;
            case L'ü': t[i] = 0x2d; break;
            case L'Ó': is_special[i] = 0x20;
            case L'ó': t[i] = 0x2f; break;
            case L'Ú': is_special[i] = 0x20;
            case L'ú': t[i] = 0x30; break;
            case L'É': is_special[i] = 0x20;
            case L'é': t[i] = 0x33; break;

```

```

        case L'Á': is_special[i] = 0x20;
        case L'á': t[i] = 0x34; break;
        case L'Ű': is_special[i] = 0x20;
        case L'ű': t[i] = 0x31; break;
        case L'í': is_special[i] = 0x20;
        case L'í': t[i] = 0x64; break;

        case '\n': t[i] = 0x28; break;
        case ' ': t[i] = 0x2c; break;
        case ',': t[i] = 0x36; break;
        case '.': t[i] = 0x37; break;
        case '-': t[i] = 0x38; break;
        case '/': t[i] = 0x54; break;
        case '*': t[i] = 0x55; break;
        case '+': t[i] = 0x57; break;
        case '?': is_special[i] = 0x20; t[i] = 0x36; break;
        case '!': is_special[i] = 0x20; t[i] = 0x21; break;
        case ':': is_special[i] = 0x20; t[i] = 0x37; break;
        case '\t': is_special[i] = 0x00; t[i] = 0x2b; break;
        case 127: is_special[i] = 0x00; t[i] = 0x2a; break;
        case '@': is_special[i] = 0x40; t[i] = 0x19; break;

        case '0': t[i] = 0x35; break;
        case '1': t[i] = 0x1e; break;
        case '2': t[i] = 0x1f; break;
        case '3': t[i] = 0x20; break;
        case '4': t[i] = 0x21; break;
        case '5': t[i] = 0x22; break;
        case '6': t[i] = 0x23; break;
        case '7': t[i] = 0x24; break;
        case '8': t[i] = 0x25; break;
        case '9': t[i] = 0x26; break;

        default: t[i] = 0x00; break;
    }
}

t[length] = 0x00;
is_special[length]= 0x00;
}

```

A függvény működése a következő: A message változóban (A message változó különös módon nem egy egyszerű char, hanem wchar_t. Erre azért van szükség, hogy a változóba elférjenek a magyar ékezetes karakterek is.) található az üzenet, ami átkonvertálásra vár, a length pedig ennek az üzenetnek a hossza. A függvény végig meg a message összes karakterén és megkeresi, hogy az a melyik esetnek felel meg a switch-en belül. Majd a t kimeneti változóba elmenti, hogy melyik billentyűzet leütés felel meg az adott karakternek. Az is_special változó pedig arra szolgál, hogy ha az adott karakter megjelenítéséhez szükséges valamilyen segéd gomb megnyomása például a nagy betűk megjelenítéséhez a shift billentyű egy idejű lenyomása,

akkor ez is elmentésre kerüljön. Miután a teljes üzenet átkonvertálásra került a `t` és `is_special` változókba, elküldésre kerülhetnek a `send_hid` függvénnyel.

```
void send_hid(wchar_t* message, int length)
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_set);

    int t[length+1];
    int is_special[length+1];
    wchar_t* language;

    if(lang == NULL)
        language = HUN;
    else
        language = lang;

    int res = wcscmp(language, HUN);
    if(res == 0)
        convert_message_hun(message, t, is_special, length);
    else
        convert_message_eng(message, t, is_special, length);
    for(int i=0; i<length+1;++i)
    {
        if(i!=0 && t[i-1]=='\n')
            USBD_Delay(50);

        report[2] = t[i];
        report[0] = is_special[i];

        USBD_HID_Keybaord_SendReport(&hUsbDevice, report, len);
        USBD_Delay(150);
    }

    report[0] = 0x00;
    USBD_HID_Keybaord_SendReport(&hUsbDevice, report, len);

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_reset);
}
```

A függvény működése a következő: Létrehozok 2 tömböt, amiben tárolni fogom a konvertálás eredményeit, majd a korábban elmentett nyelv változó (ezt a paramétert a kliens alkalmazás fogja elküldeni az eszköznek) alapján meghívom az angol vagy magyar kiosztású billentyűzethez tervezett konvertert. Miután elmentésre került az elküldendő karakterek végig megyek a tömbökön és beállítom a `uint8_t report[8]` megfelelő bájtjait a leütendő karakterekre és meghívom az SDK küldés függvényét. (Mivel ez a függvény addig írja a beadott karaktert amíg nem kap másikat, így az üzenet lezárásaként el kell küldeni egy üres karaktert is.)

Miután a hid kapcsolat felált áttértem a soros kommunikáció konfigurálására. Az `library` az `usb` kommunikációt úgy állította össze, hogy míg a küldést az eszköznek kell indítania a `CDC_Transmit` hívásával addig az olvasásnál csak kezelnie kell a megkapott adatokat. Ehhez

pedig mindössze az `usbd_cdc_acm_if.c` ben lévő `CDC_Receive` függvényt kellett módosítani az alábbi módon:

```
static int8_t CDC_Receive(uint8_t cdc_ch, uint8_t *Buf, uint32_t *Len)
{
    /* USER CODE BEGIN 6 */

    CDC_Transmit(cdc_ch, Buf, *Len); // echo back on same channel
    USBD_CDC_SetRxBuffer(0, &hUsbDevice, &Buf[0]);
    USBD_CDC_ReceivePacket(0, &hUsbDevice);
    USB_CDC_RxHandler(Buf, *Len); // my function
    return (USBD_OK);

    /* USER CODE END 6 */
}
```

Vagyis elegendő átadni a saját függvényemnek az kapott adatokat, majd abban kezelni azokat. Ennek kezelését részletesebben kifejtem az eszköz és a kliens alkalmazás kommunikációja részénél.

Ahhoz, hogy a soros kommunikáció biztonságos legyen az adatokat titkosítani kell. A titkosítás az ECDH (Elliptic-curve Diffie-Hellman) protokollon alapszik. Ez egy key agreement protokoll, amely lehetővé teszi két, elliptikus görbével rendelkező privát-publikus kulcspárral rendelkező fél számára, hogy megosztott titkot hozzanak létre egy nem biztonságos csatornán. Ez a megosztott titok közvetlenül használható kulcsként, vagy egy másik kulcs származtatására. A kulcs vagy a származtatott kulcs ezután felhasználható a következő kommunikáció titkosításához szimmetrikus kulcsú titkosítással.

A megoldásomban az eszköz kulcsainak generálásához micro-ECC SDK-t használtam. Ez egy olyan könyvtár, ami egy kis erőforrás igényű és optimalizált megvalósítását tartalmazza az EDCH és ECDSA (Elliptic Curve Digital Signature Algorithm) protokolloknak. Ezekből én csak az EDCH-t használtam. Ez a következő függvényben került felhasználásra:

```
void set_up_encryption()
{
    setting_up_encryption = true;
    srand(time(NULL));
    uECC_set_rng(my_rng_function);
    const struct uECC_Curve_t* curve = uECC_secp256r1();
    uECC_make_key(my_public_key, my_private_key, curve);
    res=false;
    count=1;

    while(!res)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_set);
        char num_str[10];
```

```

for(int i=0; i<public_key_length; ++i)
{
    snprintf(num_str, sizeof(num_str), "%d", my_public_key[i]);
    CDC_Transmit(acm_id, (uint8_t*)num_str, strlen(num_str));
    HAL_Delay(50);
    CDC_Transmit(acm_id, (uint8_t*)"\n", 2);
    HAL_Delay(50);
}
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_reset);

HAL_Delay(100);

step=READ_PUBLIC_KEY;
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_set);
while(!data_recieved);
data_recieved = false;
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_reset);

if(uECC_valid_public_key(their_public_key, curve)==1)
{
    uECC_shared_secret(their_public_key, my_private_key, secret,
                      curve);
}
else
    CDC_Transmit(acm_id, (uint8_t*)"E", 2);

CDC_Transmit(acm_id, (uint8_t*)"\n", 2);
HAL_Delay(50);

uint8_t msg[] = "Comm set\n";
uint8_t m[] = "Comm set";
encrypt_and_decrypt_msg(msg, 10);
msg[9] = '\n';

HAL_Delay(200);

CDC_Transmit(acm_id, msg, 10);
HAL_Delay(100);

step=READ_REPLY_MESSAGE; pos=0;
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_set);
while(!data_recieved)
{
    HAL_Delay(500);
    if(!data_recieved)
        CDC_Transmit(acm_id, msg, 10);
}
data_recieved = false;
step=READ_COMMAND;
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, led_reset);

data[8]='\0';
HAL_Delay(100);

encrypt_and_decrypt_msg(data, 8);

res=true;

```



```

        for(int i=0; i<8; ++i)
        {
            if(m[i]!=data[i])
                res=false;
        }

        HAL_Delay(100);
        if(res)
        {
            CDC_Transmit(acm_id, (uint8_t*)"Yes\n", 5);
        }
        else
        {
            CDC_Transmit(acm_id, (uint8_t*)"No\n", 4);
        }

        HAL_Delay(100);
    }

    setting_up_encryption = false;
}

```

A kulcsok generálásához először szükség van egy véletlenszám-generáló függvényre, amelynek fejléce a következő: `int my_rng_function(uint8_t *dest, unsigned size)`. Ez a függvény a `dest` változó által megadott címre kell, hogy `size` méretű véletlen bájtokat generáljon. Mivel a választott mikrokontroller nem rendelkezik TRNG (true random number generator) segédprocesszorral, a pszeudo-véletlenszám-generálást alkalmaztam. Ezt továbbfejlesztettem a mikrokontroller valós idejű órájának felhasználásával, hogy a lehetőségekhez képest minél jobb véletlenszám-generátort biztosítsak. Az alábbi függvényeket használtam ehhez:

A kulcs generáláskor meghívott függvény:

```

int my_rng_function(uint8_t *dest, unsigned size)
{
    if (dest == NULL || size == 0)
    {
        return 0;
    }

    for(int i=0; i<size; ++i)
    {
        dest[i] = get_pseudorandom_number();
    }
    return 1;
}

int generator_1() { return rand(); }

int generator_2() { return rand() * rand(); }

int generator_3() { return rand() ^ (rand() << 5); }

```

Ennek feladata, hogy a 3 egyszerűbb random generátorból válasszon egyet.

```
int choose_generator() { return rand() % NUM_GENERATORS; }
```

Feladata egy random bájt generálása a kiválasztott random generátor által kapott érték felhasználásával.

```
uint8_t get_pseudorandom_number()
{
    int generator_choice = choose_generator();
    int result;

    switch (generator_choice)
    {
        case 0: result = generator_1(); break;
        case 1: result = generator_2(); break;
        case 2: result = generator_3(); break;
        default: result = 0;
    }
}
```

A generáláskori idő felhasználása az entrópia növelése érdekében.

```
RTC_TimeTypeDef sTime;
uint32_t Format = RTC_FORMAT_BCD;

if(HAL_RTC_GetTime(&hrtc, &sTime, Format) == HAL_OK)
{
    uint32_t timer_value = 0;

    timer_value |= (uint32_t)sTime.Hours << 24;
    timer_value |= (uint32_t)sTime.Minutes << 16;
    timer_value |= (uint32_t)sTime.Seconds << 8;
    timer_value |= (sTime.SecondFraction & 0xFF);
    result ^= (timer_value & 0xFF);
}

return (uint8_t)(result & 0xFF);
}
```

A random generáló függvény beállítása után szükség van egy curve változóra, amely const struct uECC_Curve_t* típusú, és a kiválasztott elliptikus görbét tárolja. A könyvtárban több, a célnak megfelelő görbe is definiálva van, én ezek közül a uECC_secp256r1 görbét választottam. Az ezen a görbén alapuló privát kulcs és közös titok mérete 32 bájt, míg a publikus kulcsé 64 bájt. Ez a választás megfelelő biztonságot nyújt, miközben minimális overheadet eredményez az üzenetek titkosítása során. A beállítások elvégzése után a rendszer legenerálja a saját kulcsait, majd elküldi a publikus kulcsát a másik félnek, és megvárja, hogy megérkezzen az ő publikus kulcsa. Amennyiben a kapott publikus kulcs érvényes, kiszámítja a közös titkot. A generált kulcsokat felhasználva az encrypt_and_decrypt_msg függvény a ChaCha20 protokoll felhasználásával titkosítani tudja az elküldendő üzenetet.

Az `encrypt_and_decrypt_msg` függvény a következőt tartalmazza:

```
void encrypt_and_decrypt_msg(uint8_t* msg, size_t len)
{
    ChaCha20_init(&ctx, secret, nonce, count++);
    ChaCha20_xor(&ctx, msg, len);
}
```

Az `init` folyamatban a korábban generált közös titok, valamint egy mozgó változó, a `count` is felhasználásra kerül. A `count` biztosítja, hogy az azonos üzenetek is eltérő titkosított formában haladjanak át a csatornán, így egy esetleges közbeékelődött támadó (middle man) sem tudja a viselkedés elemzésével felfedni a felek közötti kommunikációt. Ezeket a beállításokat a `ctx` változó tárolja. A titkosítás során az `xor` függvény segítségével a `msg` változó felülírásra kerül, immár titkosított üzenettel.

Az üzenet elküldése után a rendszer megvárja a megerősítő üzenetet, amellyel létrejön a titkosított csatorna. Ezen a csatornán keresztül a felek már biztonságosan kommunikálhatnak egymással.

Ezzel elkészült a feladat első része, amely a hardver konfigurációját és a jelszavak hardveroldali kezelését foglalja magában.

A feladat következő része a kliens alkalmazás és felhasználói felület kialakítása volt. A program a következő osztályokból áll: `Program`, `App`, `ChaCha20`, `Login`, `MyRenderer`, `PWMan` és `USB_Comm`. A `Program` osztályban található a `Main` függvény, ami az egész program belépési pontja. Ebben meghívásra kerül az `usb` kommunikáció és az applikáció konstruktorai.

```
static void Main()
{
    ApplicationConfiguration.Initialize();
    USB_Comm usbComm = new USB_Comm();
    _ = new App(usbComm);
}
```

Az `App` osztály inicializálja a `Login`, `PWMan` és `hiddenform` komponenseket, majd elindítja a felhasználói felületet.

```
public App(USB_Comm usbComm)
{
    InitializeComponents(usbComm);
    RunApplication();
}
```

```
private void InitializeComponents (USB_Comm usbComm)
{
    PWman = new PWMan();
    loginForm = new Login();

    PWman.logOut += LogOut;
    loginForm.LoginSuccessful += OnLoginSuccessful;
}
```

A rejtett form ahhoz kell, hogy az alkalmazás tudjon a bejelentkezés és a bejelentkezett felhasználó felületek között váltani. Így az egyik view bezárásával (például login vagy logout kor) nem záródik be az egész program.

```
hiddenForm = new Form
{
    ShowInTaskbar = false,
    Opacity = 0,
    FormBorderStyle = FormBorderStyle.None,
    StartPosition = FormStartPosition.Manual,
    Size = new System.Drawing.Size(1, 1),
    Location = new System.Drawing.Point(-10000, -10000)
};
```

Feliratkoztatom a változókat, hogy az X megnyomásakor mindegyik lépjen ki és az megnyitott kommunikációs port is bezárásra kerüljön.

```
PWman.FormClosed += (sender, e) => Application.Exit();
loginForm.FormClosed += (sender, e) => Application.Exit();
hiddenForm.FormClosed += (sender, e) => Application.Exit();
PWman.FormClosed += usbComm.Program_FormClosed;
}
```

Ha volt elmentett bejelentkező egy korábbi sessionból, akkor abba lép vissza. Ha nem volt, akkor pedig a bejelentkező felület nyílik meg.

```
private void RunApplication()
{
    string filePath = Program.projectFolderPath + "\\defaultuser.txt";

    if (File.Exists(filePath))
    {
        string user_email = App.LoadUserEmail(filePath);
        OnLoginSuccessful(user_email);
    }
    else
    {
        loginForm.hideErrors();
        loginForm.Show();
    }

    try
    {
        Application.Run(hiddenForm);
    }
    catch { }
}
```

Kiolvassa a defaultuser.txt ben lévő email címet és visszaadja azt a hívónak.

```
private static string LoadUserEmail(string filePath)
{
    string user_email = "";
    string[] lines = File.ReadAllLines(filePath);
    foreach (string line in lines)
    {
        if (!string.IsNullOrEmpty(line))
        {
            user_email = line.Substring(line.IndexOf("email:") + "email:".Length).Trim();
            break;
        }
    }
    return user_email;
}
```

Átadja jelszó kezelő osztálynak a beolvasott email címet, majd betölti a bejelentkezett felhasználó felületét és elrejt a bejelentkező felületet.

```
private void OnLoginSuccessful(string userEmail)
{
    PWman.setEmail(userEmail);
    PWman.reload();
    PWman.Show();
    loginForm.Hide();
}
```

Kilépéskor elrejt a korábbi user felületét, elrejt a korábbi bejelentkezéskor fellépett hibák üzenetét, majd megjeleníti a bejelentkezési felületet. Ezt követően pedig törli a defaultusert, hogy a következő megnyitáskor ne bejelentkezve nyissa meg az alkalmazást.

```
private void LogOut(string s)
{
    PWman.Hide();
    loginForm.hideErrors();
    loginForm.Show();

    string filePath = Program.projectFolderPath + "\\defaultuser.txt";
    if (File.Exists(filePath))
    {
        File.Delete(filePath);
    }
}
```

A ChaCha20 osztály a mikrokontrollerhez használt library C#-beli megfelelője, hogy az egyik oldalon titkosított üzenetet fel lehessen oldani a másik oldalon.

A Login osztály felel a bejelentkezési felület megjelenítésért és az ezen végezhető műveletek kezelésért.

The screenshot shows a web application window titled 'Login'. It features two main sections: 'Log in' and 'Sign up'. The 'Log in' section includes a text input for 'Email or Username', a password input with masked characters, a checkbox labeled 'Keep me logged in', and a 'Log in' button. The 'Sign up' section includes text inputs for 'Displayed Name', 'Email', and 'Optional Username', a password input with masked characters, and a 'Sign up' button. The interface is dark-themed with a purple header bar.

Bejelentkezéskor a rendszer ellenőrzi, hogy minden szükséges mező ki van-e töltve, és hogy az adatok megfeleltethetők-e egy már regisztrált felhasználónak. Ha valamelyik feltétel nem teljesül, a rendszer értesíti a felhasználót a hiányosságokról vagy hibákról.

Password is not correct User not found!

Regisztrációkor a bejelentkezéshez hasonlóan ellenőrzi, hogy a kötelező adatok (név, email, jelszó) ki vannak-e töltve. Ezután megvizsgálja, hogy az adott email címmel vagy felhasználónévvel már létezik-e felhasználó. Ha minden feltétel teljesül, a rendszer hozzáadja az új felhasználót a felhasználók listájához, létrehoz számára egy saját mappát, és kijelzi, hogy sikeres volt a regisztráció. Ha valahol megakadna a folyamat, akkor visszajelzi ezt a felhasználónak, hogy ki tudja javítani.

Email already in use. Log in instead!

Username already in use. Choose another!

Use valid email address

A MyRenderer osztály segítségével módosítom a C# alkalmazás alapértelmezett beállításait, hogy sötét stílust alkalmazzanak. Ebben az osztályban például felüldefiniálhatók a vezérlők megjelenéséért felelős események és színek, így a különböző elemek (gombok, háttér, szöveg stb.) sötét színsémát kapnak.

```

public class DarkModeRendererer : ToolStripProfessionalRenderer
{
    public DarkModeRendererer() : base(new DarkColorTable()) { }

    // Method to set the text color for sub-menu items
    public static void SetSubItemColor(ToolStripMenuItem menuItem)
    {
        foreach (ToolStripItem subItem in menuItem.DropDownItems)
        {
            subItem.ForeColor = Color.FromArgb(241, 241, 241);

            if (subItem is ToolStripMenuItem subMenuItem)
            {
                DarkModeRendererer.SetSubItemColor(subMenuItem);
            }
        }
    }
}

public class DarkColorTable : ProfessionalColorTable
{
    public override Color MenuStripGradientBegin => Color.FromArgb(30, 30, 30);
    public override Color MenuStripGradientEnd => Color.FromArgb(30, 30, 30);

    public override Color ToolStripDropDownBackground => Color.FromArgb(45, 45, 45);

    public override Color ImageMarginGradientBegin => Color.FromArgb(45, 45, 45);
    public override Color ImageMarginGradientMiddle => Color.FromArgb(45, 45, 45);
    public override Color ImageMarginGradientEnd => Color.FromArgb(45, 45, 45);

    public override Color MenuItemSelected => Color.FromArgb(50, 50, 50);

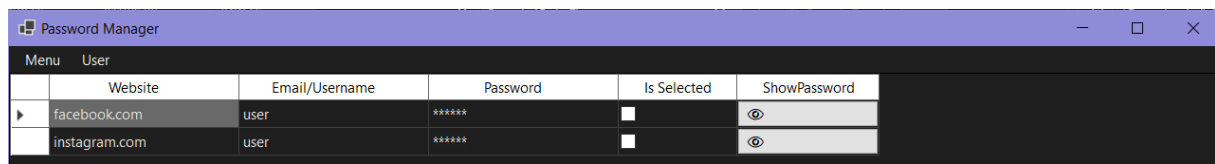
    public override Color MenuItemPressedGradientBegin =>
Color.FromArgb(50, 50, 50);
    public override Color MenuItemPressedGradientEnd => Color.FromArgb(50, 50, 50);

    public override Color ToolStripBorder => Color.FromArgb(30, 30, 30);

    public override Color MenuItemSelectedGradientBegin =>
Color.FromArgb(60, 60, 60);
    public override Color MenuItemSelectedGradientEnd => Color.FromArgb(60, 60, 60);
    public override Color MenuItemBorder => Color.FromArgb(60, 60, 60);
}

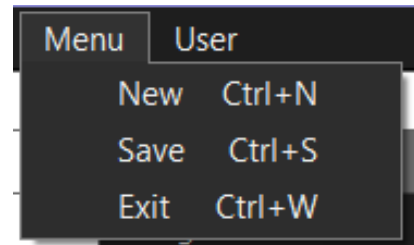
```

A PWMan a jelszó kezelés nézetért felel, ami egy felhasználó bejelentkezése után jelenik meg.



Menu	User	Website	Email/Username	Password	Is Selected	ShowPassword
		facebook.com	user	*****	<input type="checkbox"/>	
		instagram.com	user	*****	<input type="checkbox"/>	

A menüben lehetőség nyílik új jelszavak hozzáadására, a korábbi állapothoz képesti módosítások mentésére, valamint a korábban létrehozott jelszavak törlésére. Ez a kialakítás azért praktikus, mert a jelszavak mikrokontrollerre történő mentése lassú és erőforrás-igényes folyamat, amely leterheli az eszközt. Emiatt nem érdemes minden kisebb változás után mentést végezni. A jelszavak beírásakor sem szükséges azonnali mentés, mivel a rendszer a nézetben megjelenő adatokat használja. A menüpontok nevei mellett található shortcutok segítségével a műveletek kényelmesen elindíthatók. A felhasználó a jelszavak törlését a megfelelő sor kijelölésével és a *Delete* billentyű megnyomásával végezheti el. Továbbá, a *User* menü tartalmazza a *Log Out* opciót, amellyel a felhasználó kiléphet az aktuális munkamenetből, és visszatérhet a bejelentkezési képernyőre.



Menu	User
New	Ctrl+N
Save	Ctrl+S
Exit	Ctrl+W

Új jelszó hozzáadásakor, illetve meglévő jelszavak szerkesztésekor a program ellenőrzi, hogy a felhasználó kitöltötte-e a sor összes szükséges elemét. Emellett azt is validálja, hogy a megadott weboldal érvényes formátumú legyen biztosítva ezzel az adatok pontosságát és használhatóságát. Ha bármelyik feltétel nem teljesül, a program figyelmezteti a felhasználót, és kéri a hibás vagy hiányzó adatok javítását.

```
private static bool IsValidWebsite(string website)
{
    return website.Contains(".com") || website.Contains(".hu")
        || website.Contains(".org") || website.Contains(".wiki");
}

private bool IsRowFullyFilled(DataGridViewRow row)
{
    foreach (DataGridViewCell cell in row.Cells)
    {
        string columnName = row.DataGridView.
            Columns[cell.ColumnIndex].Name;

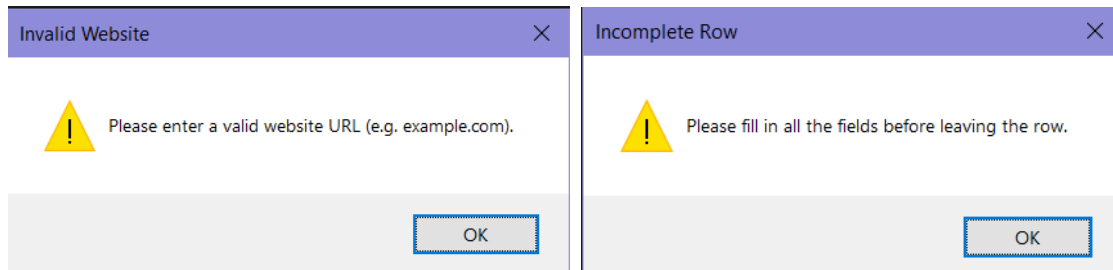
        if (columnName == RealPassword || columnName == IsShown
            || columnName == IsSelected)
        {
            continue;
        }
    }
}
```



```

        if (cell.Value == null
            || string.IsNullOrWhiteSpace(cell.Value.ToString()))
        {
            return false;
        }
    }
    return true;
}

```



Az utolsó osztály az USB_Comm, amit a következő részben fejtek ki.

Az utolsó feladat a hardver és a kliens alkalmazás közötti kommunikáció megvalósítása. A kommunikáció kezdetét a kliens alkalmazás adja az 1-es utasítás elküldésével. Ekkor a mikrokontroller a read_command állapotban van, és a beolvasott számot a mybuffer 0. pozíciójára menti. Miután a program visszatér a main függvény végtelen ciklusába, a switch szerkezet 1-es esetét felismerve meghívja a set_up_encryption függvényt, amely lehetővé teszi a felek közötti publikus kulcs cseréjét. Ha a beállítás sikeresen lezárul, a mikrokontroller ismét a parancsvárakozás állapotába kerül. Ezután a kliens alkalmazás elküldi a 2-es utasítást, amellyel a mikrokontroller beolvassa és elmenti, hogy a kliens rendszeren éppen milyen nyelvű billentyűzet van használatban. Ezt követően újra a parancsok fogadására vár. Ha a felhasználó kiválaszt egy sort, majd megnyomja az eszközön található gombot, az eszköz titkosított üzenetben értesíti a kliens alkalmazást arról, hogy szüksége van a felhasználónévre és a jelszóra. A kliens alkalmazás ezeket az adatokat szintén titkosítva elküldi az eszköznek. Az eszköz ezután a send_hid függvény meghívásával begépel a kapott felhasználónevet és jelszót.

A mikrokontrolleren a beolvasáshoz használt függvény a következő:

```
void USB_CDC_RxHandler(uint8_t* Buf, uint32_t Len)
{
```

A step változóban tárolom, hogy melyik lépésnél jár éppen a kód.

```
switch(step)
{
```

Itt olvasom be a kliens alkalmazás publikus kulcsát.

```
case READ_PUBLIC_KEY:
    if(Len==64)
    {
        for(int i=0; i<Len; ++i)
        {
            their_public_key[i] = Buf[i];
        }

        data_recieved = true;
        pos=0;
        ++step;
    } break;
```

Itt olvasom be a set_up_encryption()-höz a megerősítő üzenet. Ezzel biztosítom, hogy mindkét eszköz fel tudta oldani a titkosítást, így a későbbiekben is megfelelően fognak tudni kommunikálni.

```
case READ_REPLY_MESSAGE:
    if(Len==8)
    {
        for(int i=0; i<Len; ++i)
        {
            data[i] = Buf[i];
        }

        data_recieved = true;
        pos=0;
    } break;
```

A kliens alkalmazás utasításokat küldhet az eszköznek, hogy az mit csináljon. Ehhez itt olvasom be, hogy milyen kódot küldött.

```
case READ_COMMAND:
    if(Buf[0]>='0' || Buf[0]<='9')
    {
        mybuffer[0] = Buf[0]-'0';

        if(mybuffer[0]==SET_KEY_BOARD_LANG)
        {
            mybuffer[1] = Buf[2];
        }

        data_recieved = true;
        pos=0;
    } break;
```

Miután a felhasználó megnyomta a gombot és ezáltal kérte a kiválasztott jelszó beírását, a kliens alkalmazás titkosítva elküldi a felhasználó nevet és jelszót, amit itt fogadok. Ha esetleg a felhasználó azelőtt nyomta volna meg a gombot, hogy bármit kiválasztott volna, akkor nem kerül semmi elküldésre, hogy ez ne okozzon neki hibát.

```
case LOGIN_REQ:
    switch(pos)
    {
        case 0:
            if(Len==1)
            {
                nothing_to_send=true;
                data_recieved = true;
                break;
            }
            else
            {
                uint8_t buffer[Len];
                for(int i=0; i<Len; ++i)
                    buffer[i] = (wchar_t)Buf[i];

                encrypt_and_decrypt_msg(buffer, Len);
                email_or_username_len = Len;
                email_or_username =
                    (wchar_t*)malloc(Len*sizeof(wchar_t));
                for (int i = 0; i < Len; ++i)
                {
                    email_or_username[i] = (wchar_t)buffer[i];
                }
            }
            ++pos;
            break;

        case 1:
            uint8_t buffer[Len];
            for(int i=0; i<Len; ++i)
                buffer[i] = Buf[i];
            encrypt_and_decrypt_msg(buffer, Len);
            password_len = Len;
            password = (wchar_t*) malloc(Len*sizeof(wchar_t));
            for (int i = 0; i < Len; ++i)
            {
                password[i] = (wchar_t)buffer[i];
            }
            data_recieved = true;
            break;
    }
}
```

A függvény az eszköz különböző állapotaiban különböző adatokat vár, amiket különböző módon dolgoz fel. Ezeket az adott kódrészletnél kifejtettem.

Végül pedig az USB_Comm osztályról ejtenék néhány szót. A konstruktorában először legenerálom a titkosításhoz szükséges kulcsokat. Ezt követően elindítom a szálát, amely a tényleges kommunikációt végzi.

```
public USB_Comm()
{
    currentLanguage = InputLanguage.CurrentInputLanguage;
    name = currentLanguage.Culture.Name;

    domain_params = new ECDomainParameters(ecParams);
    generator.Init(new ECKeyGenerationParameters(domain_params, new
SecureRandom()));
    key_pair = generator.GenerateKeyPair();
    my_public_key = (ECPublicKeyParameters)key_pair.Public;
    my_public_key_bytes = my_public_key.Q.GetEncoded(false);

    waitHandle = new ManualResetEvent(false);

    Thread serialThread = new Thread(InitializeSerialCommunication);
    serialThread.Start();
}
```

Ebben a függvényben először megnyitom a virtuális soros portot. Ezután feliratkoztatom a myDataReceived függvényt az adat érkezés eseményre, amivel fogadni fogom a kezdeti beállításokhoz az adatot. A folyamat elindításához pedig elküldöm az eszköznek az 1-es parancsot. Miután a titkos csatorna sikeresen létrejött, elküldöm a 2-es parancsot és a billentyűzet aktuális nyelvét. Ezt követően átállítom az eseménykezelőt a processRequest függvényre, és várok, hogy megérkezzen a válasz.

```
private void InitializeSerialCommunication()
{
    try
    {
        serial_port = new SerialPort(COM_PORT, 19200, Parity.None, 8,
StopBits.One);
        serial_port.Handshake = Handshake.None;

        var init = new SerialDataReceivedEventHandler(myDataReceived);
        var req = new SerialDataReceivedEventHandler(processRequest);
        serial_port.DataReceived += init;

        serial_port.Open();

        if (serial_port.IsOpen)
        {
            serial_port.Write(SET_UP_ENCRYPTION);
        }
        else
        {
            Console.WriteLine("Serial is not open :(");
        }

        waitHandle.WaitOne();
        waitHandle.Reset();
    }
}
```

```

        string ms = "2 " + (name.Contains("hu") ? hu : en);
        serial_port.Write(ms);

        isComPortOpen = true;

        serial_port.DataReceived -= init;
        serial_port.DataReceived += req;

        waitHandle.WaitOne();

    }
    catch (Exception ex)
    {
        Console.WriteLine("Exception occurred: {ex.Message}");
    }
}

```

A myDataReceived függvényben fogadom az eszköz publikus kulcsát, majd elküldöm a saját publikus kulcsomat. Majd megvárom, hogy az eszköz visszajelezzon a folyamat sikerességéről.

A processRequest függvényben ellenőrzöm, hogy megfelelő üzenetet kaptam-e az eszköztől, valamint azt is, hogy a felhasználó kiválasztott-e már egy sort. Ha minden rendben van, a következő lépésben a felhasználó nevét és jelszavát kódolom, hogy titkosított formában elküldhessem az eszköznek.

Összegzésül elmondható, hogy sikerült megvalósítani a hardvert, amely képes beírni a felhasználó által kért jelszót, és egy kliens alkalmazást, amely képes ezeket kezelni. Bár a megoldás már így is elég komplex, néhány fejlesztési lehetőség még mindig van benne.

Például, sajnos időhiány miatt nem sikerült újra strukturálni a kliens alkalmazást úgy, hogy az eszköz flash memóriájában tárolja az adatokat ahelyett, hogy a kliens alkalmazás fájlrendszerében történne a mentés. Szintén tervben volt, hogy az adatok tárolása titkosítva történjen, így biztosítva, hogy ne lehessen őket egyszerűen kiolvasni a flash tárolóból.

Ahogy korábban említettem, az eszköz nem rendelkezik TRNG hardverrel. A biztonság javítása érdekében mindenképpen érdemes lenne olyan mikrokontrolleres boardot keresni, amely rendelkezik ezzel a funkcionalitással. A biztonságot azzal is lehetne növelni, ha az eszközhöz ujjlenyomat-olvasó is csatlakozna, így biztosítva, hogy csak az engedélyezett felhasználók férhessenek hozzá az eszközön tárolt jelszavakhoz.

A felhasználói élmény javítása érdekében érdemes lenne a soros kommunikációt optimalizálni. Jelenleg előfordul, hogy 1-2 perc is eltelik, mire feláll a titkos csatorna.

Egy másik fejlesztési lehetőség, ha az alkalmazás automatikusan elindulna a háttérben, amint az eszközt csatlakoztatják, és csak az adatok szerkesztéséhez lenne szükség az alkalmazás megnyitására. Ehhez pedig szükséges volna, hogy a kliens alkalmazás képes legyen kommunikálni a böngészővel, és onnan lekérni az éppen használt weboldalt.

A félév során rengeteget tanultam egy komplex projekt megvalósításáról, kiemelve csak néhány fontosabb elemet: a mikrokontrolleres alkalmazások fejlesztése, az USB kommunikáció működése és a titkosítási technikák alkalmazása. Ha valaki szeretné részletesen megtekinteni a projektet, az elérhető a következő GitHub repositoryban: [Project Laboratory](#).

Források:

- <https://github.com/bzolka/csharp-oop-intro>
- <https://www.radioshuttle.de/en/turtle-en/nucleo-st-link-interface-en/>
- https://stm32world.com/wiki/Black_Pill#google_vignette
- <https://www.instructables.com/STM32-As-HID-USB-Keyboar-STM32-Tutorials/>
- <https://gist.github.com/MightyPork/6da26e382a7ad91b5496ee55fdc73db2>
- https://usb.org/sites/default/files/hut1_3_0.pdf
- <https://youtu.be/dEQwSl8mCFs?si=-A3ERuqH8uAWe1YK>
- https://en.wikipedia.org/wiki/Elliptic-curve_Diffie-Hellman
- <https://youtu.be/gAtBM06xwaw?si=uYuT5FAk5mVCdMkn>
- https://youtu.be/d8I74SJVc8E?si=xVpD_ls8SIEyjWbw
- <https://youtu.be/vYp4SFYMKfQ?si=GqzQmw9QPIBPW-N6>
- <https://www.c-sharpcorner.com/UploadFile/eclipsed4utoo/communicating-with-serial-port-in-C-Sharp/>
- <https://github.com/kmackay/micro-ecc/tree/master>
- <https://github.com/marcizhu/ChaCha20>
- <https://github.com/alambe94/I-CUBE-USBD-Composite/issues/26>
- <https://deepbluembedded.com/stm32-usb-cdc-virtual-com-port-vcp-examples/>