

USB Custom Human Interface Device + RUST Útmutató

A dokumentum bemutatja hogyan készítsd el nulláról a saját USB HID eszközt, amely RUST kódot is meghív C-ből, egy STM32F412ZG mikrovezérlőn. Próbáltam rugalmasan, más eszközre is megvalósíthatóan bemutatni. A forrásokat egyben legalul találod. Sok sikert! - Szabó Bence Kristóf

1. Környezet megteremtése

Legelső lépés, letölteni a Visual Studio Code-ot: <https://code.visualstudio.com/download>

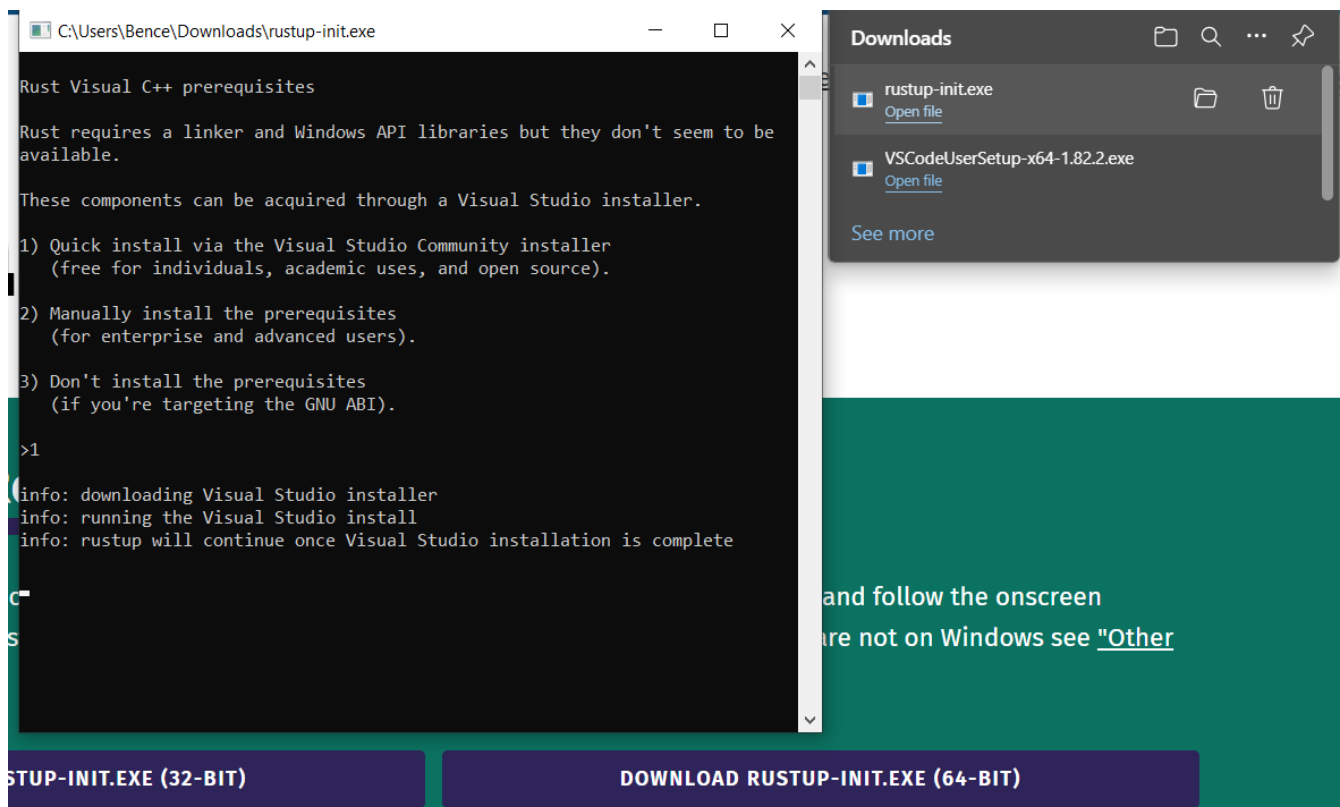
☒ Register Code as an editor for supported file types

☒ Add to PATH (requires shell restart)

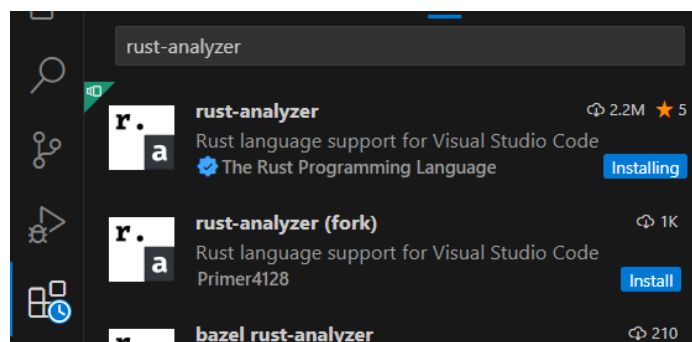
Telepítés során adjuk hozzá a környezeti változókhöz (PATH)

Következő, hogy beszerezzük a RUST telepítőjét -> <https://www.rust-lang.org/tools/install>

Futtatás után írd be a parancssorba a kívánt telepítési lehetőség számát: 1 - 2 - 3



Miután a rustup feltette magát, újra válassz a lehetőségek közül! Ha kész van, a VSCode-ban telepítsd a Rust-analyzer névre hallgató bővítményt:

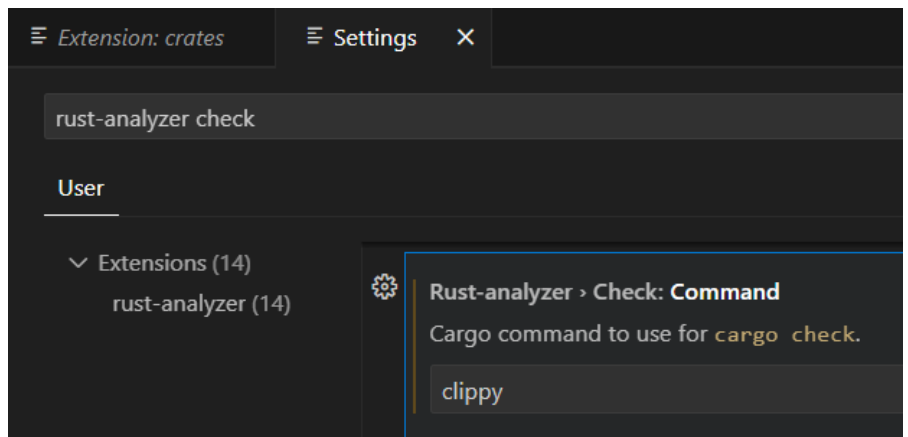
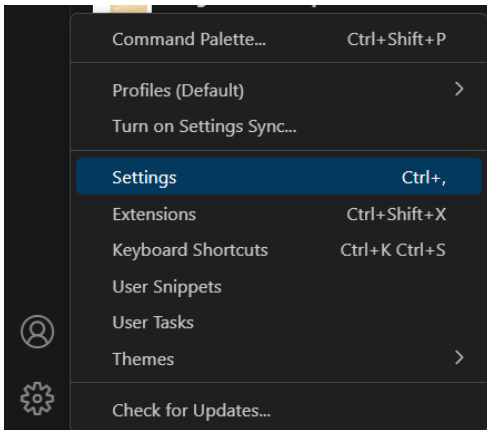


A következő részt Youtube videók alapján is végigcsinálhatod (ezen oldal alján találsz).

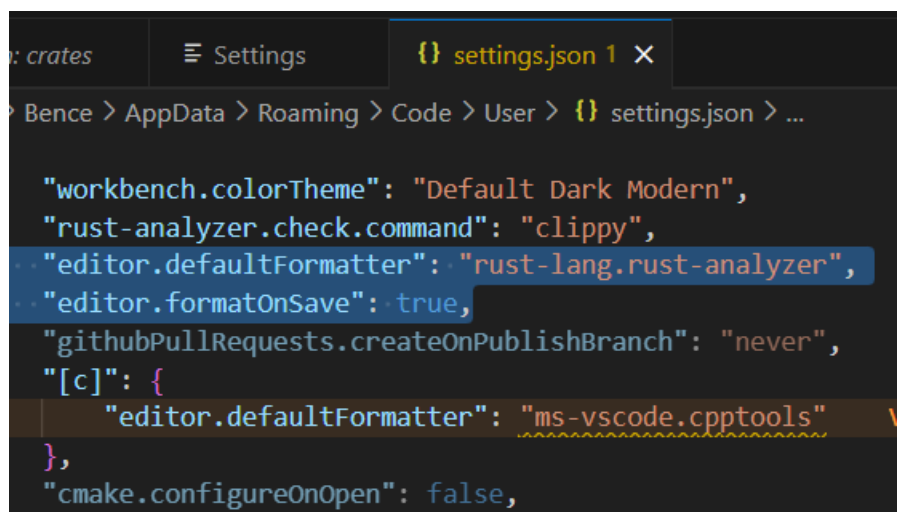
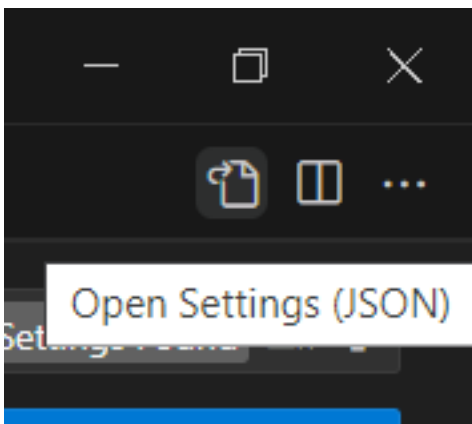
Az ehhez hasonló kommenteket ** karakterrel jelöltem.

A többi hasznos bővítmény: crates, CodeLLDB, Even Better TOML, Error Lens.

Még pontosabb visszajelzések érdekében állítsd át a "rust-analyzer check"-et clippy-re, ha még nem az.



Az "Open Settings (JSON)" lehetőségre kattintva, ellenőrizd az alábbiakat:



** Itt van 2 nagyszerű videó is a RUST telepítésével és bővítményeivel kapcsolatban.

	https://www.youtube.com/watch?v=yo4kWLtSPCY
	https://www.youtube.com/watch?v=BU1LYFkpJuk

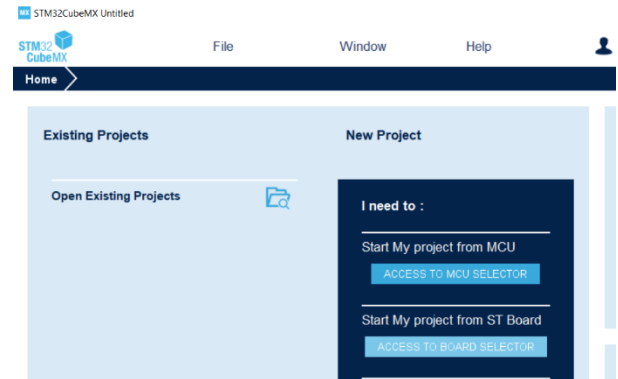
2. Custom HID generálása CubeMX-vel

Töltsd le és telepítsd fel a cubeMX-et: <https://www.st.com/en/development-tools/stm32cubemx.html>

Készíts egy Custom_USB_HID-et:

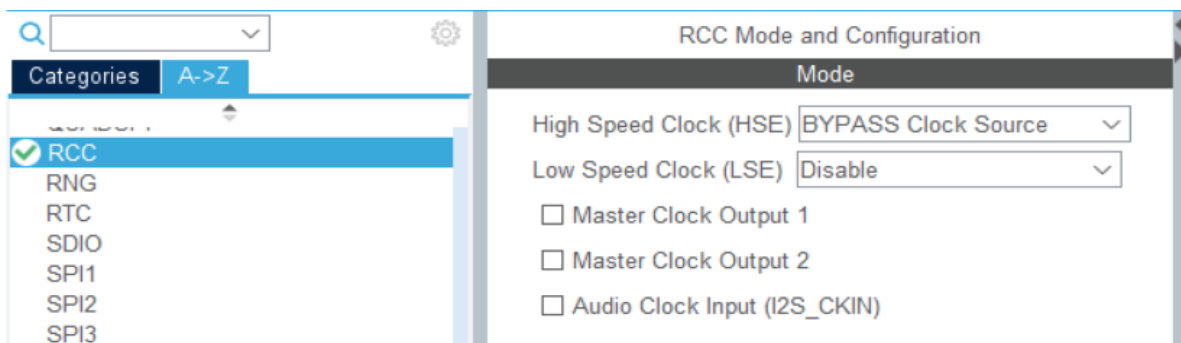
Először add meg, hogy mely pineket szeretnéd majd használni. (** Ledek: PB0, PB7, PB14, Gomb: PC13 - STM32F412ZG - esetében úgy emlékszem ezek).

1. Access to board selector (Válaszd ki az mcu-dat)
2. Clear Pinouts (Töröld az alap beállításokat!)
3. Setup Output/Input pin(s) (LEDs, Button),
névadáshoz jobb klikk, "Enter user label".

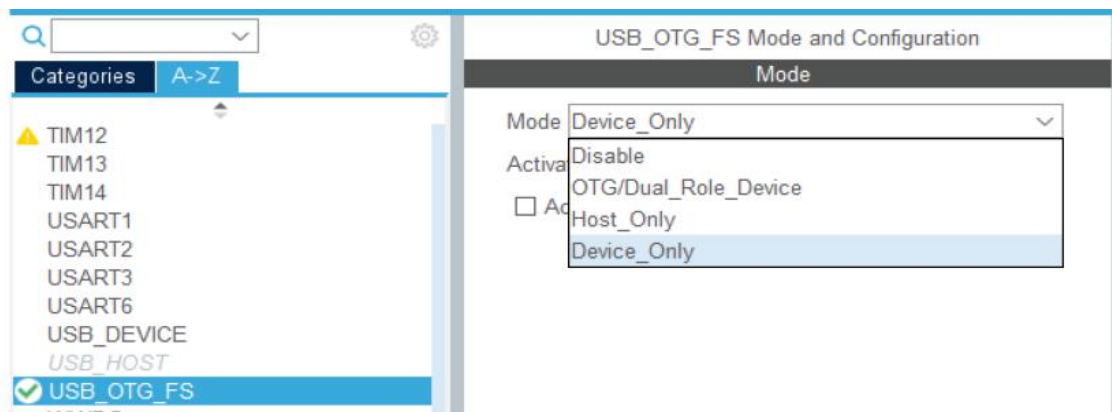


Most pedig állítsuk be azokat amik custom usb eszközzé teszik a mikrovezérlőnket.

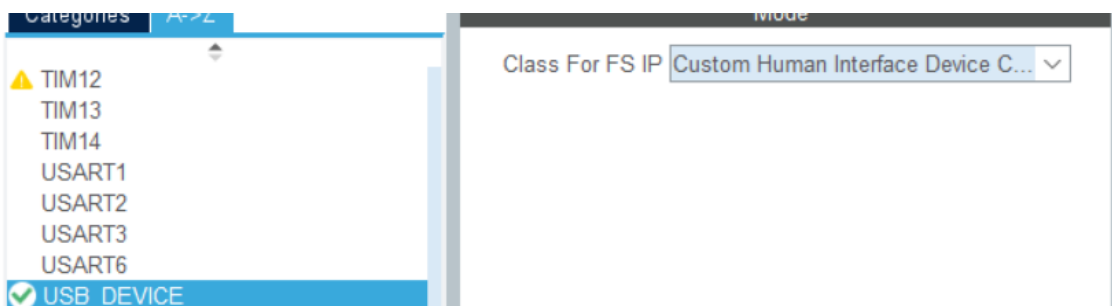
- Első ami kell az RCC, High Speed Clock >> BYPASS Clock Source



- Következő az USB_OTG_FS >> Device_Only



- Majd USB_DEVICE >> Custom Human Interface Device Class

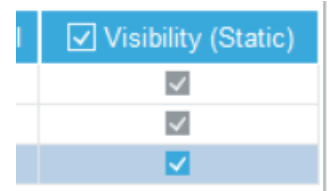


- Ugyanitt az USB_DEVICE-ban, csak lejjebb, átírható a PRODUCT_STRING. (Elnevezheted eszközüdet)

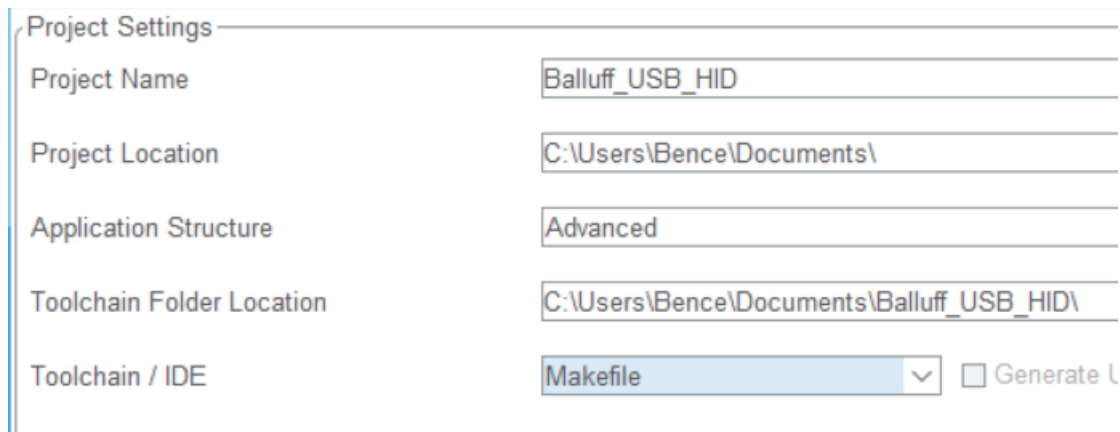


- Fentebb a nagy kék füleken váltsunk a következőre >> “Clock Configuration” és ha felkínálja fogadjuk el az automatikus beállításokat, ha nem készültünk olyan szándékkal, hogy mi magunk adjuk meg az értékeket.

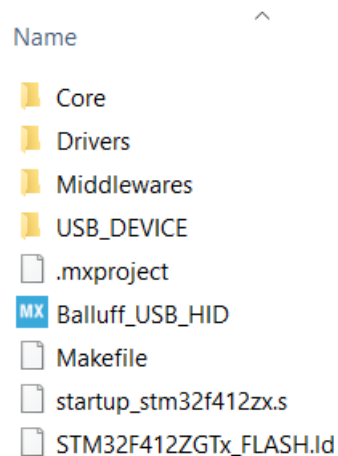
Következő nagy kék fül >> “Project Manager” azon belül az Advanced settings >> Visibility lehetőségénél minden legyen pipa. Ehhez katt a tetején a kék sávra.



A “Project Manager”-ben most a Project beállításai között a legfontosabb, hogy a “Toolchain / IDE” lehetőséget állítsuk “Makefile”-ra. (Így tudunk majd vele Visual Studio Code-ban dolgozni).



- Ha valami szükséges (pl: a mikrovezérlőd fájlljai), azt ki fogja dobni a CubeMX és akkor töltsd le. Lehet arra kér majd, hogy jelentkez be, stb-stb. Tégy neki eleget és egy sikeresen generálódott kód lesz a jutalmad.



** Ha valami nem sikerült, segítenek az alábbi videók megcsinálni, de te cubeMX-ben, ne pedig CubeIDE készítsd el. <https://www.youtube.com/watch?v=3JGRt3BFYrM>

Ne feledd, hogy makefile-ra van szükségünk! <https://www.youtube.com/watch?v=ZP2fd2qatj0>

3. RUST projekt létrehozása

VSCode-ban, hozz létre RUST projektet ugyanazon névvel, mint a cubeMX projekted neve.

\$ cargo new [a projected neve] --lib (**Ha máskor alkalmazást szeretnél akkor --bin)

```
PS C:\Users\Bence> cd C:\Users\Bence\Documents\RustProjects
PS C:\Users\Bence\Documents\RustProjects> cargo new Balluff_USB_HID --lib
Created library `Balluff_USB_HID` package
PS C:\Users\Bence\Documents\RustProjects>
```

Másold át a generált fájlokat ezen RUST projekted mappájába!

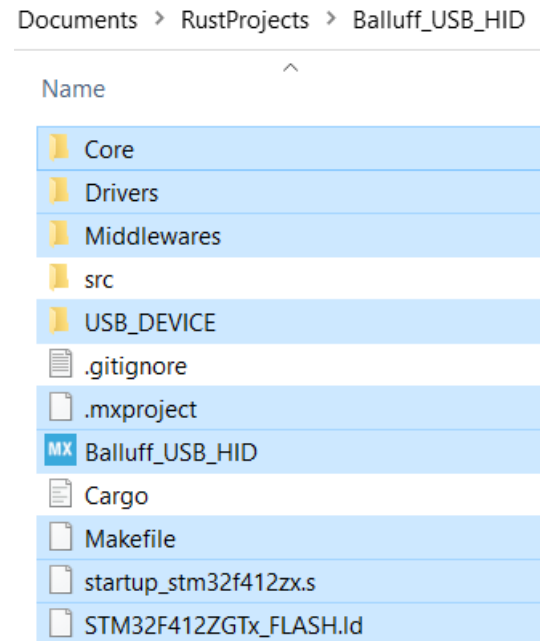
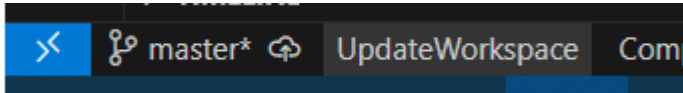
Nyisd meg a mappát és kattints a(z) “src/lib.rs” fájlba.

Ha a rust-analyzer lefutott telepítsd még az alábbi bővítményeket:

- C/C++
- C/C++ Extension Pack
- Makefile Tools
- STM32 VS Code Extension
- STM-Helper.

(**Ne lepődj meg, hogy több bővítmény is telepítődik ezekkel)

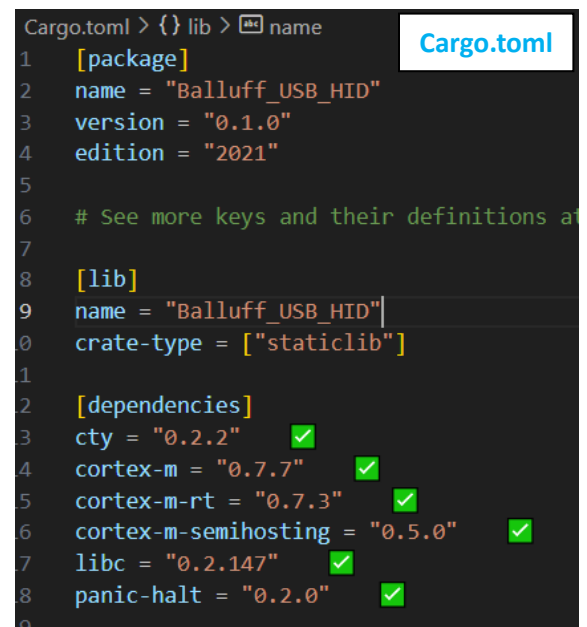
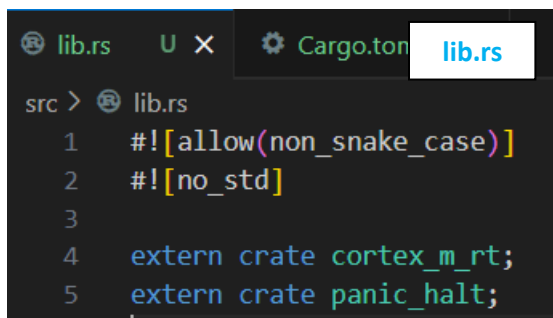
A helpernek köszönhetően megjelennek alul újabb lehetőségek, kattins az “UpdateWorkspace” lehetőségre!



Generált egy “.vscode” nevű mappát. Nyomj rá a “build”-re! Ha mindent jól csináltunk, sikeresen lefutott.

**Nekem a “c_cpp_properties” fájl errort jelzett, mert nem talál egy mappát. Az útvonalban 9.2.1 verzió volt, de a gépemen 10.3.1 van. Megoldás: csak át kellett írni az útvonalban a számokat.

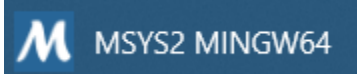
Módosítsuk a “cargo.toml” és “src/lib.rs” fájlokat a következők szerint



Töltsük le az MSYS2-öt, hogy a C és RUST fájl is ugyanazt a gcc buildert használja: <https://www.msys2.org/>

Az oldalon a 6. pontban látható, hogy miután megvagy a telepítéssel érdemes futtatni a:

pacman -S mingw-w64-ucrt-x86_64-gcc parancsot



Indítsd el az MSYS2 MINGW64-et és írd be, hogy:

pacman -S mingw-w64-x86_64-rust

****Lényegében ugyanazokat a crate-eket cargo-kat kell letölteni mint a RUST feltelepítésekor. Bár (!) a tesztprogram írásakor a VSCode is helyesen fordította a rust fájlokat, míg az eredeti demo-nál nem.**

VSCode terminálban futtasd a "rustup target add thumbv7em-none-eabihf" parancsot.

Más mikrovezérlő esetén ellenőrizheted itt a targeted:

<https://docs.rust-embedded.org/book/intro/install.html>

Ellenőrizd a "rustup target list" paranccsal, hogy a targeted (installed) állapotban van-e.

Helyezd a(z) msys2/mingw64/bin-t a környezeti változód (Path) legtetejére.

A ".cargo/bin" legyen a második.

MSYS2 MinGW64 Terminálban jöhet szintén a "rustup target add thumbv7em-none-eabihf"

****Ha panaszkodik, hogy a rustup nem -bash parancs. Másold át a "C:\Users\FELHASZNÁLÓD\.cargo\bin" mappából a rustup alkalmazást az "(Útvonal ahol az msyst telepítetted)\msys2\mingw64\bin" mappába. Ha továbbra sem sikerül a targetet letölteni, talán nem is probléma. Legalábbis ezen dokumentummal párhuzamosan készített program, működött cargo-val is.**

1. Kövesd az embedded book utasításait: <https://docs.rust-embedded.org/book/intro/install/windows.html>

2. Töltsd le a GNU Toolchain-t: <https://developer.arm.com/downloads/-/gnu-rm>

3/A. És az OpenOCD-t is: <https://github.com/xpack-dev-tools/openocd-xpack/releases/>

3/B. Tedd egy mappába a kicsomagolt tartalmat és add hozzá a bin könyvtárat a PATH-hoz.

4. Telepíteni kell az <https://www.st.com/en/development-tools/stsw-link009.html>, vagy az OpenOCD nem fog működni.

5. Teszteléshez írd a terminálba, hogy: "openocd -v"

4. C és RUST összefésülése

Most át fogunk állítani ezt-azt ennek a videónak megfelelően, hogy az eszköz helyesen működjön:

<https://www.youtube.com/watch?v=3JGRt3BFYrM>

Mappa: USB_DEVICE >> Target >> usbd_conf.h fájlban az USBD_CUSTOMHID_OUTREPORT_BUF_SIZE értéke "2U"-ról "64U" legyen, az USBD_CUSTOM_HID_REPORT_DESC_SIZE szintén "2U"-ról, de "33U"-ra.

```
77  /*-----*/
78  #define USBD_CUSTOMHID_OUTREPORT_BUF_SIZE 64U
79  /*-----*/
80  #define USBD_CUSTOM_HID_REPORT_DESC_SIZE 33U
81  /*-----*/
```

usbd_conf.h

Mappa: Middlewares\ST\STM32_USB_Device_Library\Class\CustomHID\Inc >> usbd_customhid.h fájlban a CUSTOM_HID_EPIN_SIZE és CUSTOM_HID_EPOUT_SIZE >> "0x02U"-ról >> "0x40"-re.

```
47  #ifndef CUSTOM_HID_EPIN_SIZE
48  #define CUSTOM_HID_EPIN_SIZE 0x40
49  #endif /* CUSTOM_HID_EPIN_SIZE */
50
51  #ifndef CUSTOM_HID_EPOUT_ADDR
52  #define CUSTOM_HID_EPOUT_ADDR 0x01U
53  #endif /* CUSTOM_HID_EPOUT_ADDR */
54
55  #ifndef CUSTOM_HID_EPOUT_SIZE
56  #define CUSTOM_HID_EPOUT_SIZE 0x40
57  #endif /* CUSTOM_HID_EPOUT_SIZE */
```

usbd_customhid.h

- Ki kell cserélni a 107-es sort is, hogy az OutEvent 2 byte helyett egy buffert várjon. (uint8_t*)

```
107  //int8_t(*OutEvent)(uint8_t event_idx, uint8_t state);
108  int8_t(*OutEvent)(uint8_t*);
```

usbd_customhid.h

- A host általi lekérdezés gyakoriságát is gyorsíthatjuk. Cseréljük ki az alábbi 2 kép által:

```
184  CUSTOM_HID_FS_BINTERVAL, /* bInterval: Polling Interval */
185  /* 34 */
186
187  0x07, /* bLength: Endpoint Descriptor size */
188  USB_DESC_TYPE_ENDPOINT, /* bDescriptorType: */
189  CUSTOM_HID_EPOUT_ADDR, /* bEndpointAddress: Endpoint Address */
190  0x03, /* bmAttributes: Interrupt endpoint */
191  CUSTOM_HID_EPOUT_SIZE, /* wMaxPacketSize: 2 Bytes max */
192  0x00,
193  CUSTOM_HID_FS_BINTERVAL, /* bInterval: Polling Interval */
183  0x00,
184  0x1, /* bInterval: Polling Interval */
185  /* 34 */
186
187  0x07, /* bLength: End
188  USB_DESC_TYPE_ENDPOINT, /* bDescriptorT
189  CUSTOM_HID_EPOUT_ADDR, /* bEndpointAdd
190  0x03, /* bmAttributes
191  CUSTOM_HID_EPOUT_SIZE, /* wMaxPacketSi
192  0x00,
193  0x1, /* bInterval: Polling Interval */
```

usbd_customhid.c

****Változóval volt megadva, hát átírtam 0x1-re, mert máshol is használatban van az érték. Tégy te is így!**

- Igazítsd a forráskódot a header fájlhoz:

usbd_customhid.c

```
633 ((USBD_CUSTOM_HID_ItfTypeDef *)pdev->pUserData[pdev->classId])->OutEvent(hhid->Report_buf[0],
634 hhid->Report_buf[1]);| too many arguments in function call
685 ((USBD_CUSTOM_HID_ItfTypeDef *)pdev->pUserData[pdev->classId])->OutEvent(hhid->Report_buf[0],
686 hhid->Report_buf[1]);| too many arguments in function call
687 hhid->IsReportAvailable = 0U;
632 //NAKED till the end of the application processing /
633 ((USBD_CUSTOM_HID_ItfTypeDef *)pdev->pUserData[pdev->classId])->OutEvent(hhid->Report_buf);
```

- Illessz be egy buffer-t az usbd_custom_hid_if.c fájl ~65. sorába!

```
C usbd_custom_hid_if.c U X usbd_custom_hid_if.c
USB_DEVICE > App > C usbd_custom_hid_if.c > [0] buffer
64 /* USER CODE BEGIN PRIVATE_DEFINES */
65 uint8_t buffer[0x40];|
66 /* USER CODE END PRIVATE_DEFINES */
```

- CUSTOM_HID_OutEvent_FS függvény paramétereit is módosítsd!

****Én pl. ami nem kellett azt kikommenteztem, és beszúrtam egy memória másolás függvényt is. (!)**

```
126 static int8_t CUSTOM_HID_OutEvent_FS(void);
127 static int8_t CUSTOM_HID_OutEvent_FS(uint8_t *state);
128
static int8_t CUSTOM_HID_OutEvent_FS(uint8_t *state)
{
    /* USER CODE BEGIN 6 */
    // UNUSED(event_idx);
    memcpy(buffer, state, 0x40);
    UNUSED(state);
}
```

- A reportdesc-et is ki kell bővíteni, hogy a PC felismerje USB HID eszközként. Alább egy táblázatban találsz a kódrészletet:





usbd_custom_hid_if.c

```
90 /** Usb HID report descriptor. */
91 ALIGN_BEGIN static uint8_t CUSTOM_HID_ReportDesc_FS[USBD_CUSTOM_HID_REPORT_DESC_SIZE]
92 {
93     /* USER CODE BEGIN 0 */
94     0x00,
95     /* USER CODE END 0 */
96     0xC0 /* END_COLLECTION */
97 };
98
```


<pre> 91 ALIGN_BEGIN static uint8_t CUSTOM_HID_ReportDesc_FS 92 { 93 /* USER CODE BEGIN 0 */ 94 0x06, 0x00, 0xff, // Usage Page(Undefined) 95 0x09, 0x01, // USAGE(Undefined) 96 0xa1, 0x01, // COLLECTION (Application) 97 0x15, 0x00, // LOGICAL_MINIMUM(0) 98 0x26, 0xff, 0x00, // LOGICAL_MAXIMUM(255) 99 0x75, 0x08, // REPORT_SIZE(8) 100 0x95, 0x40, // REPORT_COUNT(64) 101 0x09, 0x01, // USAGE (Undefined) 102 0x81, 0x02, // INPUT (Data,Var,Abs) 103 0x95, 0x40, // REPORT_COUNT(64) 104 0x09, 0x01, // USAGE (Undefined) 105 0x91, 0x02, // OUTPUT (Data,Var,Abs) 106 0x95, 0x01, // REPORT_COUNT(1) 107 0x09, 0x01, // USAGE (Undefined) 108 0xb1, 0x02, // FEATURE (Data,Var,Abs) 109 /* USER CODE END 0 */ 110 0xc0 /* END_COLLECTION */ 111 } </pre>	<pre> /* USER CODE BEGIN 0 */ 0x06, 0x00, 0xff, // Usage Page(Undefined) 0x09, 0x01, // USAGE(Undefined) 0xa1, 0x01, // COLLECTION (Application) 0x15, 0x00, // LOGICAL_MINIMUM(0) 0x26, 0xff, 0x00, // LOGICAL_MAXIMUM(255) 0x75, 0x08, // REPORT_SIZE(8) 0x95, 0x40, // REPORT_COUNT(64) 0x09, 0x01, // USAGE (Undefined) 0x81, 0x02, // INPUT (Data,Var,Abs) 0x95, 0x40, // REPORT_COUNT(64) 0x09, 0x01, // USAGE (Undefined) 0x91, 0x02, // OUTPUT (Data,Var,Abs) 0x95, 0x01, // REPORT_COUNT(1) 0x09, 0x01, // USAGE (Undefined) 0xb1, 0x02, // FEATURE (Data,Var,Abs) /* USER CODE END 0 */ 0xc0 /* END_COLLECTION */ </pre>
---	---

Létrehoztam 2 új fájlt .h és .c singleton dataprovidert.

Tedd meg te is!

	Cargo.toml	U
	dataprovider.c	U
	dataprovider.h	U
	Makefile	U

Header: dataprovider.h	Source: dataprovider.c
<pre> #ifndef DATAPROVIDER_H #define DATAPROVIDER_H struct dataprovider { int speed; }; // Az adatszerkezet globális példányának elérése struct dataprovider *getDataprovider(); // Egyetlen példány létrehozása és inicializálása void initDataprovider(); int getSpeed(); void setSpeed(int value); #endif </pre>	<pre> #include "dataprovider.h" static struct dataprovider instance; struct dataprovider *getDataprovider() { return &instance; } void initDataprovider() { instance.speed = 250; } int getSpeed() { struct dataprovider *dp = getDataprovider(); return dp->speed; } void setSpeed(int value) { struct dataprovider *dp = getDataprovider(); dp->speed = value; } </pre>

main.c -ben includeold, továbbá hozd létre a program indulásakor!

```
23  /* Private includes ----- main.c
24  /* USER CODE BEGIN Includes /
25  #include ".././dataproducer.h"
26  /* USER CODE END Includes */
27
```

```
64  */
65  int main(void)
66  {
67      /* USER CODE BEGIN 1 */
68      initDataprovider();
69      /* USER CODE END 1 */
70
```

Ezek után jöhet egy pofon egyszerű kód: megírunk egy led blinky programot:

```
94  /* Infinite loop */
95  /* USER CODE BEGIN WHILE */
96  while (1)
97  {
98      /* USER CODE END WHILE */
99      HAL_Delay(getSpeed());
100     HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
101     /* USER CODE BEGIN 3 */
102 }
103 /* USER CODE END 3 */
```

Írunk egy egyszerű RUST függvényt, ami ha ASCII 1-2-3 karakterekre különböző sebességet ad vissza.

```
#![allow(non_snake_case)]
#![no_std]

extern crate cortex_m_rt;
extern crate panic_halt;

#[no_mangle]
pub extern "C" fn get_speed_from_rust(buffer: &mut [u8; 0x40]) -> u8 {
    match buffer[1] {
        49 => 64, // ASCII 1
        50 => 128, // ASCII 2
        51 => 255, // ASCII 3
        _ => 250,
    }
}
```

Ha megvan letöltjük a cbindgen-t: “cargo install --force cbindgen” paranccsal.

Dokumentációt itt olvashatsz róla: (<https://github.com/mozilla/cbindgen>)

C kód generálásához a következő parancsot hívtam meg:

“cbindgen --crate Balluff_USB_HID --output rust_wrapper.h --lang c”

```
> cbindgen --crate Balluff_USB_HID --output rust_wrapper.h --lang c
```

Includeolni kell őket

▼ USB_DEVICE	●	21	/* Includes -	usb_custom_hid_if.c
▼ App	●	22	#include "usb_custom_hid_if.h"	
C usb_device.c	U	23		
C usb_device.h	U	24	/* USER CODE BEGIN INCLUDE */	
C usb_custom_hid_if.c	U	25	#include "../..../dataprovder.h"	
C usb_custom_hid_if.h	U	26	#include "../..../rust_wrapper.h"	
C usb_desc.c	U	27	/* USER CODE END INCLUDE */	
		28		

Szükség van továbbá a dataprovder példány létrehozására.

```
34  /* Private variables -----
35  struct dataprovder Dataprovder;|
36  /* USER CODE END PV */
```

Kiegészítjük lejjebb a dolgokat a képernyőmetszés alapján: Visszakapjuk a sebességet.
Beállítjuk a dataprovdernek. Majd visszaküld egy ASCII "ok"-t.

```
190  static int8_t CUSTOM_HID_OutEvent_FS(uint8_t *state)
191  {
192      /* USER CODE BEGIN 6 */
193      // UNUSED(event_idx);
194      memcpy(buffer, state, 0x40);
195      uint8_t num = get_speed_from_rust(&buffer);
196      setSpeed(num);
197
198      memset(buffer, 0, 0x40);
199      buffer[1] = 79; // O
200      buffer[2] = 75; // K
201      USB_CUSTOM_HID_SendReport(&hUsbDeviceFS, buffer, 0x40);
202  }
```

(!) Jöhet a makefile:

Makefile

```
8 # ChangeLog :
9 | 2023-08-28 - Extended with RUST files by SzaboBenyo
10 # 2017-02-10 - Several enhancements + project update mode
11 # 2015-07-22 - first version
12 # -----
```

1. changelog

```
69 Middleware/ST/STM32_USB_Device_Library/Class/CustomHID/Src/usbd_customhid.c \
70 dataprovider.c
```

2. C forrásfájlok kibővítése

```
155 # link script
156 # link script
157 LDSCRIPT = STM32F412ZGTx_FLASH.ld
158
159 # libraries
160 LIBS = -lc -lm -lnosys
161 LIBDIR =
162 LDFLAGS = $(MCU) -specs=nano.specs -T$(LDSCRIPT) $(LIBDIR) $(
163
164 RUST_PROJECT_PATH = ./target/thumbv7em-none-eabihf/release
165
166 RUST_PROJECT_NAME = libBalluff_USB_HID
167
168 # default action: build all
169 all: $(BUILD_DIR)/$(TARGET).elf $(BUILD_DIR)/$(TARGET).hex $(
```

3. A Flag-ek kibővítése RUST-val

```
167
168 # default action: build all
169 all: $(BUILD_DIR)/$(TARGET).elf $(BUILD_DIR)/$(TARGET).hex $(BUILD_DIR)/$(TARGET).bin $(RUST_PROJECT_PATH)/$(RUST_PROJECT_NAME).a
170
```

4. Buildeléskor a RUST könyvtárat is szeretnénk fordítani

```
171 # BUILD RUST PROJECT FOR STM32F412ZG MCU LIKE THIS --> //cargo build --target thumbv7em-none-eabihf --release
172 $(RUST_PROJECT_PATH)/$(RUST_PROJECT_NAME):
173 cargo build --target thumbv7em-none-eabihf --release
174
```

5. Ha nincs ilyen, akkor így kell megcsinálni

```
191
192 $(BUILD_DIR)/$(TARGET).elf: $(OBJECTS) Makefile
193 $(CC) $(OBJECTS) $(LDFLAGS) $(RUST_PROJECT_PATH)/$(RUST_PROJECT_NAME).a -o $@
194 $(SZ) $@
```

6. Bővítés RUST-val

Tesztelés... Sikeres volt ekkor: 2023.09.20 13:31

5. PC oldali C# app létrehozása Visual Studio – WinFormApp .NET extensions + HIDSharp

Training materials név alatt megtalálhatóak az STM USB kurzushoz tartozó fájlok:

https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/STM32-USB-training.html

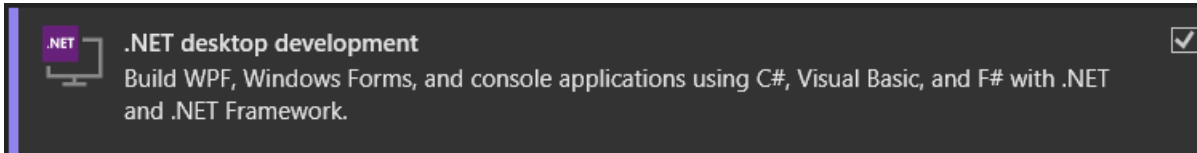
Ez ugyan az, csak egyből a Google Drive-hoz férhetsz hozzá:

https://drive.google.com/file/d/1sjU9iNvh_khDZHDM9Qau03PGKwMd4u7U/view

Vagy tőlem is letöltheted, a Debug mappa tartalmazza:

https://github.com/szabobenyo/USB_HID_C_and_RUST

De ha szeretnél egy saját WindowsFormApp-ot készíteni telepítsd ezt:



A program a <https://github.com/IntergatedCircuits/HidSharp> könyvtárat használja. Pofon egyszerű.

Így néz ki a kód:

```
using System;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using HidSharp;

namespace HID_terminal
{
    3 references
    public partial class Form1 : Form
    {
        HidDeviceLoader loader;
        HidDevice device;
        HidStream stream;
        byte []bytes;
        1 reference
        public Form1()
        {
            InitializeComponent();
            loader = new HidDeviceLoader();
            device = loader.GetDevices(1155, 22352).FirstOrDefault();
            if (device == null)
            {
                Console.WriteLine("Failed to open device.");
                MessageBox.Show("Failed to open device.", "Error",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                Environment.Exit(1);
            }
        }
    }
}
```

```

private void button_send_Click(object sender, EventArgs e)
{
    if (!device.TryOpen(out stream))
    {
        Console.WriteLine("Failed to open device."); MessageBox.Show("Failed to open device.", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Error); Environment.Exit(1);
    }

    using (stream)
    {
        int n = 0;

        bytes = new byte[device.MaxInputReportLength];
        int count=0;
        var message = new byte[64];
        ASCIIEncoding.ASCII.GetBytes(textBoxSend.Text, 0, textBoxSend.Text.Length, message, 2);
        message[0] = 0;
        message[1] = (byte)textBoxSend.Text.Length;
        stream.Write(message, 0, 2+textBoxSend.Text.Length);
        try
        {
            count = stream.Read(bytes);
        }
        catch (TimeoutException)
        {
            Console.WriteLine("Read timed out.");
        }

        if (count > 0)
        {
            this.BeginInvoke(new EventHandler(DoUpdate));
        }
    }
}

```

```

1 reference
private void DoUpdate(object sender, System.EventArgs e)
{
    string s = Encoding.UTF8.GetString(bytes, 2, 2+bytes[1]);
    textBoxRecieved.AppendText(s);
    textBoxRecieved.AppendText("\n");
}

```

Maga a form app:

The screenshot shows a Windows application window titled "HID terminal". The window contains a user interface with two main sections. On the left, under the label "Data send", there is a text input field and a "Send" button below it. On the right, under the label "Received data", there is a large, empty rectangular area intended for displaying received data.

Források:

Visual Studio Code-ot -> <https://code.visualstudio.com/download>

RUST telepítőjét -> <https://www.rust-lang.org/tools/install>

Rust YT video 2: <https://www.youtube.com/watch?v=BU1LYFkpJuk>

Rust YT video 1: <https://www.youtube.com/watch?v=yo4kWLtSPCY>

cubeMX: <https://www.st.com/en/development-tools/stm32cubemx.html>

Makefile toolchain: <https://www.youtube.com/watch?v=ZP2fd2qatj0>

MSYS2: <https://www.msys2.org/>

Targetek listája: <https://docs.rust-embedded.org/book/intro/install.html>

RUST telepítése: <https://docs.rust-embedded.org/book/intro/install.html>

Embedded book: <https://docs.rust-embedded.org/book/intro/install/windows.html>

GNU Toolchain-t: <https://developer.arm.com/downloads/-/gnu-rm>

OpenOCD-t is: <https://github.com/xpack-dev-tools/openocd-xpack/releases/>

MCU-hoz OpenOCD szükséglet: <https://www.st.com/en/development-tools/stsw-link009.html>

Custom USB HID course video: <https://www.youtube.com/watch?v=3JGRt3BFYrM>

cbindgen -> <https://github.com/mozilla/cbindgen>

Training materials -> https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-moocs/STM32-USB-training.html

Training materials google drive ->

https://drive.google.com/file/d/1sjU9iNvh_khDZHDM9Qau03PGKwMd4u7U/view