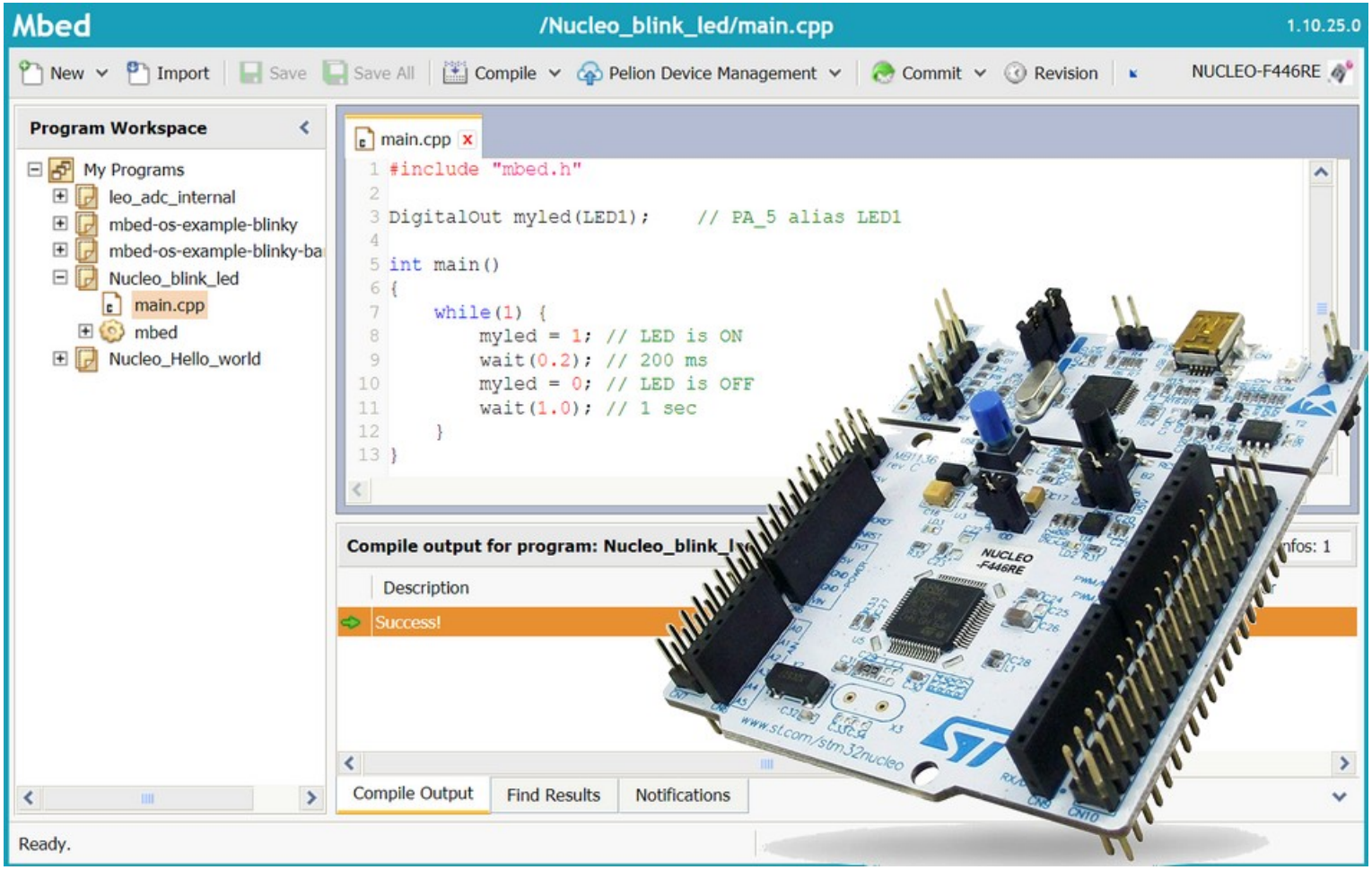
STM32 mikrovezérlők programozása ARM mbed környezetben



10. USB eszközök

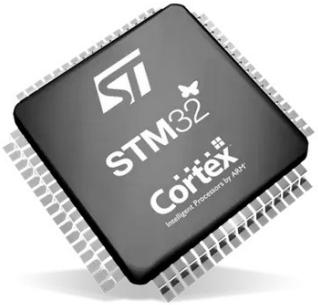
Felhasznált és ajánlott irodalom

- Cserny István: **A FRDM-KL25Z kártya programozása mbed környezetben**
  Rob Toulson, Tim Wilmhurst:
  Fast and Effective Embedded Systems Design: Applying the ARM mbed
- Perry Xiao: Designing Embedded Systems and the IoT eith ARM mbed
- Dogan Ibrahim: ARM-based Microcontroller Projects Using mbed

- **ARM mbed honlap: https://os.mbed.com/**
  - ARM mbed Compiler: https://ide.mbed.com/compiler/
  - ARM mbed 2 Handbook: **https://os.mbed.com/handbook/RTOS**
  - ARM mbed forráskód: https://github.com/ARMmbed/mbed-os

Adatlapok:

- **USB-IF:** USB 2.0 specification
- **USB-IF:** USB HID Usage Tables
- **STM32F446RE** adatlap és termékinfo
- **STM32F446** Family Reference Manual



Példaprogramok

Lab10_USB_Serial – egyszerű kiíratás

Lab10_USBHID_demo – kétirányú üzenetküldés

LAB10_USBHID_pnp – kétirányú üzenetküldés

LAB10_USB_keyboard – billentyűzet emuláció

LAB10_USB_kvikbord – billentyűparancs küldés

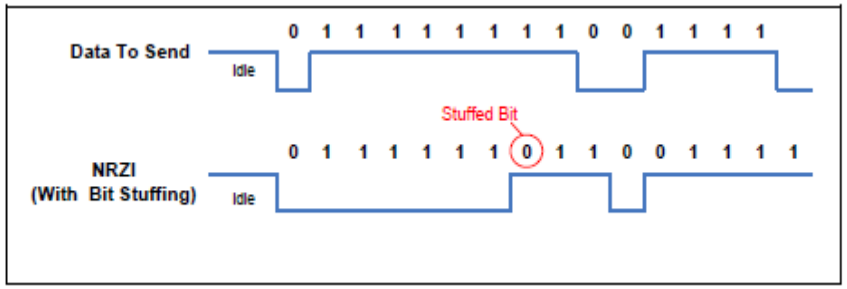
A mintaprogramok az os.mbed.com/users/cspista/code/ oldalon is megtalálhatók

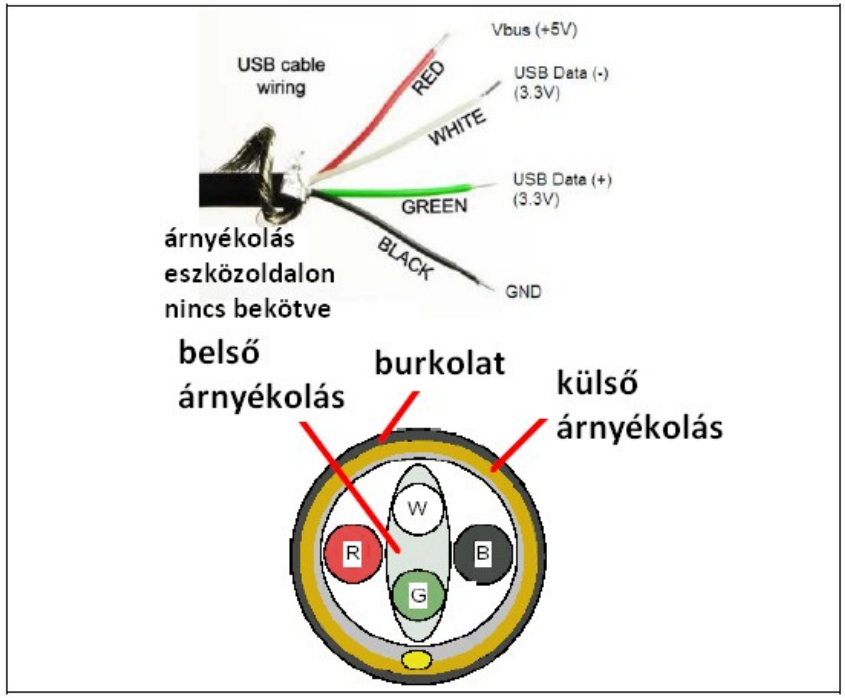
Az USB-ről röviden

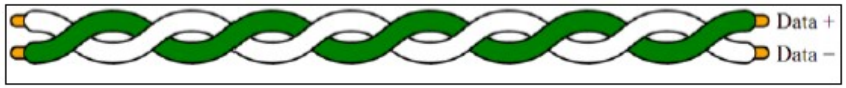
- USB = Univerzális soros busz (Universal Serial Bus)
- Szabvány leírása: www.usb.org
  - **USB 1.0** - 1995 1,5 Mbit/s
  - **USB 2.0** - 2000 1,5/12/480 Mbit/s (Full speed = 12 Mbit/s)
  - **USB 3.0** - 2009 super speed: 5 Gbps (további 2 érpár felhasználásával)
- **Minden eszköznek jeleznie kell, milyen sebességű átvitelre képes:**
  - Az alacsony sebességű eszközök a **D-** vonalat húzzák fel.
  - A teljes sebességű eszközök a **D+** vonalat húzzák fel 3,3 V-ra egy 1,5 kΩ-os ellenállással
  - A nagysebességű eszközök induláskor teljes sebességű eszközként azonosítják magukat, s később, a host-tal történő egyeztetés után kapcsolnak nagysebességű üzemmódba

Az USB fizikai megvalósítása

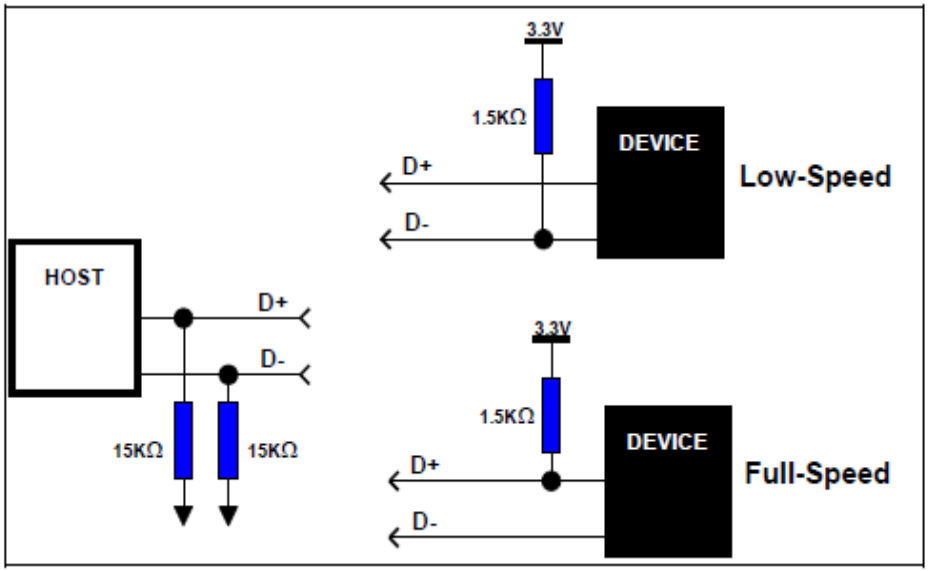
NRZI (Non Return to Zero Inverted) – a jel polaritást vált, ha az adatbit 0.
Bit stuffing: Egy 0 bit beszúrása, ha több, mint 6 db 1-es jön egymás után.

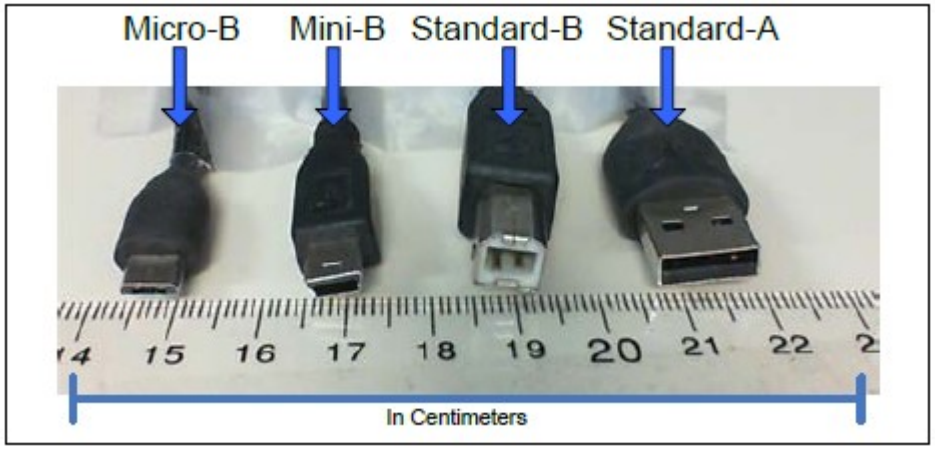




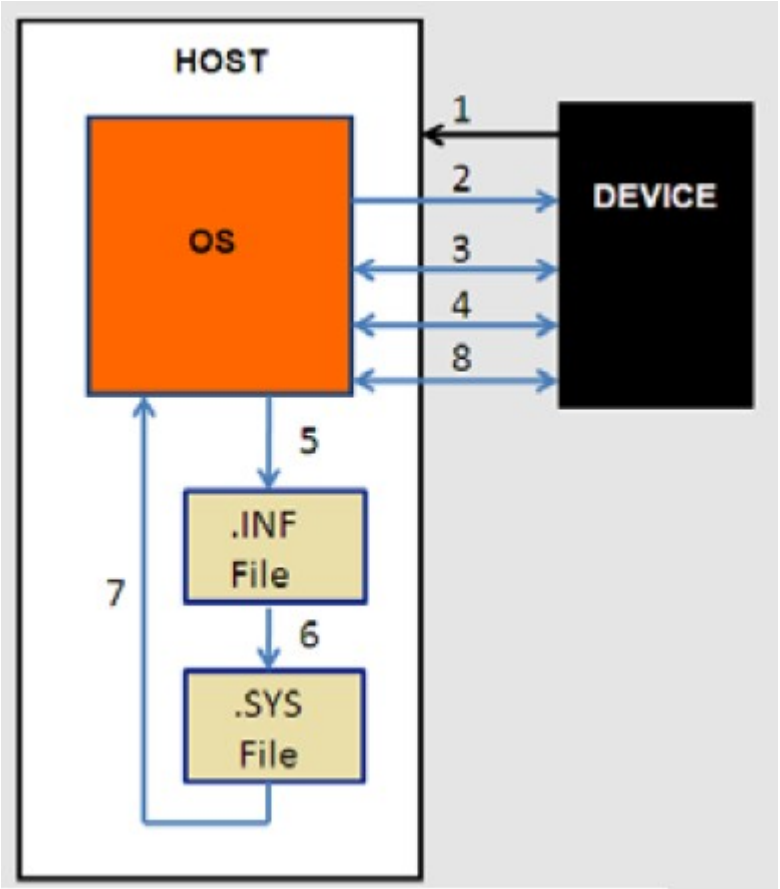


Forrás: Cypress AN57294





USB számbavétel (enumeration)



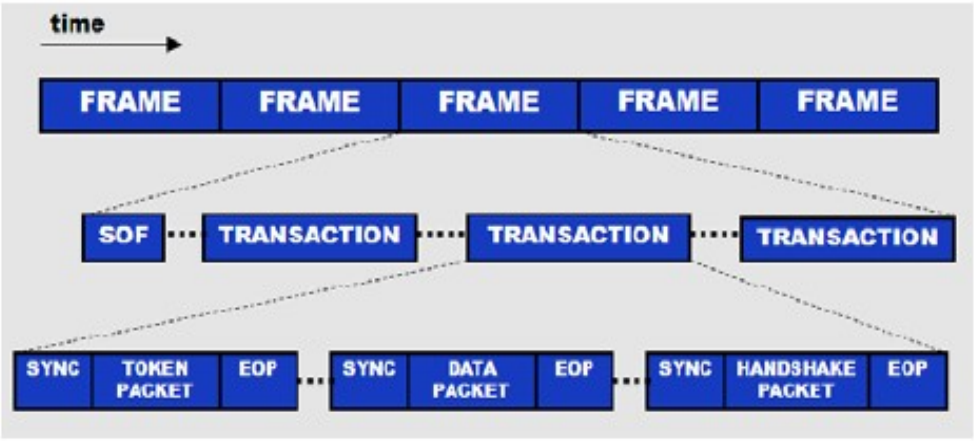
1. Az eszköz csatlakozik a gazdagéphez
2. A gazdagép *reset* parancsot küld az eszköznek
3. Az eszköz válaszol a kérésre és a gazdagép új címet állít be az eszköznek.
4. A gazdagép eszközleírót kér az új címmel azonosított eszköztől. Az eszköz válaszol
5. A gazdagép megkeresi és olvassa az .INF állományt
6. Az .INF specifikálja az eszköz meghajtóját
7. Az eszközmeghajtó betöltése
8. A gazdagép kiválasztja az eszköz konfigurációját.

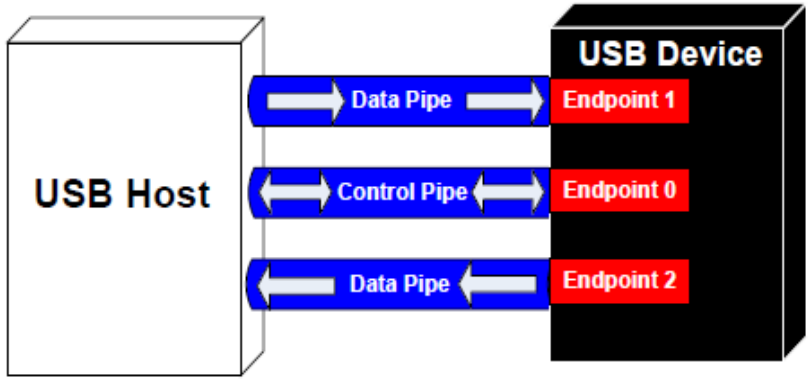
Az eszköz konfigurált és kész a használatra

Az USB termékek azonosítására szolgáló VID, PID számpárost az eszköz gyártója adja meg.
VID = gyártó azonosítója (az USB-IF-től vásárolható), **PID** = termékazonosító (gyártó adja)

USB kommunikációs protokoll

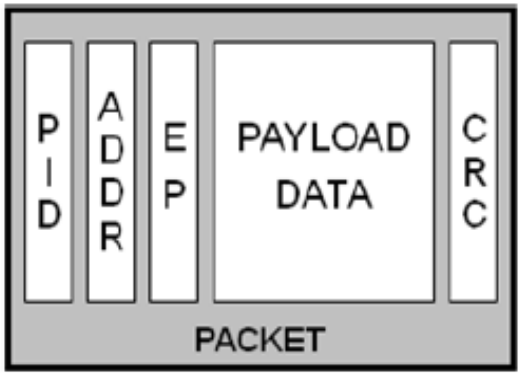
A kommunikáció **keretekre** van osztva (1 keret = 1 ms), a kereten belül **tranzakciók** zajlanak.
A kommunikáció a **host** és az általa megcímzett **végpont** között történik.
Minden eszközben kell egy **vezérlő végpontnak** is lennie (ez a nulladik sorszámú).





Adatcsomag felépítése:

- **Packet ID** (4 bit) lehet pl. IN, OUT, SET, SOF
- **Eszköz cím** (7 bit)
- **Végpont cím** (4 bit: max. 16 IN és max. 16 OUT)
- **Adatblokk** (max. 64/1024 bájt)
- **CRC** ellenőrző kód (5/16 bit)



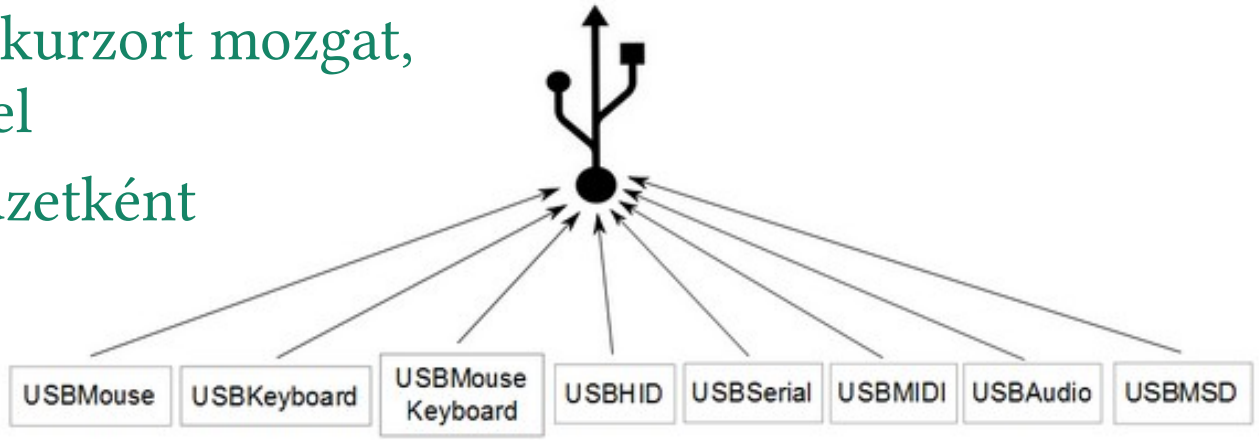
Végpont/kommunikációs osztályok

Az USB négy alapvető adatátviteli típust támogat:

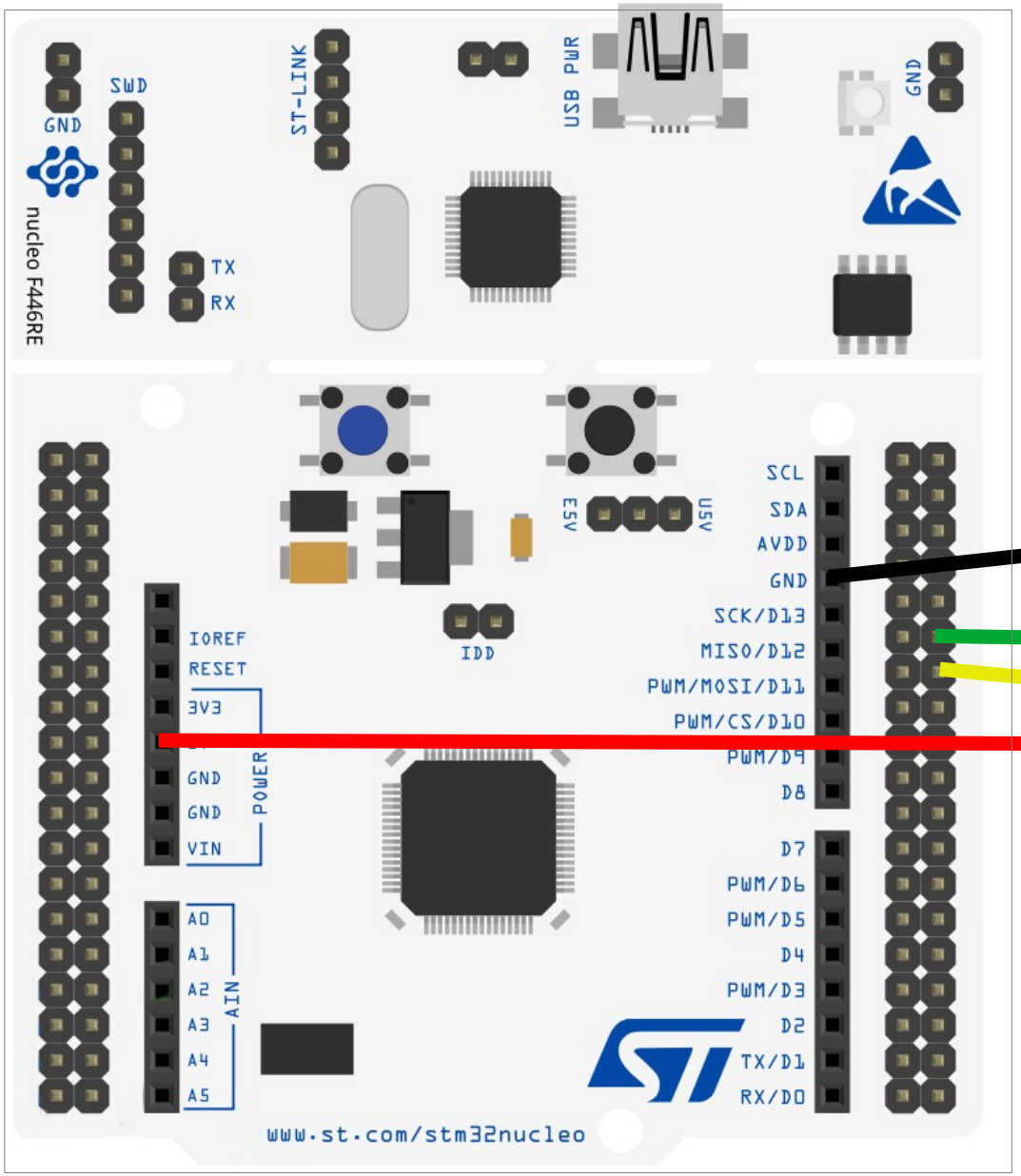
- **Vezérlő** (control): minden eszköznek rendelkeznie kell egy vezérlő típusú végponttal. Ezen keresztül zajlik az USB számbavétel (enumeration) folyamata
- **Ömlesztett** (bulk): nagy mennyiségű, de nem időkritikus adat mozgatására szolgál. Ilyen pl. a **CDC** kommunikációs eszközosztály (Communication Device Class)
- **Megszakításos** (interrupt): kis késleltetésű átvitelt garantál. Pl. szabályos időnként küldendő adatot szokás mozgatni vele (**HID** = Human Interface Devices, pl. billentyűzet, egér)
- **Valós idejű** (isochronus): időkritikus folyamatos átvitelhez, mint pl. a hang-, és videó átvitel
- A mai előadás mintapéldáinál a **CDC** és a **HID** eszközosztályokkal fogunk találkozni.

Az USBDevice programkönyvtár

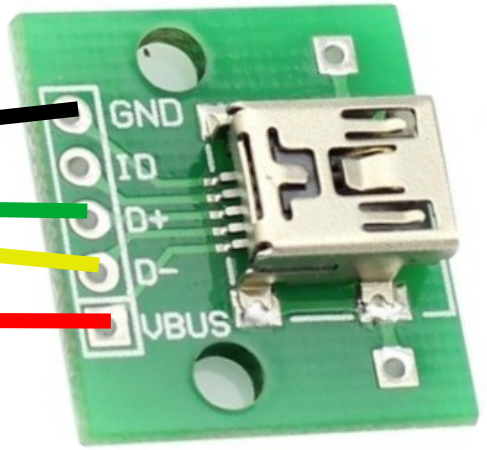
- **Az mbed kompatibilis mikrovezérlő, mint USB eszköz:**
- **USBMouse** – egér, amely kurzort mozgat, kattintást és görgetést kezel
- **USBKeyboard** – billentyűzetként kezeli a mikrovezérlőt
- **USBHID** – adatküldés és fogadás nyers adatokkal
- **USBSerial** – virtuális soros port
- **USBMIDI** – MIDI eszközként használja a mikrovezérlőt
- **USBAudio** – hangeszközként (forrás vagy lejátszó) kezeli a mikrovezérlőt
- **USBMSD** – tömegtároló eszközosztály (mint pl. az USB pendrive vagy a kártyaolvasó)



Az USB csatlakozó bekötése



PA12 USB D+
PA11 USB D-



Az USBSerial objektumosztály

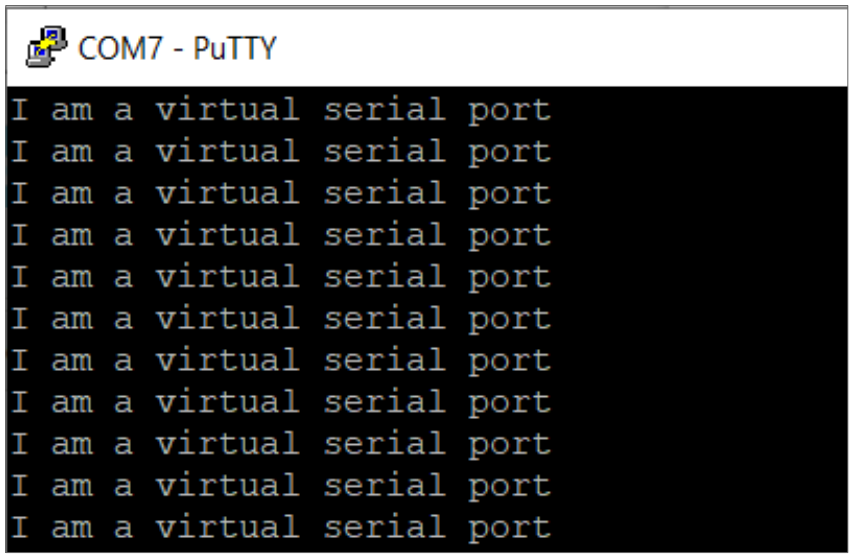
- USB *"bulk transfer"* kommunikáció, a **CDC** (Character Device Class) eszközosztályba tartozik. A PC felől virtuális soros portnak látszik
- Az **USBSerial** objektumosztályt az **USBDevice** programkönyvtár része.

Függvénynév	**Rövid leírás**
USBSerial név (vid=0x1f00, pid=0x2012, release=0x0100, blocking=true)	Példányosítja és inicializálja az **USBSerial** objektumosztályt, s az opcionális paraméterek szerint módosítja a konfigurációt
_putc(int c)	Egy karakter küldése a host felé, blokkoló típusú várakozással
_getc()	Egy karakter fogadása a host felől, blokkoló típusú várakozással.
writeBlock(*buf, size)	Adatblokk kiírása (max. 64 karakter)
puts(s)	Szövegkonstans karakterfüzér kiíratása
gets(s,n)	n db karakter fogadása az s tömbbe
printf()	Formázott szöveg kiíratása
scanf()	Formázott szöveg beolvasása
available()	Beérkezett karakterek száma
writeable()	Van-e szabad hely a kimeneti tárban?
attach(fptr)	Visszahívási függvény rendelése érkező csomagokhoz

Lab10_USB_Serial/main.cpp

- A legegyszerűbb mintapélda másodpercenként kiír egy sort

```
#include "mbed.h"         // Ehhez importáltuk az mbed 2 csomagot
#include "USBSerial.h"    // Ehhez importáltuk az USBDevice csomagot

USBSerial serial;         //Virtual serial port over USB
int main(void) {
    while(1) {
        serial.printf("I am a virtual serial port\r\n");
        wait(1);
    }
}
```



Az **USBSerial használatánál** a PC oldalon **nem kell beállítani az adatsebességet,** mert az USB virtuális soros port esetében ennek nincs szerepe

Az adatátvitel nem karakterenként, hanem keretezett USB csomagonként történik (bulk transfer)

Az USB HID eszközosztály

- A **HID** (Human Interface Device) talán a legjobban támogatott eszközosztály a szabványosított USB eszközosztályok közül. Ide tartozik az USB egér, billentyűzet, botkormány, távirányító és sok más eszköz
- **Tulajdonságok:**
  - Minden tranzakció vezérlő vagy megszakítás típusú átvitelt használ
  - Tranzakciónként legfeljebb 64 bájt vihető át
  - A maximális átviteli sebesség 1 tranzakció/ms, ami átszámítva legfeljebb 64 KB/s.
  - Csak egy kiviteli és egy beviteli végpont használható (a vezérlő csatorna mellett)
  - A gazdagép periodikusan kérdezi le a **HID** eszközt.

Általános (generic) HID eszközök

- A szabványos **HID** eszközöknél az ún. **HID report descriptor** (HID jelentés leíró tábla) megmondja, hogy az átvitt adatcsomag egyes bájtjai (vagy bitjei) mit jelentenek.
- Az általános (generic) **HID** eszközöknél a **HID report descriptor** csak az átvitt adatok mennyiségét mondja meg, azok értelmezése a mikrovezérlőbe töltött firmware-re és a PC-n futó alkalmazásra van bízva. Ebben az esetben tehát csak az átvitel mikéntje szabványos, az adatok felhasználása azonban gyártóspecifikus.
- A **HID** átvitel használatának előnyei:
  - Nem kell gyártóspecifikus meghajtó
    (az operációs rendszer eleve „tudja” kezelni az eszközt)
  - Garantált átviteli időzítés

Az USBHID objektumosztály

- Az **USBHID** osztály az előzőekben említett *"Általános HID Eszközt"* (Generic HID Device) valósítja meg. A mikrovezérlő és a számítógép között saját protokoll szerint kommunikálhatunk

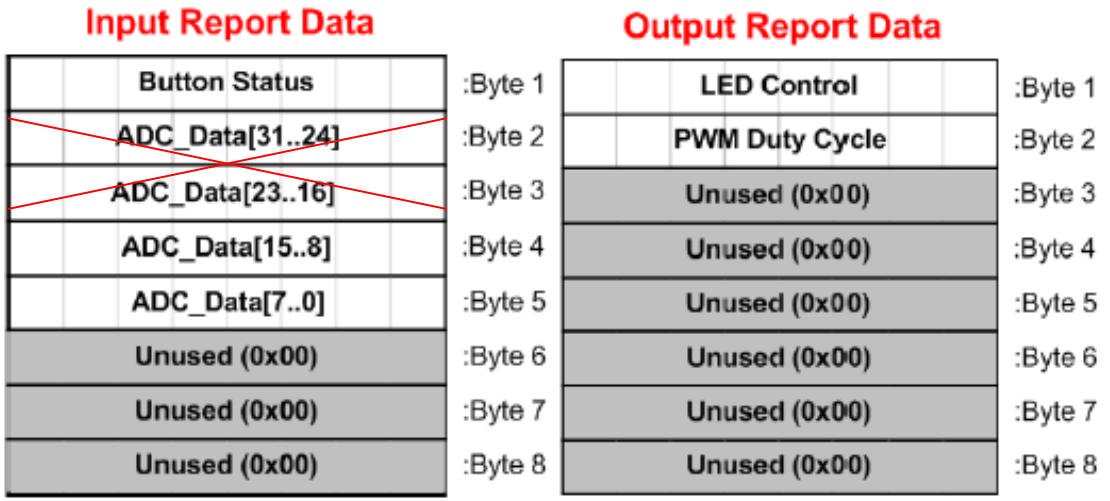
Függvénynév	Rövid leírás
USBHID név(output_report_length=64, input_report_length=64, vendor_id=0x1234, product_id=0x0006, product_release=0x0001, connect=true)	Konstruktor függvény. Példányosítja és inicializálja az **USBHID** objektumosztályt, megadja a kimenő és bejövő adatblokk méretét, a VID/PID azonosítót, s a **connect** paraméter értékétől függően kapcsolódik (vagy nem)
read(HID_REPORT* report)	Egy adatblokk fogadása blokkoló várakozássa
readNB(HID_REPORT* report)	Egy adatblokk fogadása nem blokkoló várakozással
send(HID_REPORT* report)	Egy adatblokk küldése blokkoló várakozással
sendNB(HID_REPORT* report)	Egy adatblokk küldése nem blokkoló módon

- A **számítógépen kell olyan alkalmazás,** amely képes kezelni az **USBHID** adatforgalmat. Ez lehet akár egy szkript is, amihez használhatjuk a Python értelmező **pywinusb** kiegészítését. Egy másik lehetőség a **hidapi** könyvtár, amelyet C/C++ konzol alkalmazásokból vagy grafikus alkalmazásokból is használhatunk. Az **USBHID bindings** és az **USBHID C bindigs** oldalon találunk mintapéldákat mindkettő használatára.

Lab10_USBHID_demo

- Ezt a programot a Cypress AN82072 alkalmazási mintapéldájához igazítva készítettük el, hogy annak a letölthető grafikus PC alkalmazását használni tudjuk. Ennek megfelelően mindkét irányba 8 bájtos csomagokat küldünk
- Az Input/Output irány a gazdagép szempontjából értendő
- **Input Report:** a mikrovezérlő küldi, első bájtja a **B1** gomb bemenet állapota, a következő 4 bájt az **A0** analóg csatorna ADC konverziójának eredménye (kössünk ide egy potmétert, vagy egy analóg szenzort)
- **Output Report:** (a PC küldi) vezérlési funkciót lát el: az első bájt a beépített LED-et vezérli (0: ki, 1: be), a második bájt pedig egy, a **D3** kimenetre kötött LED-et vezérel (1 - 100 közötti érték lehet, ami a PWM jel százalékos kitöltését adja meg).

A 8 bájtos adatcsomagokat csak a firmware és a PC alkalmazás fogja tudni értelmezni! Az ADC adat nálunk 16 bites...



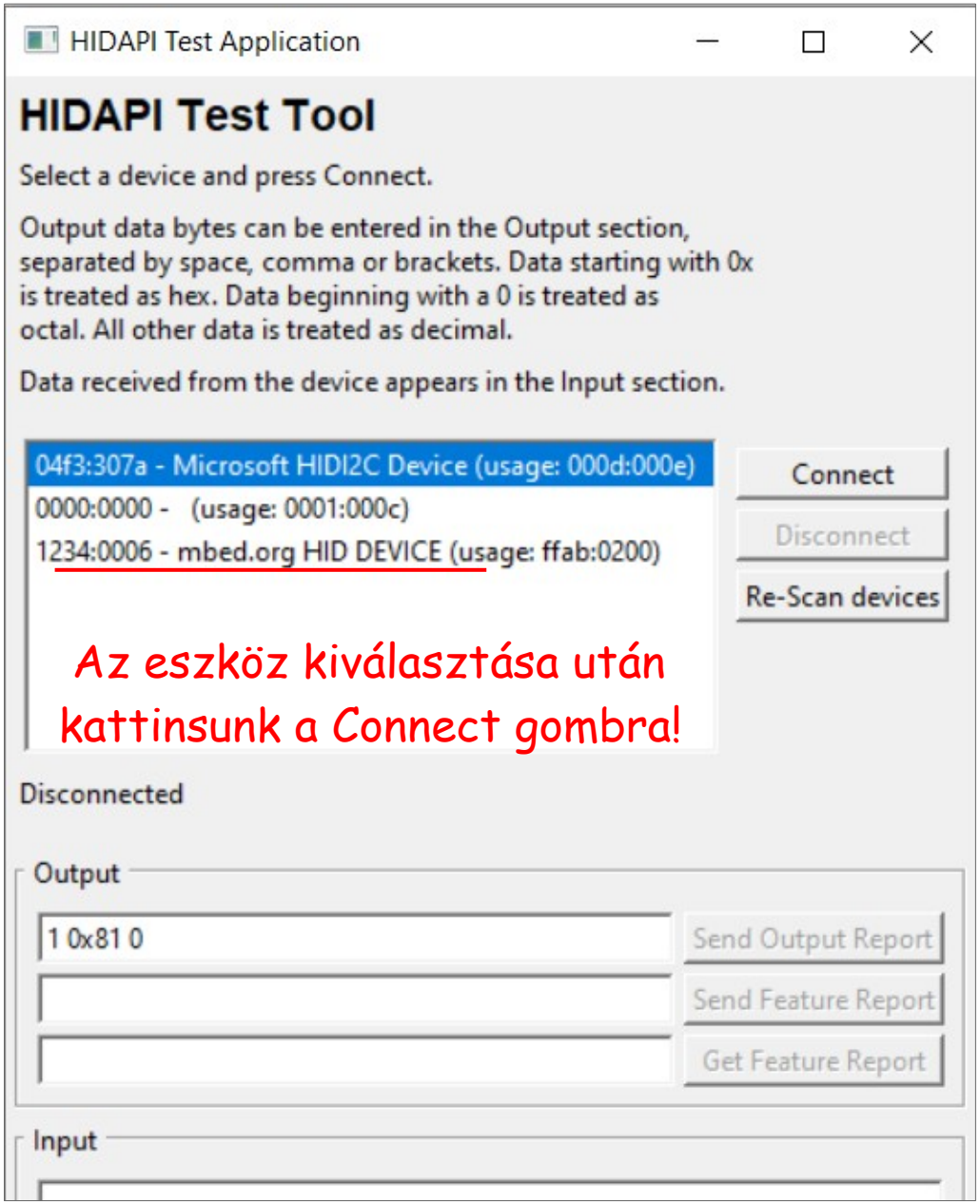
Lab10_USBHID_demo/main.cpp

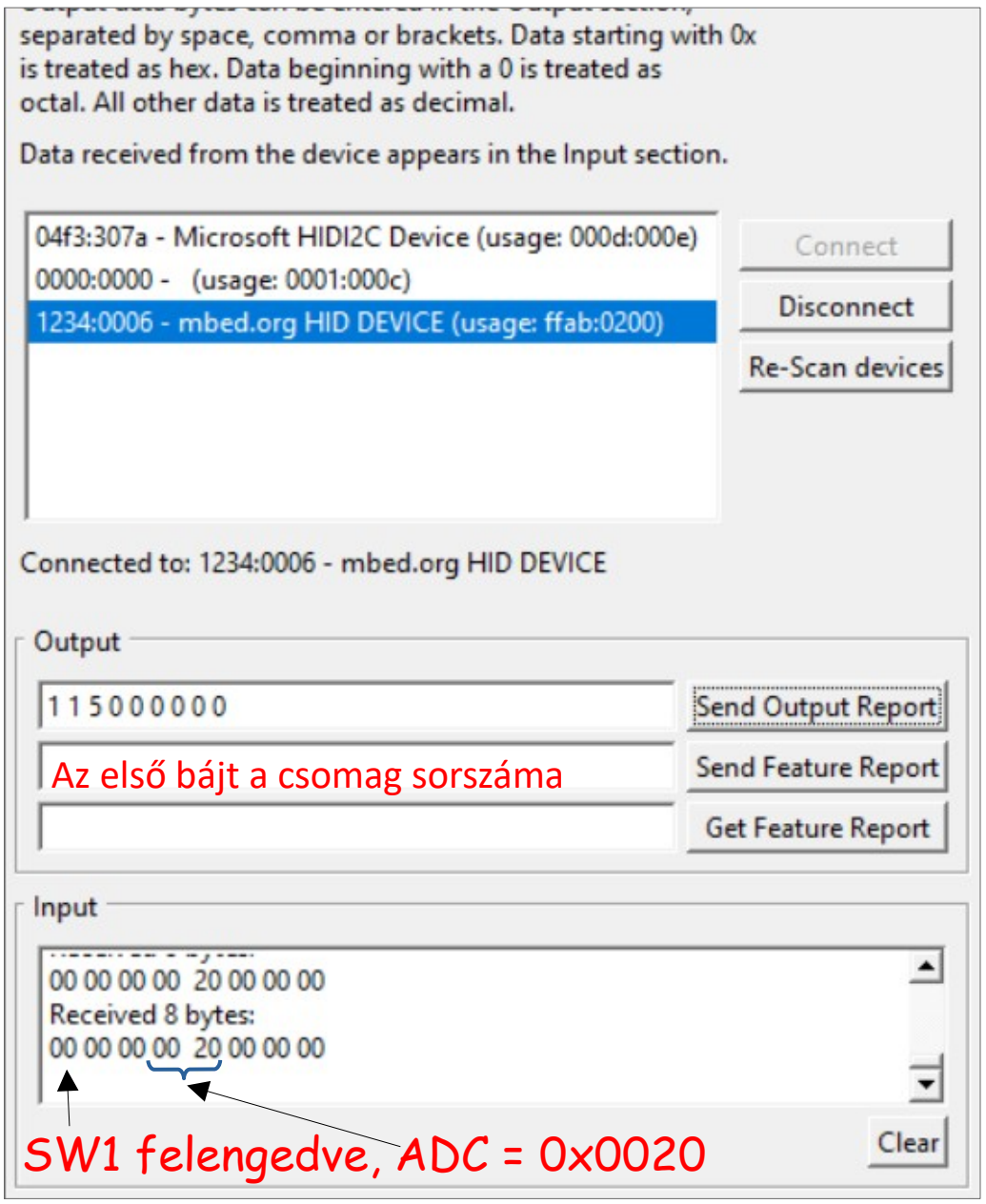
```
#include "mbed.h"
#include "USBHID.h"
USBHID hid(8, 8);                   //Record size 8 bytes,Vid:0x1234,Pid:0x0006
HID_REPORT send_report;             //This report will contain data to be sent
HID_REPORT recv_report;             //This report will contain data received
DigitalOut LED_1(LED1);             //Buitin LED at PA5
PwmOut LED_2(D3);                   //External LED at D3 (PB3)
DigitalIn SW1(BUTTON1,PullUp); //Builtin pushbutton at PC13
AnalogIn adc(A0);                   //Analog input at A0 (PA0)
int main(void) {
    send_report.length = 8;
    LED_1 = 0;
    LED_2.period_ms(20);
    while (1) {
        uint16_t raw = adc.read_u16();                  //Read ADC (A0 chan)
        for (int i = 0; i < send_report.length; i++) //Fill the report
            send_report.data[i] = 0x00;
        send_report.data[0] = !SW1.read();
        send_report.data[3] = (raw>>8);
        send_report.data[4] = (raw & 0xff);
        hid.send(&send_report);                         //Send the report
        if(hid.readNB(&recv_report)) {                  //try to read a msg
            LED_1 = recv_report.data[0];
            LED_2.write(recv_report.data[1]*0.01f);
        }
    }
}
```

USB HID eszközök működésének vizsgálata

- *Alan Ott* multiplatformos programkönyvtára, a **HIDAPI** Linux, Mac OS X és Windows alatt is kezelni tudja az USB, illetve Bluetooth HID eszközöket. A programkönyvtár C/C++ alkalmazásainkból meghívható, így egyszerűen készíthetünk USB HID eszközt kezelő saját alkalmazásokat
- A **HIDAPI** programcsomag egy **testgui** nevű grafikus felületű tesztprogramot is tartalmaz, amellyel egyszerűen kipróbálhatjuk és vizsgálhatjuk HID eszközeinket
- A programcsomag régebbi, **hidapi-0.7.0** változata még tartalmazta a közvetlenül futtatható (előre lefordított) Windowsos **testgui.exe** tesztprogramot (az újabb változatok már csak a forráskódját teszik közzé), most ezt a régebbi változatot fogjuk használni

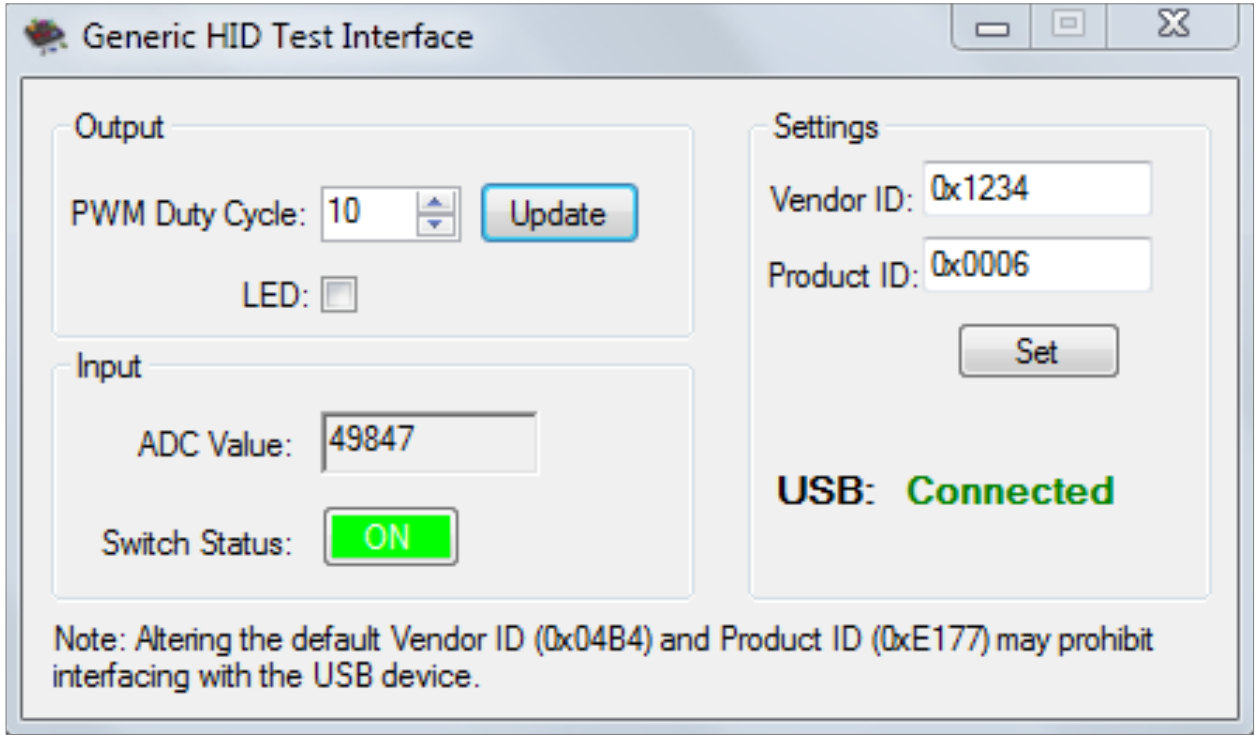
Lab10_USBHID_demo és a HIDAPI testgui.exe





A Lab10_USBHID_demo futtatása

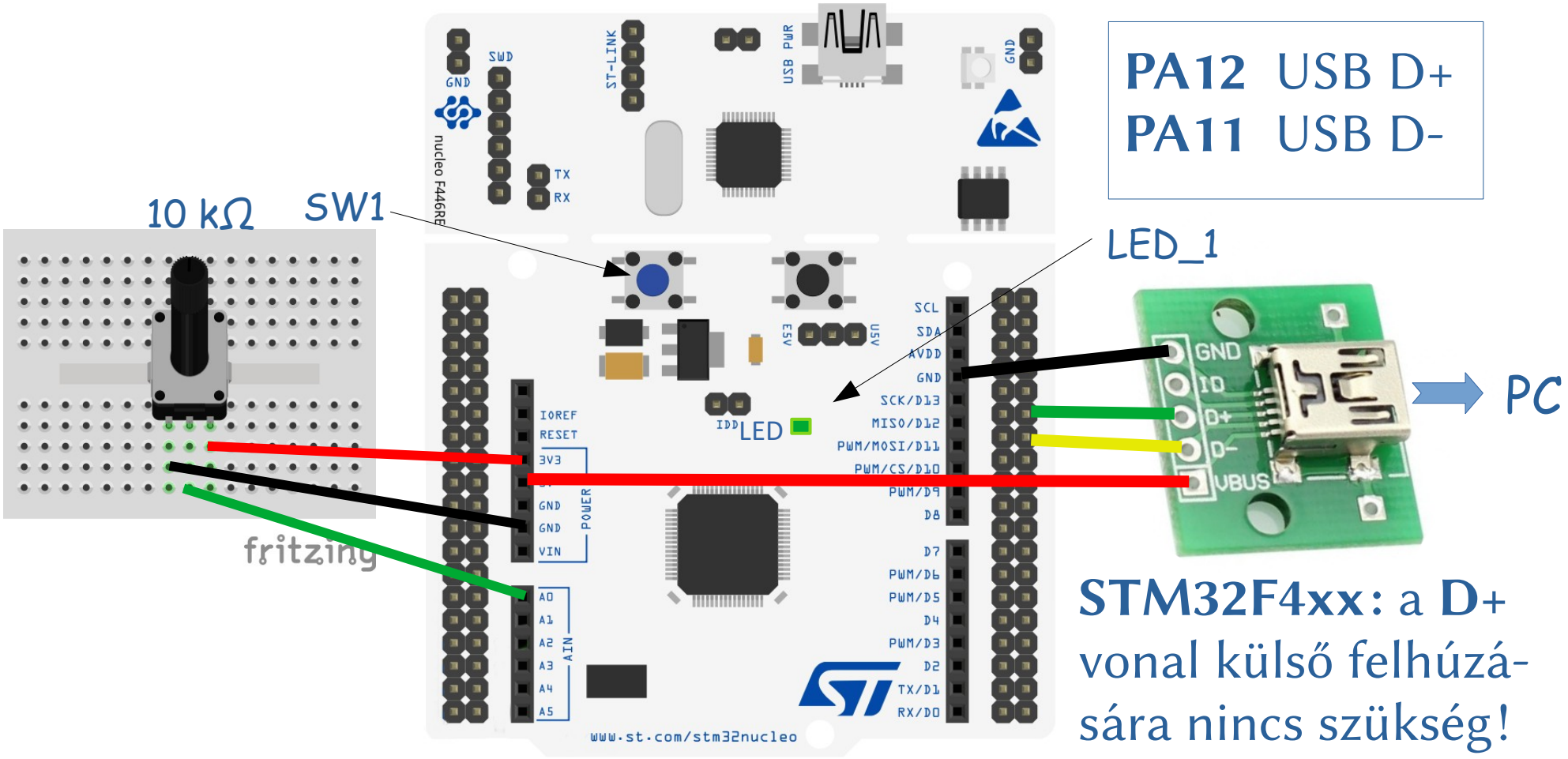
- Be kell állítanunk a **VID** és **PID** azonosítókat és rá kell kattintanunk a **Set** gombra!
- A piros LED-et a **LED** felirat melleti jelölőnégyzet kiválasztásával kapcsolható be és ki.
- A zöld LED fényereje a **PWM Duty Cyle** címke mellett állítható be 1 és 100 közötti értékre, de csak az **Update** gomb kattintásakor lesz kiküldve.
- A bejövő értékek (az ADC konverzió 16 bites eredménye és a **D3** digitális bemenet állapota) a vezérlőelemek alatt látható.



A program kipróbálásához az **AN82072** alkalmazási mintapélda letöltött ZIP állományából a Host Application/Windows Application mappából a **Generic HID UI.exe** és a **CyUSB.dll** állományok kellenek

Lab10_USBHID_pnp

- A 2021. április 8-i előadás **F446RE_USB_HID_PnP** projektjét írjuk át **Mbed** környezetre
- A kapcsolási vázlat az alábbi ábrán látható



STM32F4xx: a **D+** vonal külső felhúzására nincs szükség!

Lab10_USBHID_pnp

- A program a PC felől érkező 64 bájtos üzenetekből az első bájtot parancsként értelmezi, az alábbi táblázat szerint

Parancs	Válasz	Szöveges leírás
0x80	nincs	LED állapot átbillentése (LED1 a PA5 kivezetésre van kötve)
0x81	0x81 1/0	Nyomógomb állapotának lekérdezése (SW1 a PC13 kivezetésre van kötve)
0x37	0x37 LSB MSB	ADC IN0 csatorna értékének lekérdezése (IN0 a PA0 kivezetésre van kötve)
egyéb	0xFF	Nem értelmezett parancs

- A **0x80** parancsra nem küldünk választ
- A **0x81** parancsnál a második bájt 1 vagy 0, a **PC13** bemenet állapotától függően
- A **0x37** parancsnál az **ADC** által visszaadott érték 0 – 4095 közötti szám, amit *low endian* sorrendben küldünk ki az üzenet második és harmadik bájtjában
- PC oldalon a Microchip Library for Applications (MLA) **USB HID PnP demo** mintaalkalmazását „vettük kölcsön” és adaptáltuk (Vid/Pid csere, ProgressBar1 maximuma 1024 helyett 4096)

Lab10_USBHID_pnp/main.cpp – 2/1.

- A HID eszköz itt 64 bájtos jelentéseket küld/fogad
- A VID: 0x0483, PID: 0x5750 páros az ST Mictroelectronics STM32 Custom Human interface eszköz azonosítója

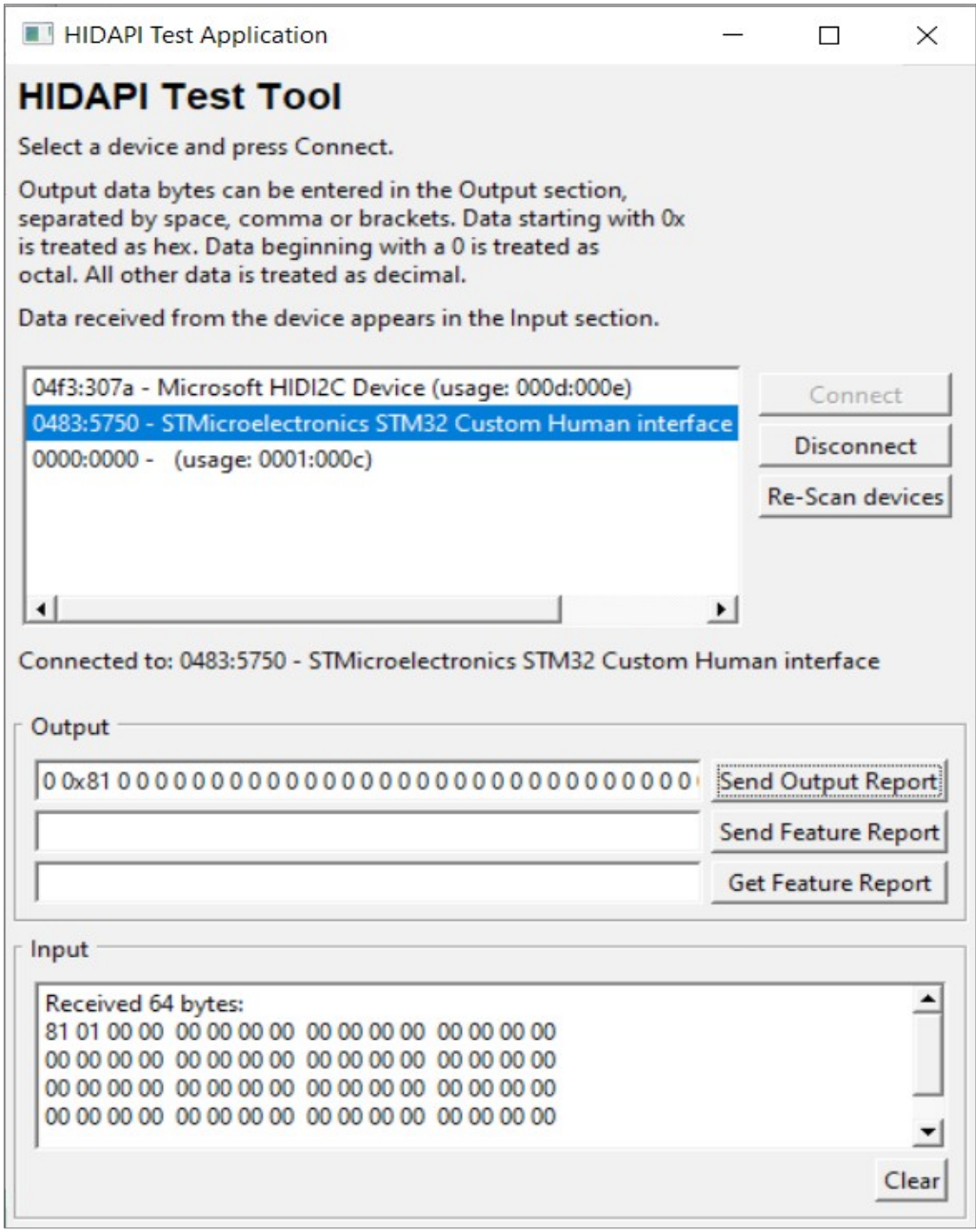
```
#include "mbed.h"
#include "USBHID.h"

USBHID hid(64,64,0x0483,0x5750); // Microchip HID PnP Demo adopted for STM32

HID_REPORT send_report;          // This report will contain data to be sent
HID_REPORT recv_report;          // This report will contain data received

DigitalOut LED_1(LED1);          // Buitin KED at PA5
DigitalIn SW1(BUTTON1, PullUp);  // Builtin button at PC13
AnalogIn adc(A0);                // Analog input at A0 (PA0)

int main(void) {
  uint16_t val = 0;
  send_report.length = 64;
  LED_1 = 0;
  for (int i = 0; i < send_report.length; i++)                //Fill the report
    send_report.data[i] = 0x00;
  while (1) {
```

Lab10_USBHID_pnp/main.cpp – 2/2.

```
    if (hid.readNB(&recv_report)) {                    // Try to read a message
      switch (recv_report.data[0]) {
        case 0x80: //--- Toggle LED state
          LED_1 = !LED_1;
          break;
        case 0x81: //--- Get push button state
          send_report.data[0] = recv_report.data[0];
          send_report.data[1] = SW1;
          send_report.data[2] = 0;
          hid.send(&send_report);                      // Send the report
          break;
        case 0x37: //--- Read POT command.
          send_report.data[0] = recv_report.data[0];
          val = adc.read_u16() >> 4;                   // Convert to 12-bits
          send_report.data[1] = val & 0xFF;            // Measured value LSB
          send_report.data[2] = val >> 8;              // Measured value MSB
          hid.send(&send_report);                      // Send the report
          break;
        default:  {
            send_report.data[0] = 0xFF;                // Invalid command
            hid.send(&send_report);                    // Send the report
        }
      }
    }
  }
}
```

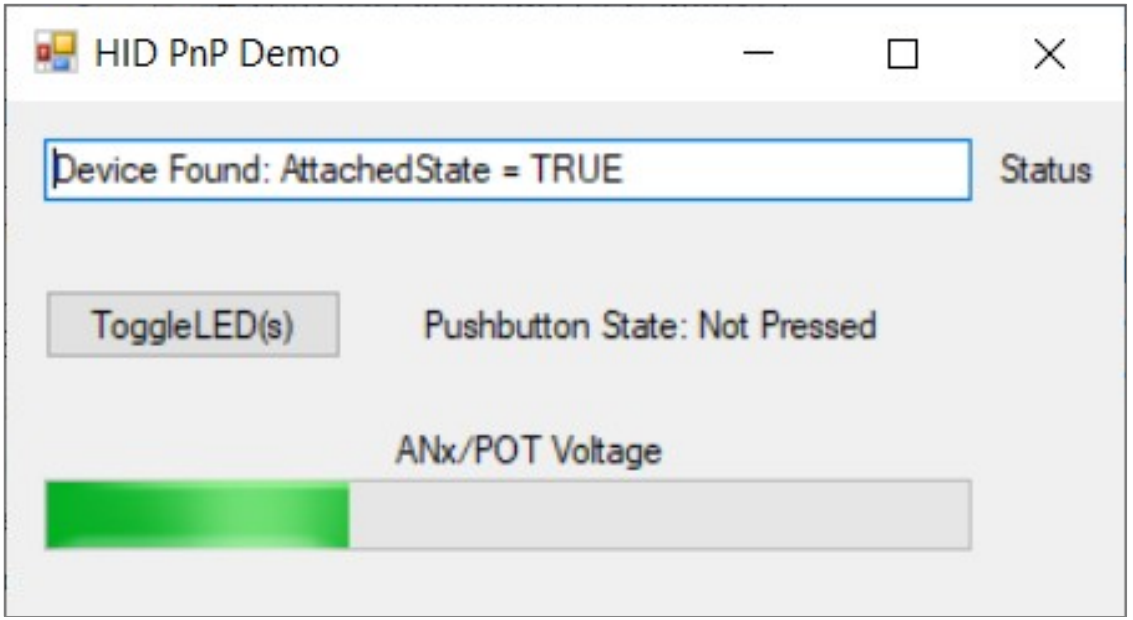
Lab10_USBHID_pnp: tesztelés

- A HIDapi **testgui.exe**-vel tesztelünk
- A **0483:5750** eszközt kiválasztjuk, majd a **Connect** gombra kattintunk
- Az **Output** ablakba 65 bájtot kell írni (az első kötelezően nulla és nem része az üzenetcsomagnak)
- A **Send Output Report** gombra kattintva küldhetjük ki az üzenetet
- A beérkező üzenetek az input ablakban jelennek meg: Az ábrán látható válasz (0x81 01 ...) azt jelzi, hogy nincs lenyomva a nyomógomb
- A 37 4d 09 00 kezdetű üzenet pedig azt jelezné, hogy a **PA0** bemenet jele 0x94D, ami kb. 1,91 V-nak felel meg



Lab10_USBHID_pnp próba HID PnP Demo-val

- A **HID PnP Demo** alkalmazás automatikusan megkeresi a **0483:5750** azonosítójú eszközt és kapcsolódik (vagy megszakadás után újrakapcsolódik). Kapcsolós esetén TRUE a státusz
- A **NUCLEO-F446RE** kártya beépített LED-je a **ToggleLED** gombbal ki/be kapcsolgatható, a beéépített **B1** nyomógomb állapota és az **A0** bemeneten mért analóg jel értéke pedig automatikusan frissül



Az USBKeyboard osztály

- Az **USBKeyboard** objektumosztály a billentyűzet emulálására szolgál az USB-porton keresztül, szabványos USB HID keyboard protokoll szerint
  Küldhetünk karakterfüzéreket, billentyűkódokat, módosítókkal *(pl. CTRL + 's')*, továbbá funkció- és a médiavezérlő-billentyű kódokat is

Tagfüggvények:

- **USBKeyboard név(***Vid=0x1235, Pid=0x0050, release=0x0001***)** – Konstruktor
- **keyCode** (key, modifier=0) – karakterkód küldése módosítóval
- **_putc** (int c) – karakter küldése
- **mediaControl** (media_key) – médiavezérlő gomb kódot küld
- **LockStatus** () – a *lock* állapotok beolvasása
- **send** (*report*) – HID jelentés küldése (blokkoló várakozással)
- **sendNB** (*report*) – HID jelentés küldése (nem blokkoló várakozással)
- **read** (*report*) – HID jelentés fogadása (blokkoló várakozással)
- **readNB** (*report*) – HID jelentés fogadása (nem blokkoló várakozással)

Speciális billentyűkódok és módosítók

```
/* Modifiers, left and right keys. */
enum MODIFIER_KEY {
    KEY_CTRL = 0x01,
    KEY_SHIFT = 0x02,
    KEY_ALT = 0x04,
    KEY_LOGO = 0x08,
    KEY_RCTRL = 0x10,
    KEY_RSHIFT = 0x20,
    KEY_RALT = 0x40,
    KEY_RLOGO = 0x80,
};
```

```
enum MEDIA_KEY {
    KEY_NEXT_TRACK,      // next Track
    KEY_PREVIOUS_TRACK, // Previous track
    KEY_STOP,            // Stop
    KEY_PLAY_PAUSE,      // Play/Pause
    KEY_MUTE,            // Mute
    KEY_VOLUME_UP,       // Volume Up
    KEY_VOLUME_DOWN,     // Volume Down
};
```

```
enum FUNCTION_KEY {
    KEY_F1 = 128,    /* F1 key */
    KEY_F2,          /* F2 key */
    KEY_F3,          /* F3 key */
    KEY_F4,          /* F4 key */
    KEY_F5,          /* F5 key */
    KEY_F6,          /* F6 key */
    KEY_F7,          /* F7 key */
    KEY_F8,          /* F8 key */
    KEY_F9,          /* F9 key */
    KEY_F10,         /* F10 key */
    KEY_F11,         /* F11 key */
    KEY_F12,         /* F12 key */
    KEY_PRINT_SCREEN,
    KEY_SCROLL_LOCK,
    KEY_CAPS_LOCK,
    KEY_NUM_LOCK,
    KEY_INSERT,
    KEY_HOME,
    KEY_PAGE_UP,
    KEY_PAGE_DOWN,
    RIGHT_ARROW,
    LEFT_ARROW,
    DOWN_ARROW,
    UP_ARROW,
};
```

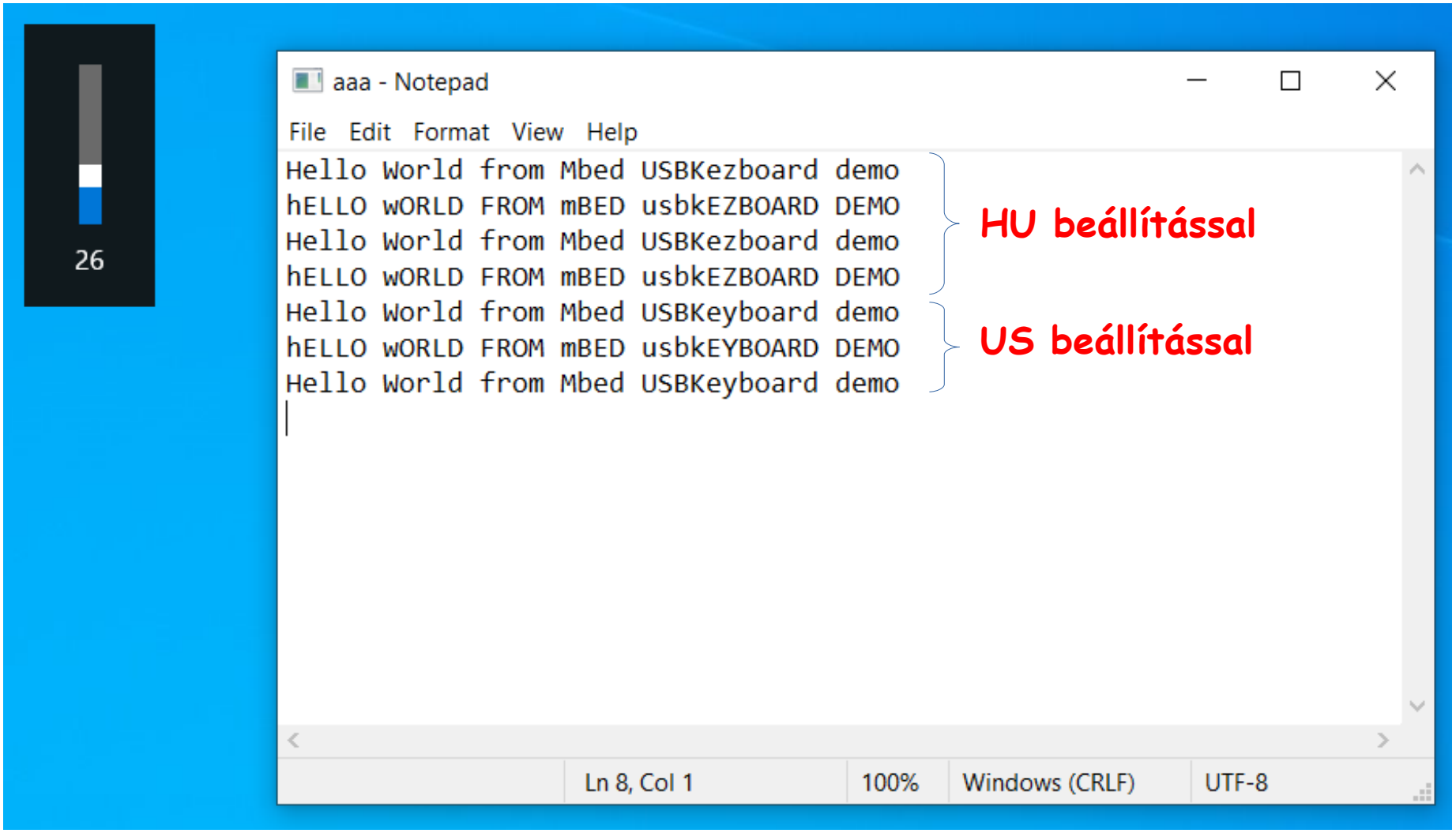
LAB10_USB_keyboard/main.cpp

- A program a beépített nyomógomb lenyomásakor egy fokozattal csökkenti a hangerő beállítását, kiküld egy sor szöveget, egy **CTRL-S** parancsot és átbillenti a **CAPS LOCK** beállítást

```
#include "mbed.h"                 // Ehhez importáljuk az mbed 2 könyvtárat
#include "USBKeyboard.h"          // Ehhez importáljuk az USBDevice könyvtárat

DigitalIn SW1(BUTTON1,PullUp); // A beépített nyomógombot használjuk
USBKeyboard keyboard;

int main(void) {
    while(1) {
        while(SW1==1) {}       // SW1 lenyomásra várunk
        keyboard.mediaControl(KEY_VOLUME_DOWN);
        keyboard.printf("Hello World from Mbed USBKeyboard demo\r\n");
        keyboard.keyCode('s', KEY_CTRL);
        keyboard.keyCode(KEY_CAPS_LOCK);
        wait(0.02);
        while(SW1==0) {}       // SW1 felengedésére várunk
        wait(0.02);
    }
}
```

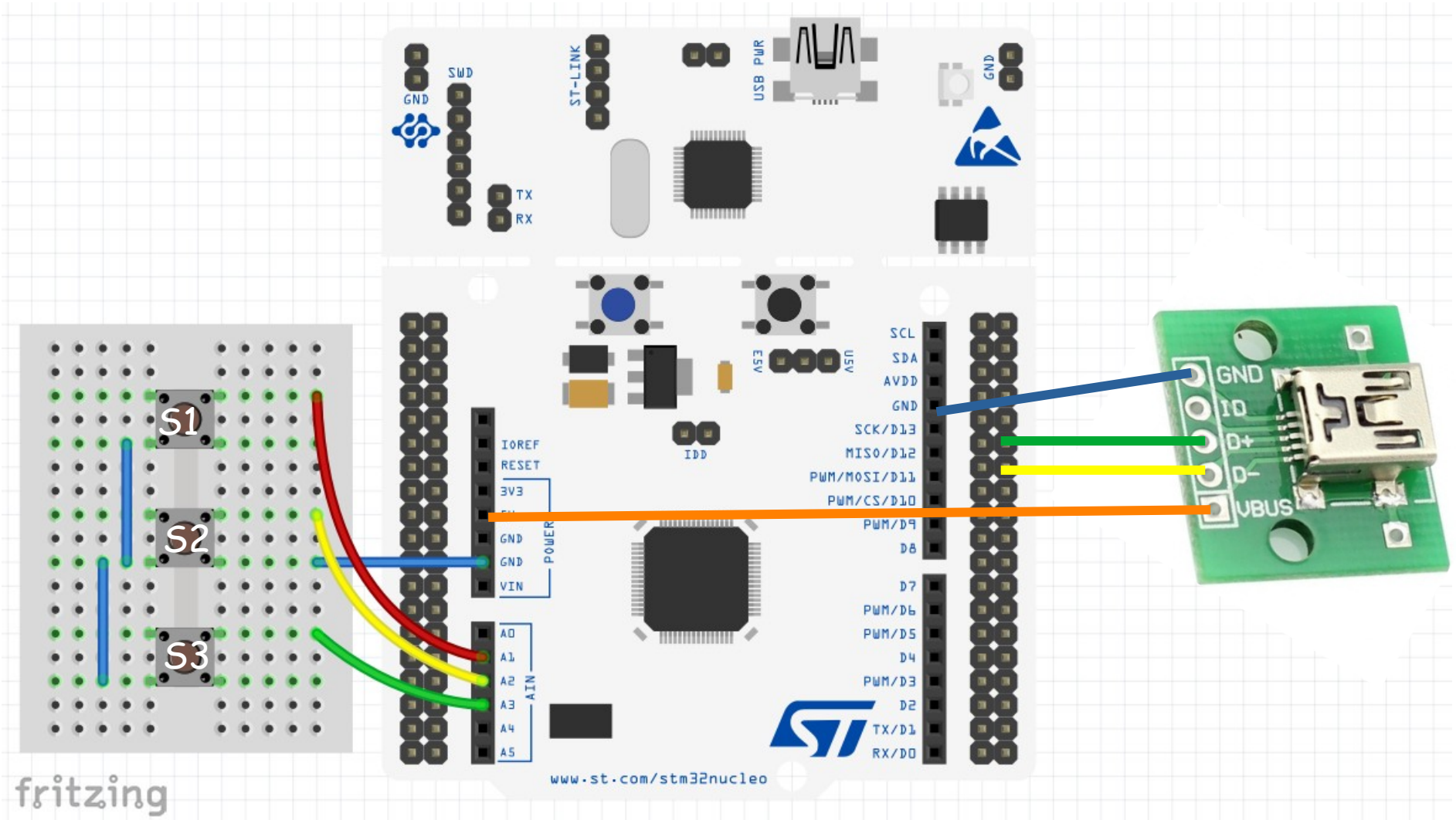
LAB10_USB_keyboard

- Windows 10 alatt, **Notepad.exe**-vel próbáltuk ki a programot, **HU** és **US** billentyűzet beállítással



Lab10_USB_kwikboard

- Bővítsük ki a kapcsolást három nyomógombbal, kössük ezeket rendre az **A1, A2, A3** bemenetek, illetve a **GND** kivezetés közé!



LAB10_USB_kwikboard/main.cpp

- Bővítettük az előző programot is: az **S1, S2** és **S3** nyomógombok **Win+'R'** parancsokat indítanak egy-egy weblap megnyitásához

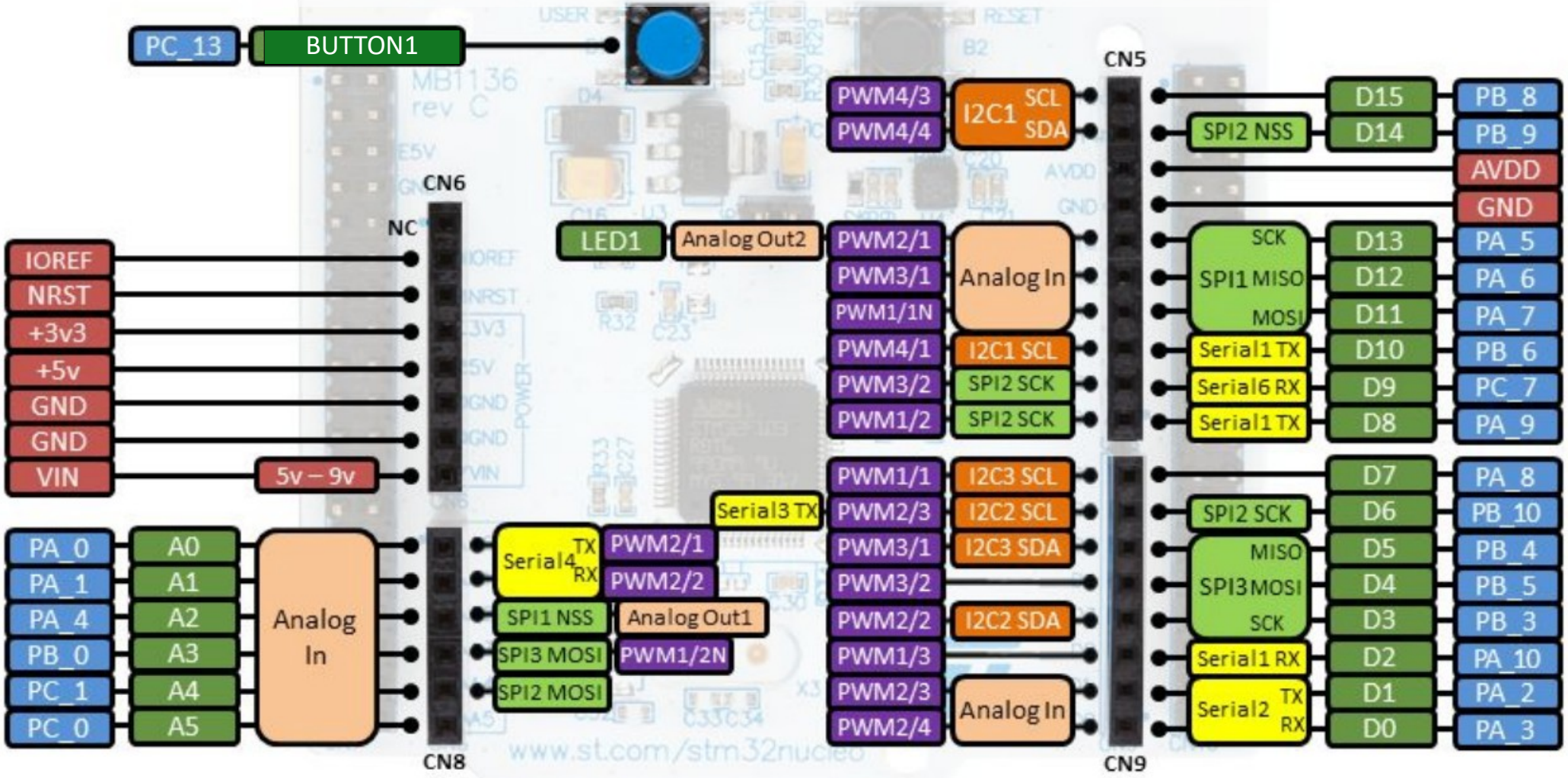
```
#include "mbed.h"
#include "USBKeyboard.h"
DigitalIn B1(BUTTON1,PullUp); //A beépített nyomógomb
DigitalIn S1(A1,PullUp);
DigitalIn S2(A2,PullUp);
DigitalIn S3(A3,PullUp);
USBKeyboard keyboard;
int B1state = 1;              //0: felengedésre 1: lenyomásra várunk
int S1state = 1;
int S2state = 1;
int S3state = 1;

int main(void) {
    while(1) {
        if (B1state && !B1) {
            keyboard.mediaControl(KEY_VOLUME_DOWN);
            keyboard.printf("Hello World from Mbed USBKeyboard demo\r\n");
            keyboard.keyCode('s', KEY_CTRL);
            keyboard.keyCode(KEY_CAPS_LOCK);
            B1state = 0;
            wait(1);
        } else if(!B1state && B1) B1state = 1;
```

LAB10_USB_kwikboard/main.cpp – 2/2.

```
        if (S1state && !S1) {
             keyboard.keyCode('r', KEY_LOGO);  // Windows kay + R key command
             keyboard.printf("https://megtestesules.info\r\n ");
             S1state = 0;
             wait(1);
         }
         else if(!S1state && S1) S1state = 1;

         if (S2state && !S2) {
             keyboard.keyCode('r', KEY_LOGO);  // Windows key + R key command
             keyboard.printf("https://megtestesules.info/hobbielektronika/\r\n ");
             S2state = 0;
             wait(1);
         }
         else if(!S2state && S2) S2state = 1;

         if (S3state && !S3) {
             keyboard.keyCode('r', KEY_LOGO);  // Windows key + R key command
             keyboard.printf("https://cspista.hu/\r\n ");
             S3state = 0;
             wait(1);
         }
         else if(!S3state && S3) S3state = 1;
         wait(0.02);
    }
}
```

Arduino kompatibilis kivezetések

- Kivezetés azonosításra a kék, illetve a sötétzöld címkék használhatók (pl. PA_5, D13, vagy LED1)



Morpho kivezetések

Nucleo F446RE
Morpho Headers