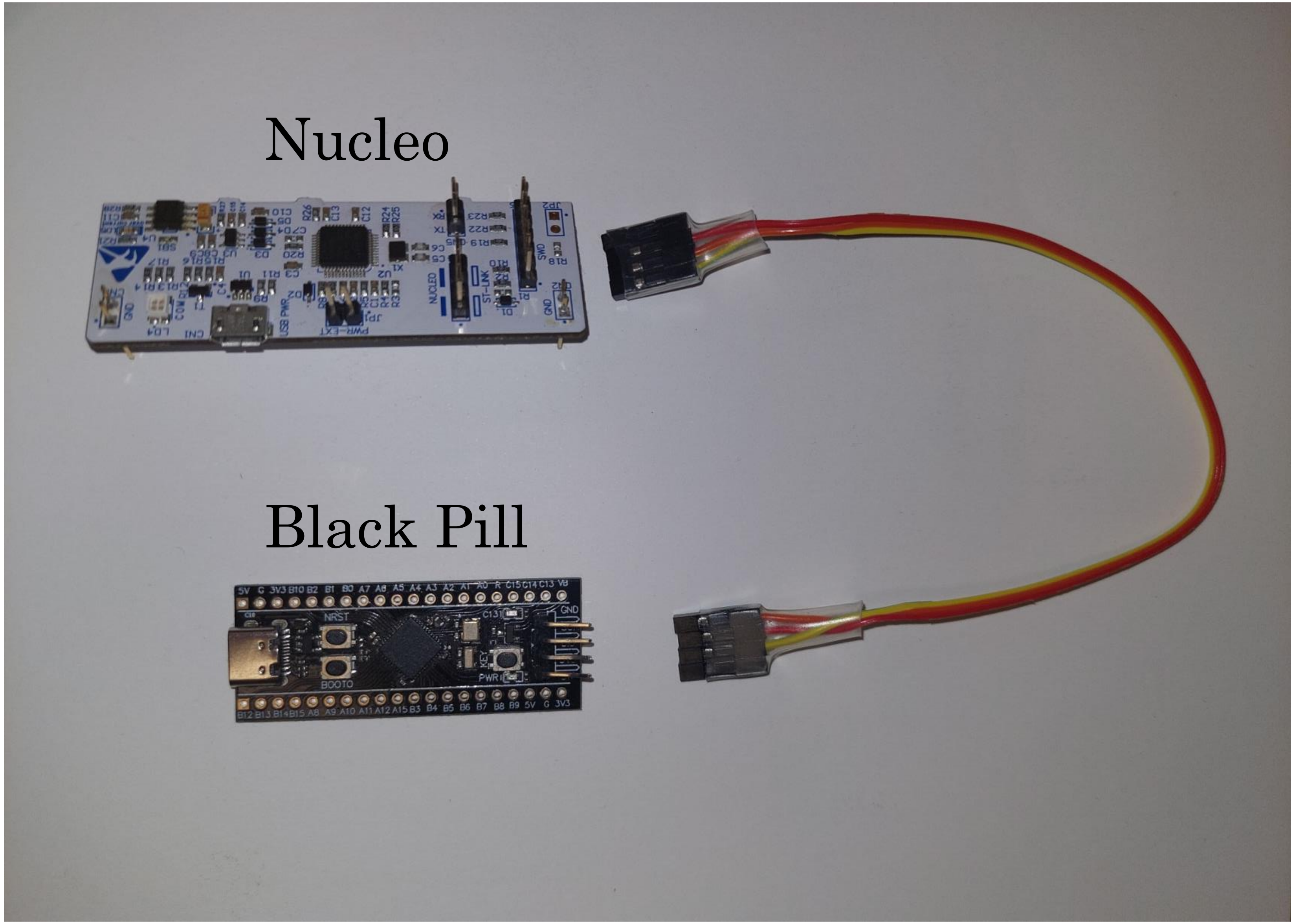
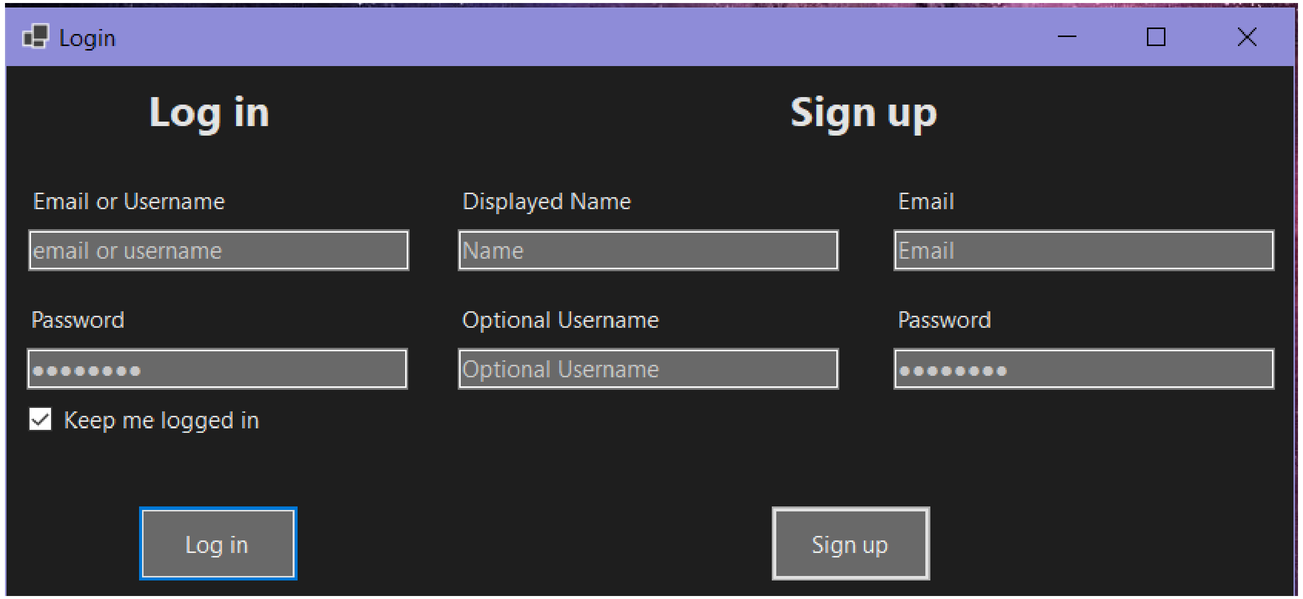
Jelszókezelő hardver fejlesztése

Készítette: Király Krisztina (IHLRE7)

Hardver



Szoftver



Email or Username: email or username

Password: ●●●●●●●●

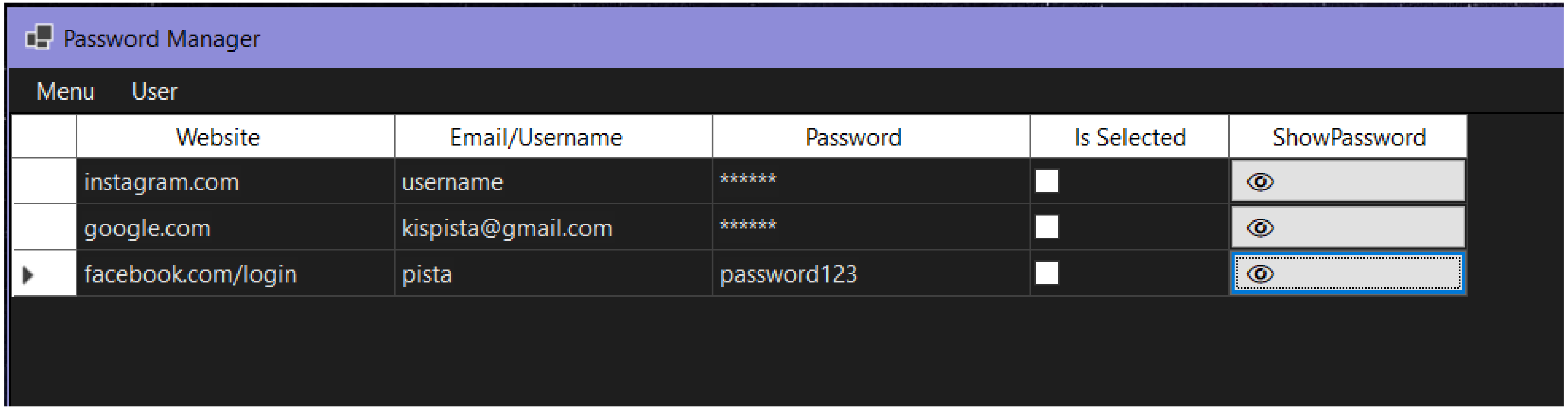
☑ Keep me logged in

Displayed Name: Name

Optional Username: Optional Username

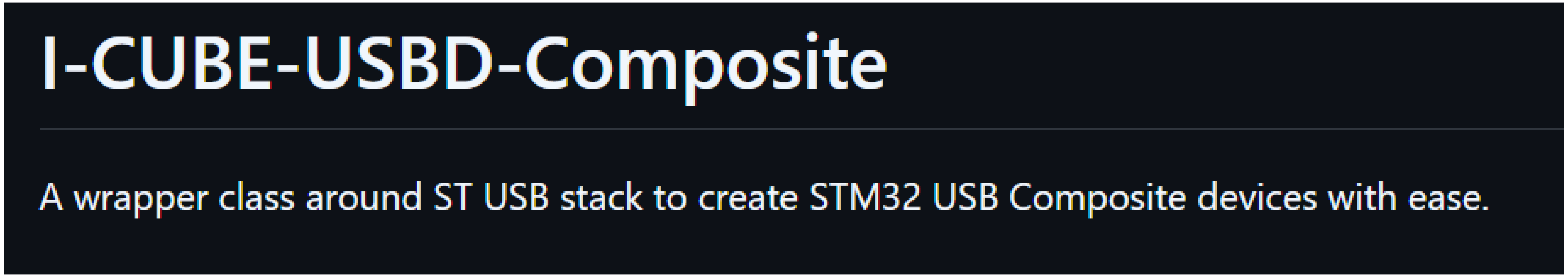
Email: Email

Password: ●●●●●●●●●



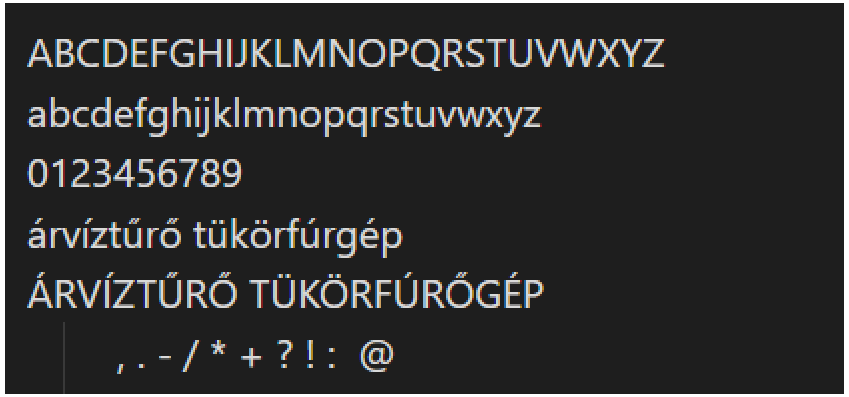
	Website	Email/Username	Password	Is Selected	ShowPassword
	instagram.com	username	******	☐	👁
	google.com	kispista@gmail.com	******	☐	👁
▶	facebook.com/login	pista	password123	☐	👁

Fejlesztés menete

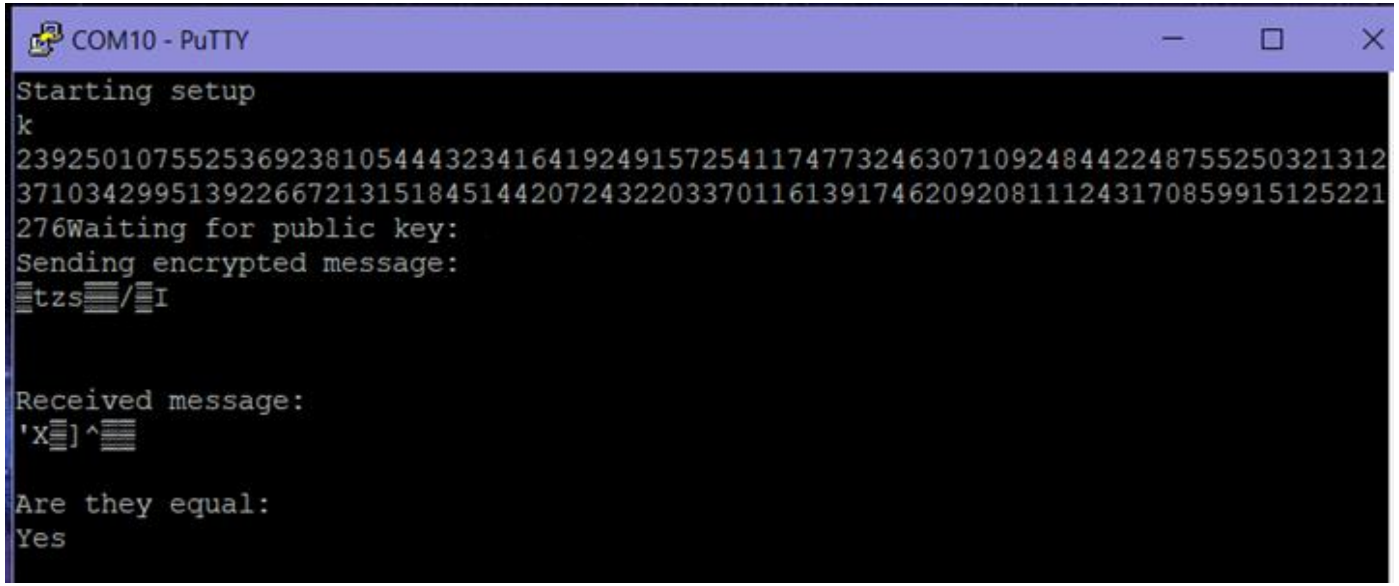


I-CUBE-USBD-Composite

A wrapper class around ST USB stack to create STM32 USB Composite devices with ease.

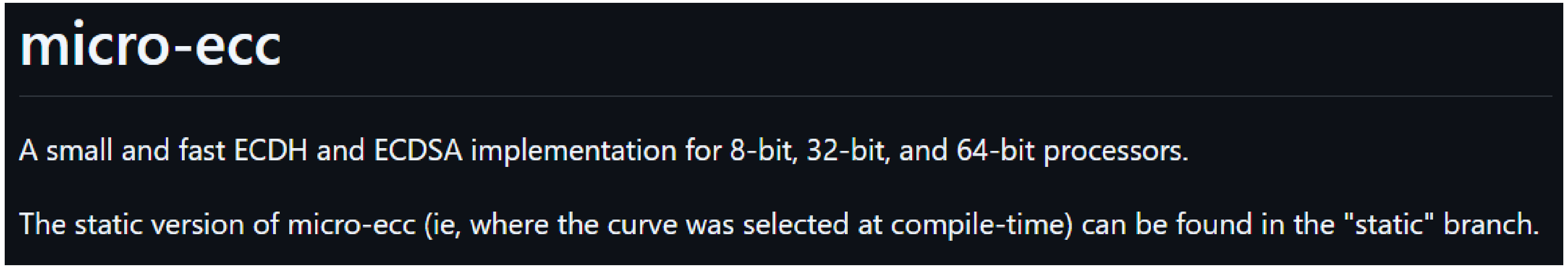


ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
árvíztűrő tükörfúrgép
ÁRVÍZTŰRŐ TÜKÖRFÚRŐGÉP
, . - / * + ? ! : @



```
Starting setup
k
23925010755253692381054443234164192491572541174773246307109248442248755250321312
37103429951392266721315184514420724322033701161391746209208111243170859915125221
276Waiting for public key:
Sending encrypted message:
▒tzs▒▒/▒I


Received message:
'X▒]^▒▒

Are they equal:
Yes
```

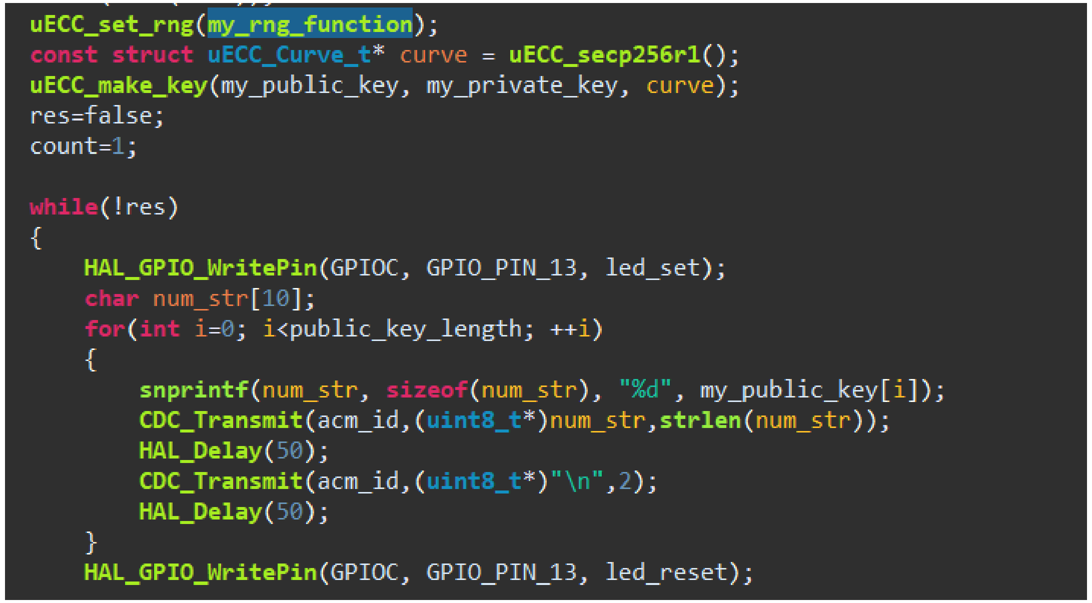
Fejlesztés menete



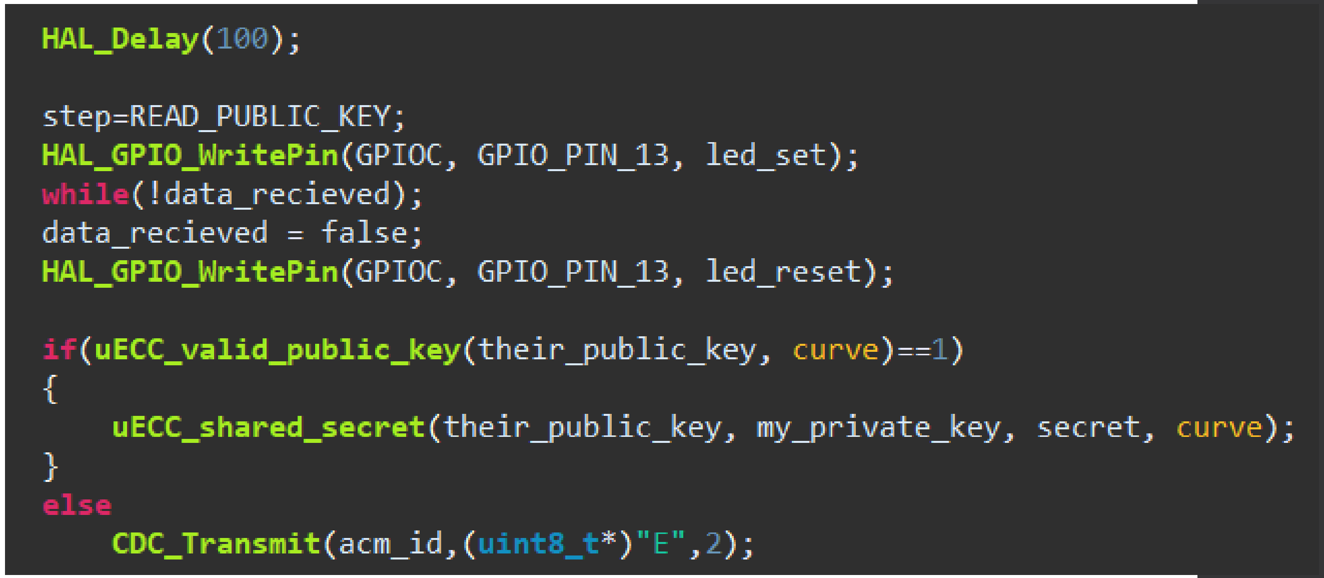
micro-ecc

A small and fast ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit processors.

The static version of micro-ecc (ie, where the curve was selected at compile-time) can be found in the "static" branch.

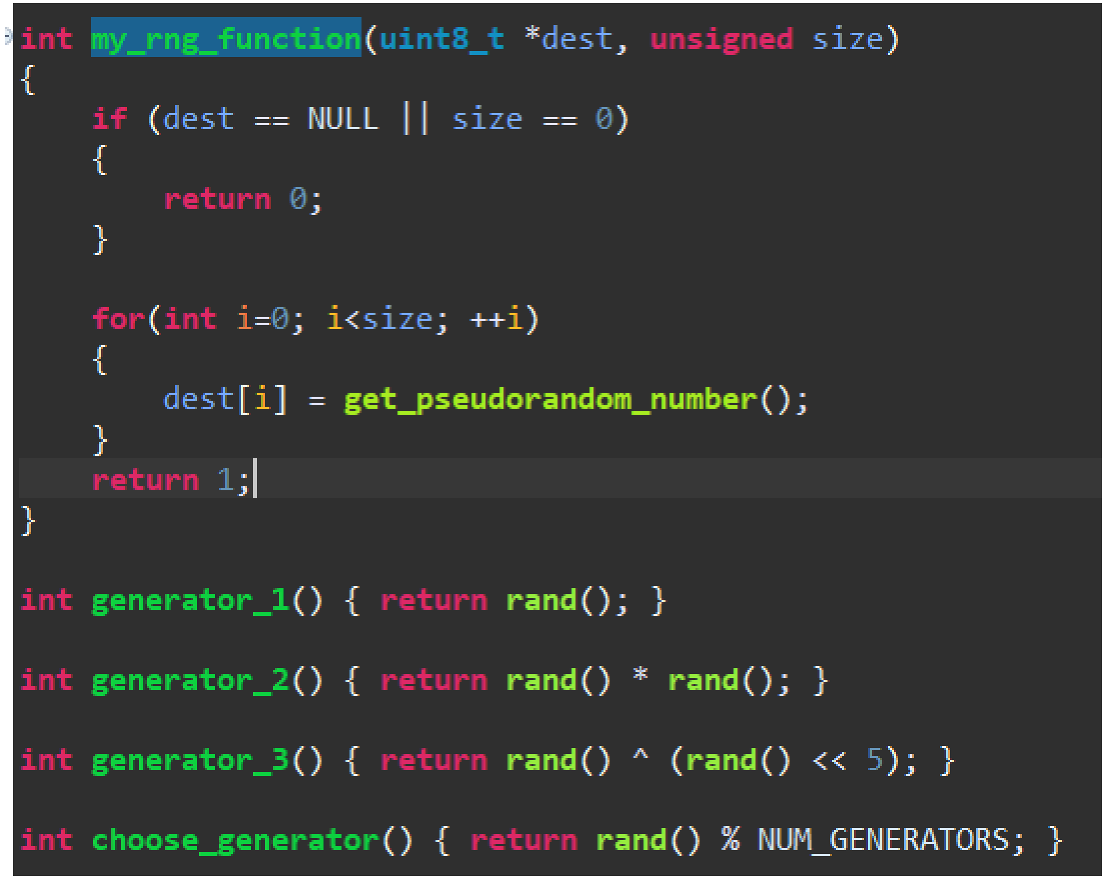


```
uECC_set_rng(my_rng_function);
const struct uECC_Curve_t* curve = uECC_secp256r1();
uECC_make_key(my_public_key, my_private_key, curve);
res=false;
count=1;

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);
```

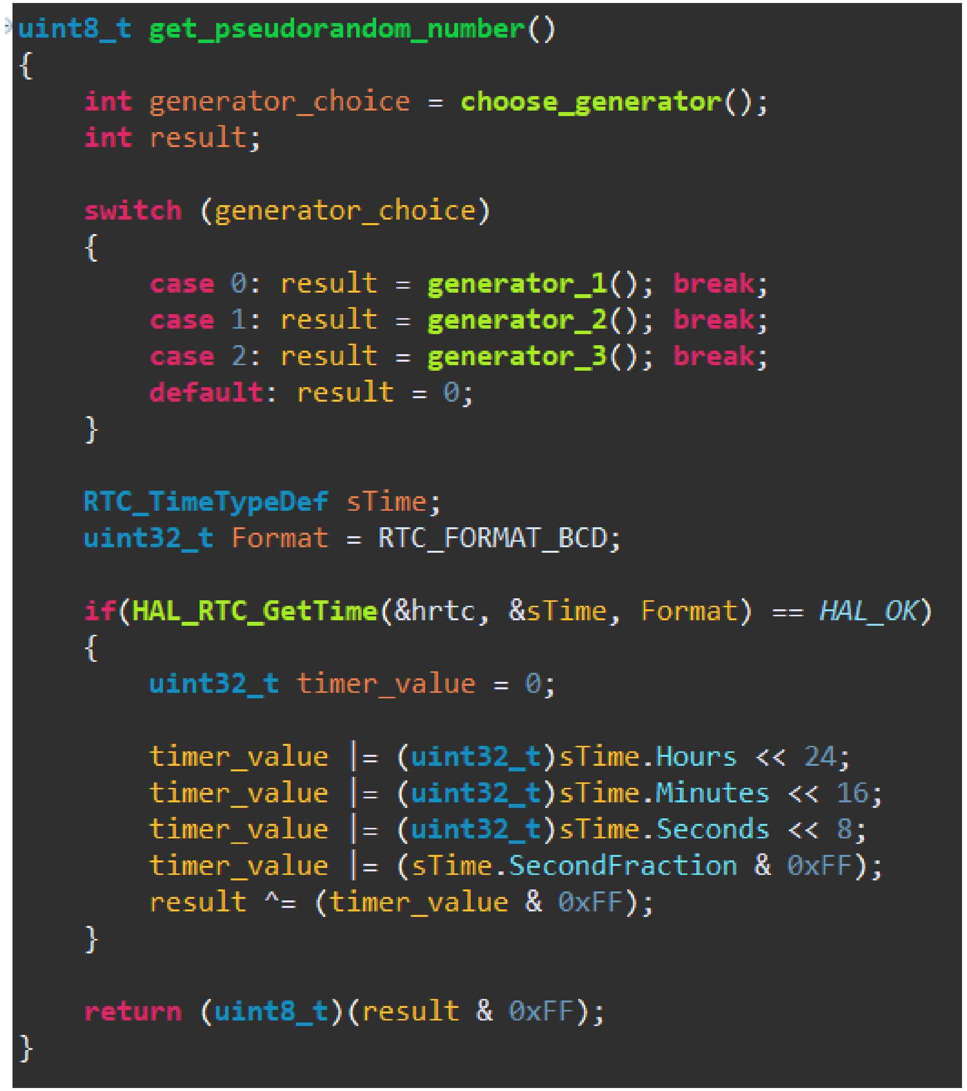


```
HAL_Delay(100);

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);
```

Fejlesztés menete

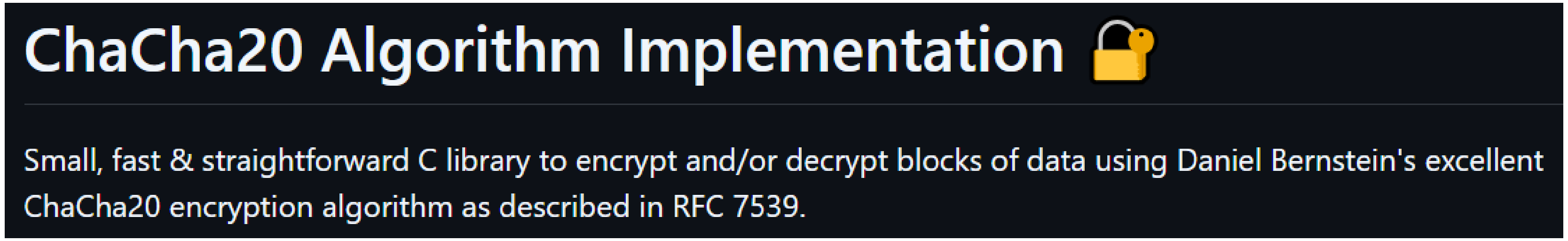


```c
int my_rng_function(uint8_t *dest, unsigned size)
{
    if (dest == NULL || size == 0)
    {
        return 0;
    }

    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); }

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



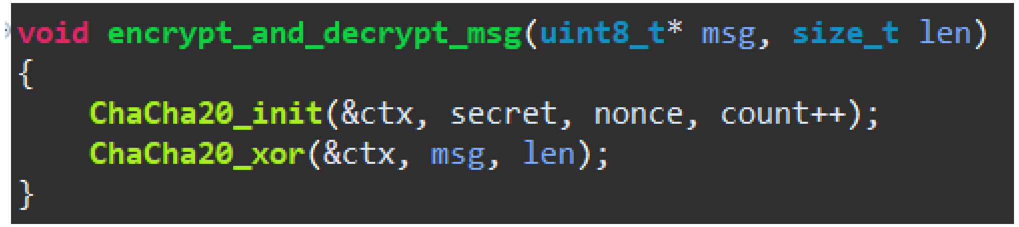
```c
uint8_t get_pseudorandom_number()
{
    int generator_choice = choose_generator();
    int result;

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

    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);
}
```

Fejlesztés menete

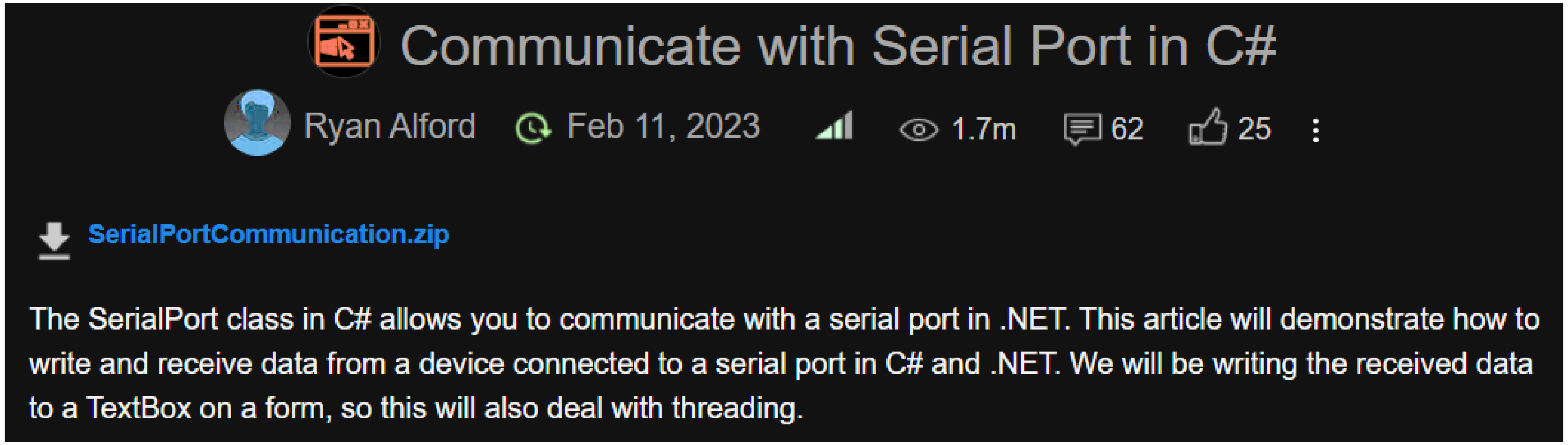


ChaCha20 Algorithm Implementation 🔐

Small, fast & straightforward C library to encrypt and/or decrypt blocks of data using Daniel Bernstein's excellent ChaCha20 encryption algorithm as described in RFC 7539.



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



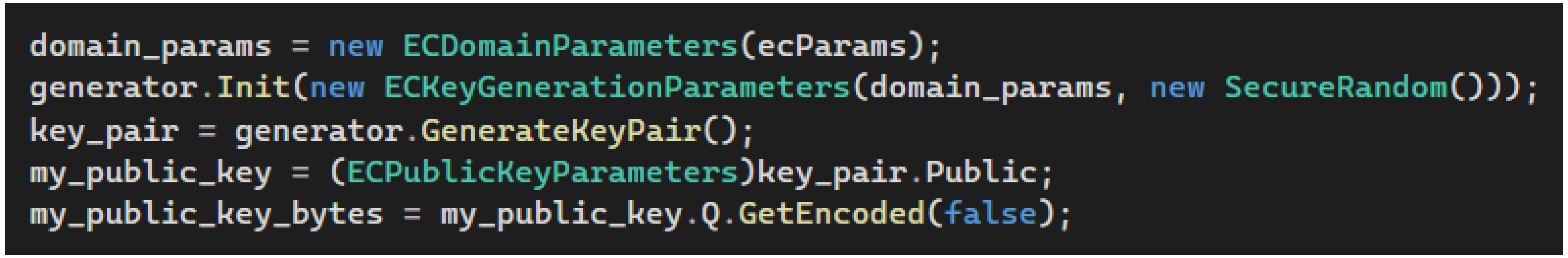
Communicate with Serial Port in C#

Ryan Alford Feb 11, 2023 1.7m 62 25

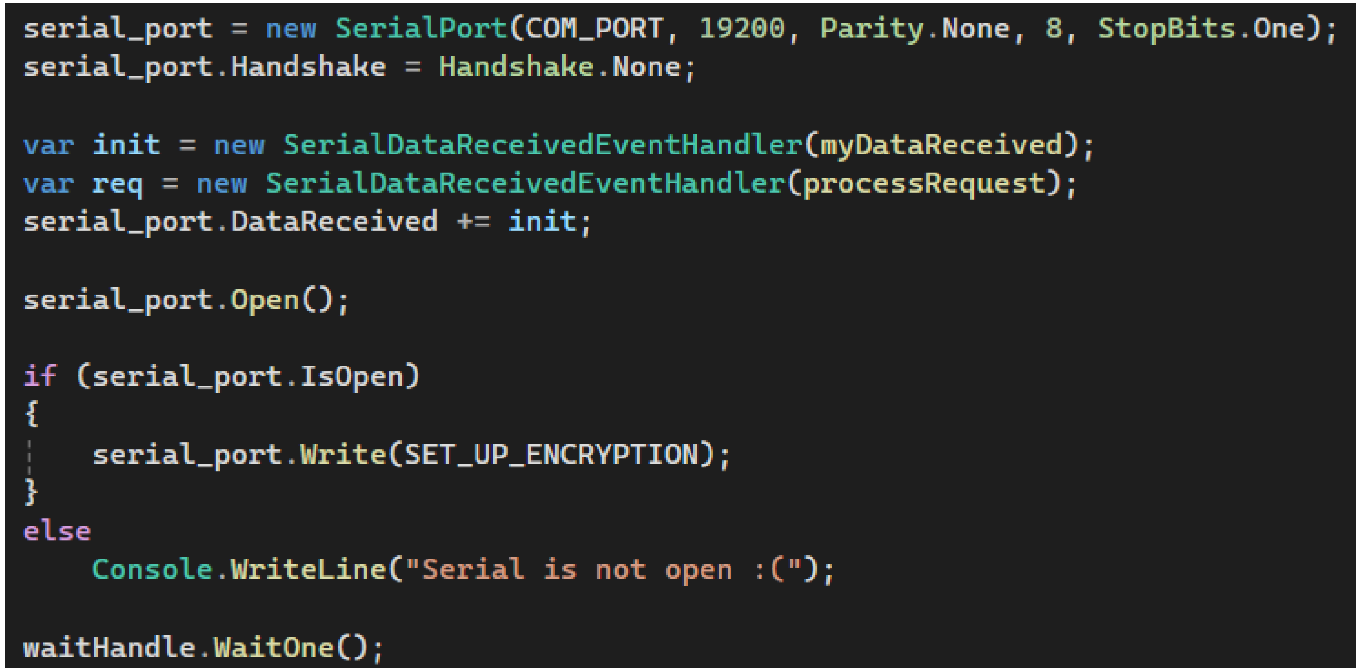
SerialPortCommunication.zip

The SerialPort class in C# allows you to communicate with a serial port in .NET. This article will demonstrate how to write and receive data from a device connected to a serial port in C# and .NET. We will be writing the received data to a TextBox on a form, so this will also deal with threading.

Fejlesztés menete

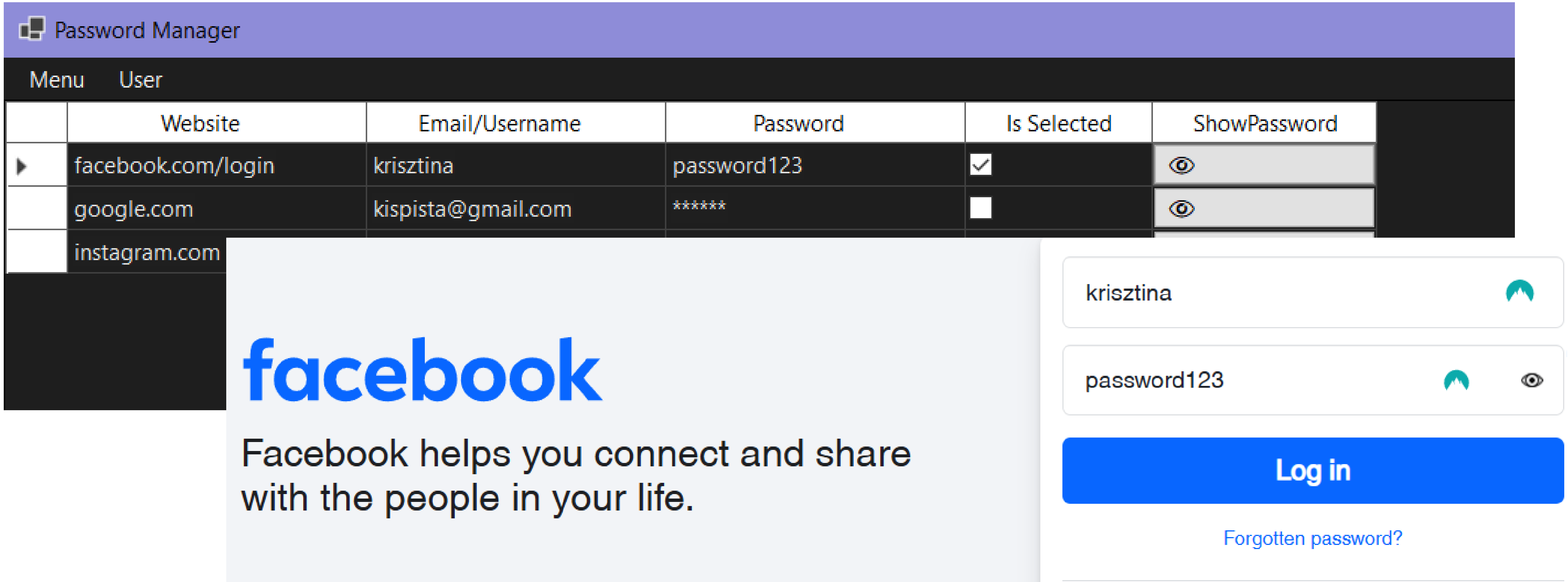


```
domain_params = new ECDomainParameters(ecParams);
generator.Init(new ECKeyGenerationParameters(domain_params, new SecureRandom()));
key_pair = generator.GenerateKeyPair();
my_public_key = (ECPublicKeyParameters)key_pair.Public;
my_public_key_bytes = my_public_key.Q.GetEncoded(false);
```



```
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();
```

Eredmény



	Website	Email/Username	Password	Is Selected	ShowPassword
▶	facebook.com/login	krisztina	password123	☑	👁
	google.com	kispista@gmail.com	******	□	👁
	instagram.com				

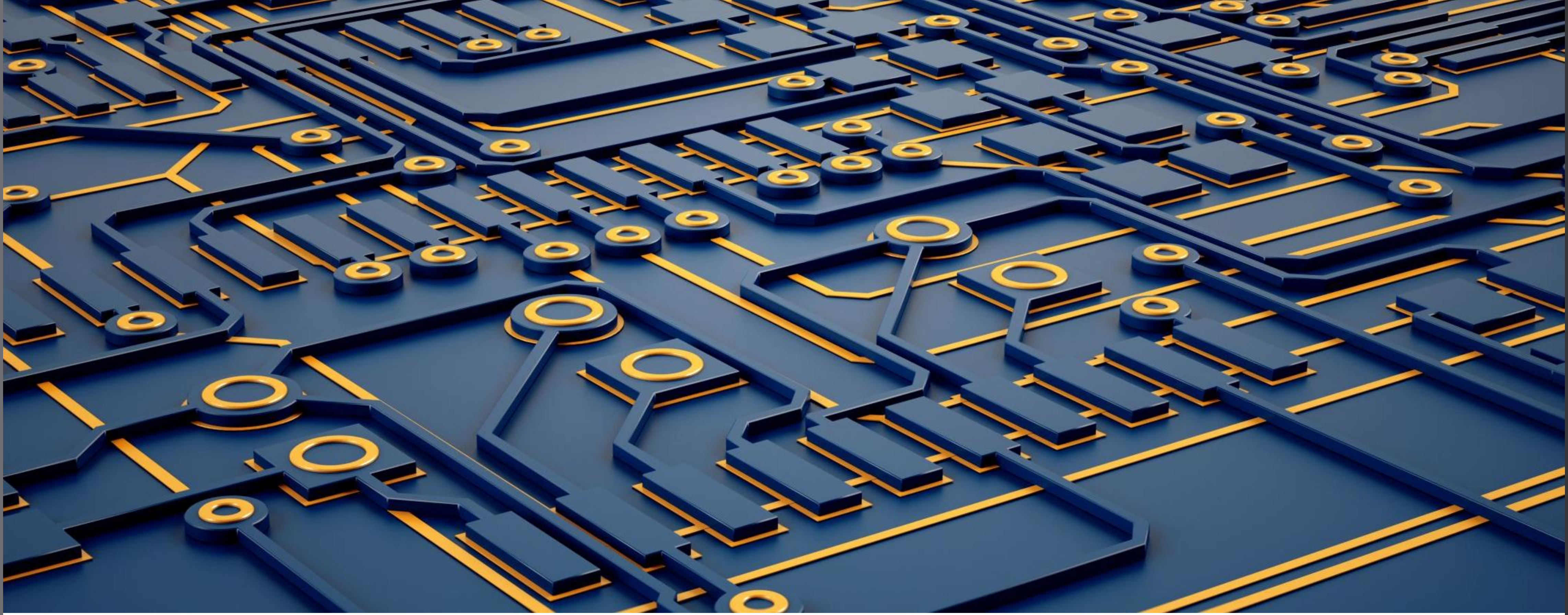
Facebook helps you connect and share with the people in your life.

krisztina

password123

Fejlesztési lehetőségek

- Weboldal felismerés
- Adatok tárolása az eszközön
- TRNG
- Soros kommunikáció optimalizálása
- Ujjlenyomat olvasóval kiegészítés



Köszönöm a figyelmet!