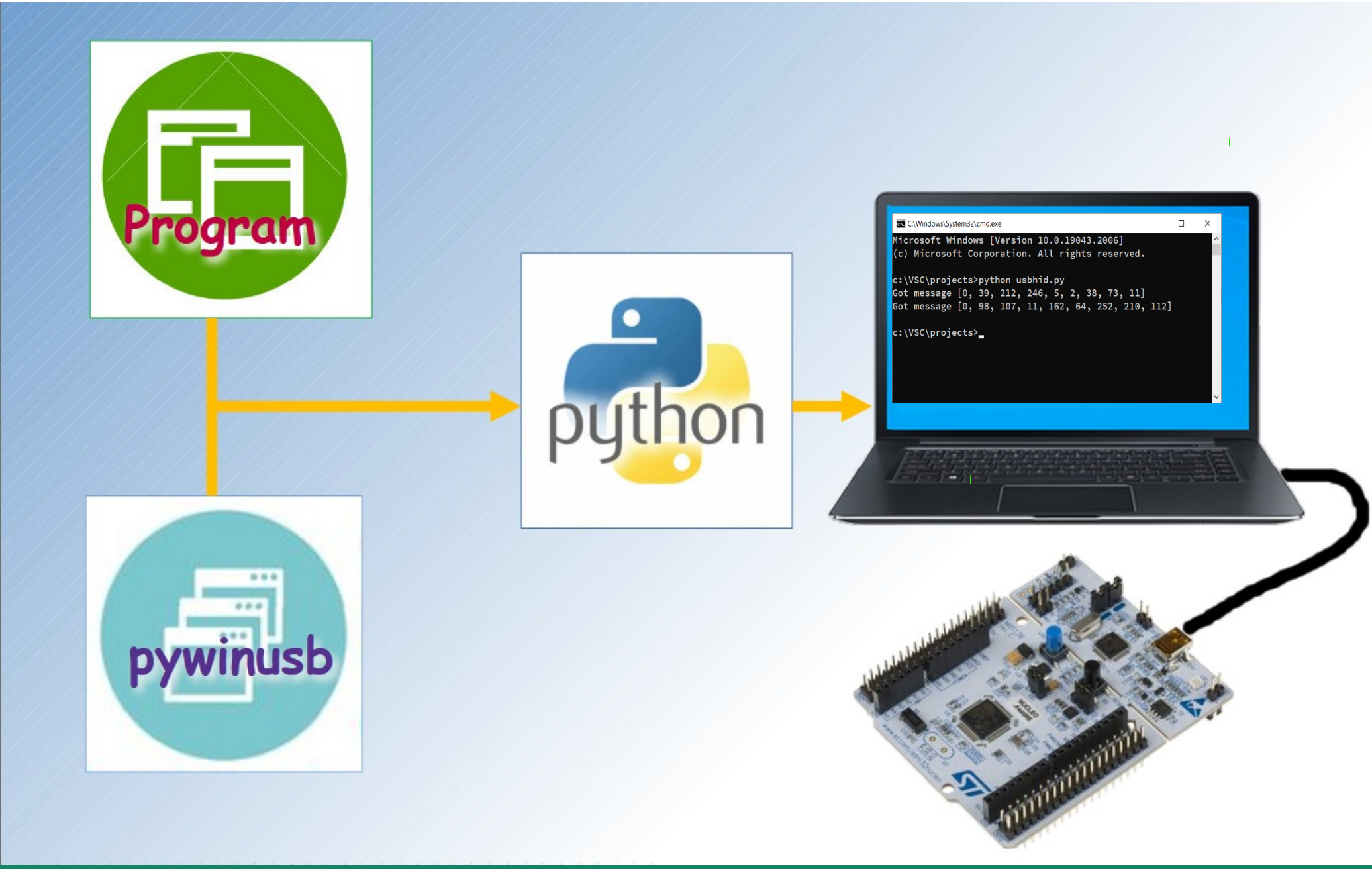
3. A pywinusb Python bővítmény használata

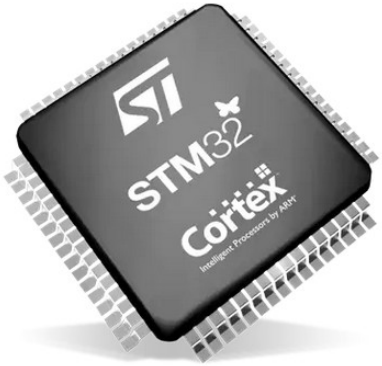


Felhasznált és ajánlott irodalom

- Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!
- P. Wentworth et al. (ford. Biró Piroska, Szeghalmy Szilvia és Varga Imre): Hogyan gondolkozz úgy, mint egy informatikus: Tanulás Python 3 segítségével
- Cserny István: A FRDM-KL25Z kártya programozása mbed környezetben
- **René Aguirre: pywinusb: USB/HID windows helper library**
- ARM mbed honlap: https://os.mbed.com/
- ARM Keil Studio: https://studio.keil.arm.com/
- **ARM mbed OS Documentation: https://os.mbed.com/docs/mbed-os/**
- Keil Studio manual: https://developer.arm.com/documentation/102 497/latest
- ARM mbed forráskód: https://github.com/ARMmbed/mbed-os

Adatlapok:

- **STM32F446RE** adatlap és termékinfo
- **STM32F446** Family Reference Manual

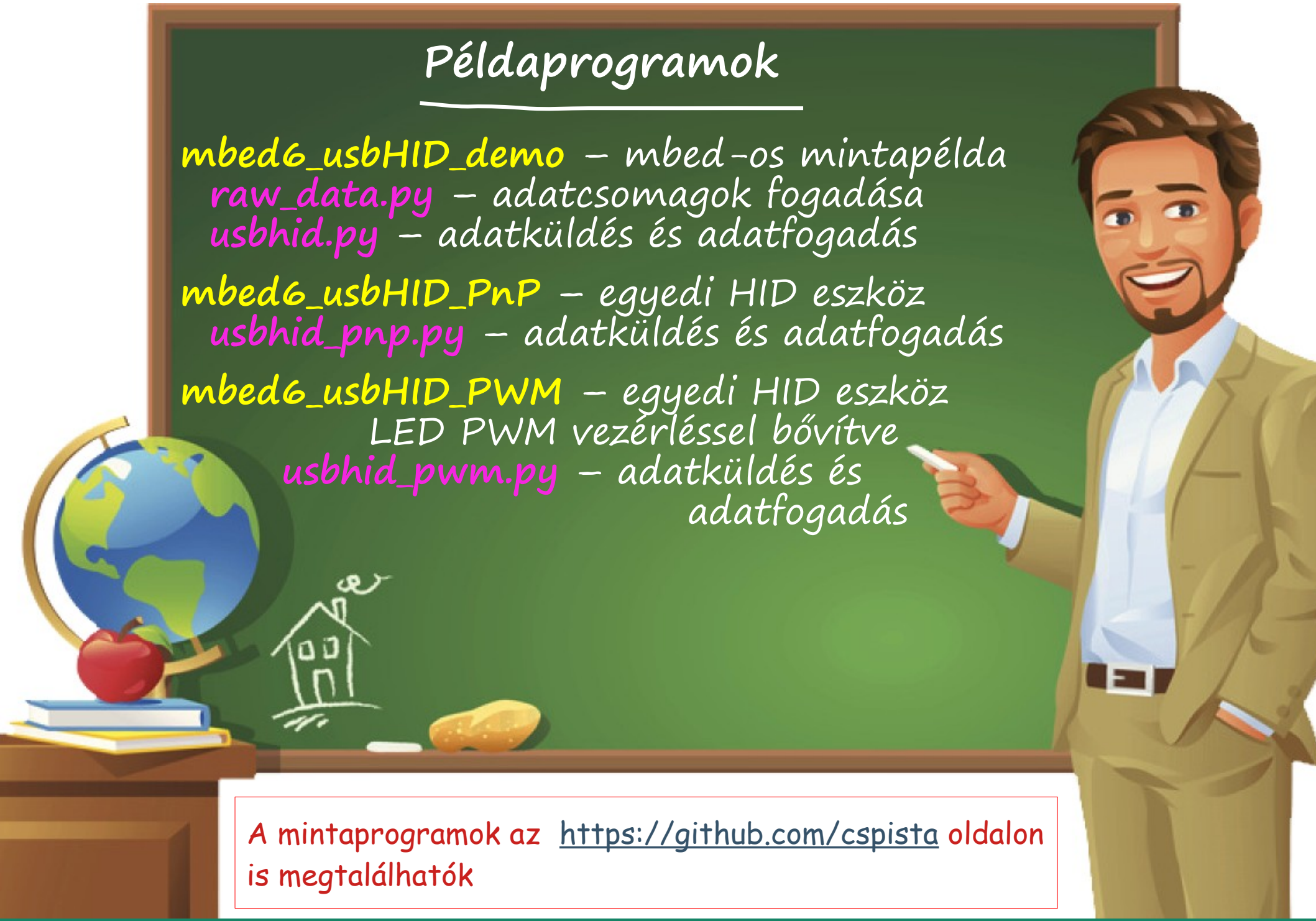


Példaprogramok

mbed6_usbHID_demo – mbed-os mintapélda
raw_data.py – adatcsomagok fogadása
usbhid.py – adatküldés és adatfogadás

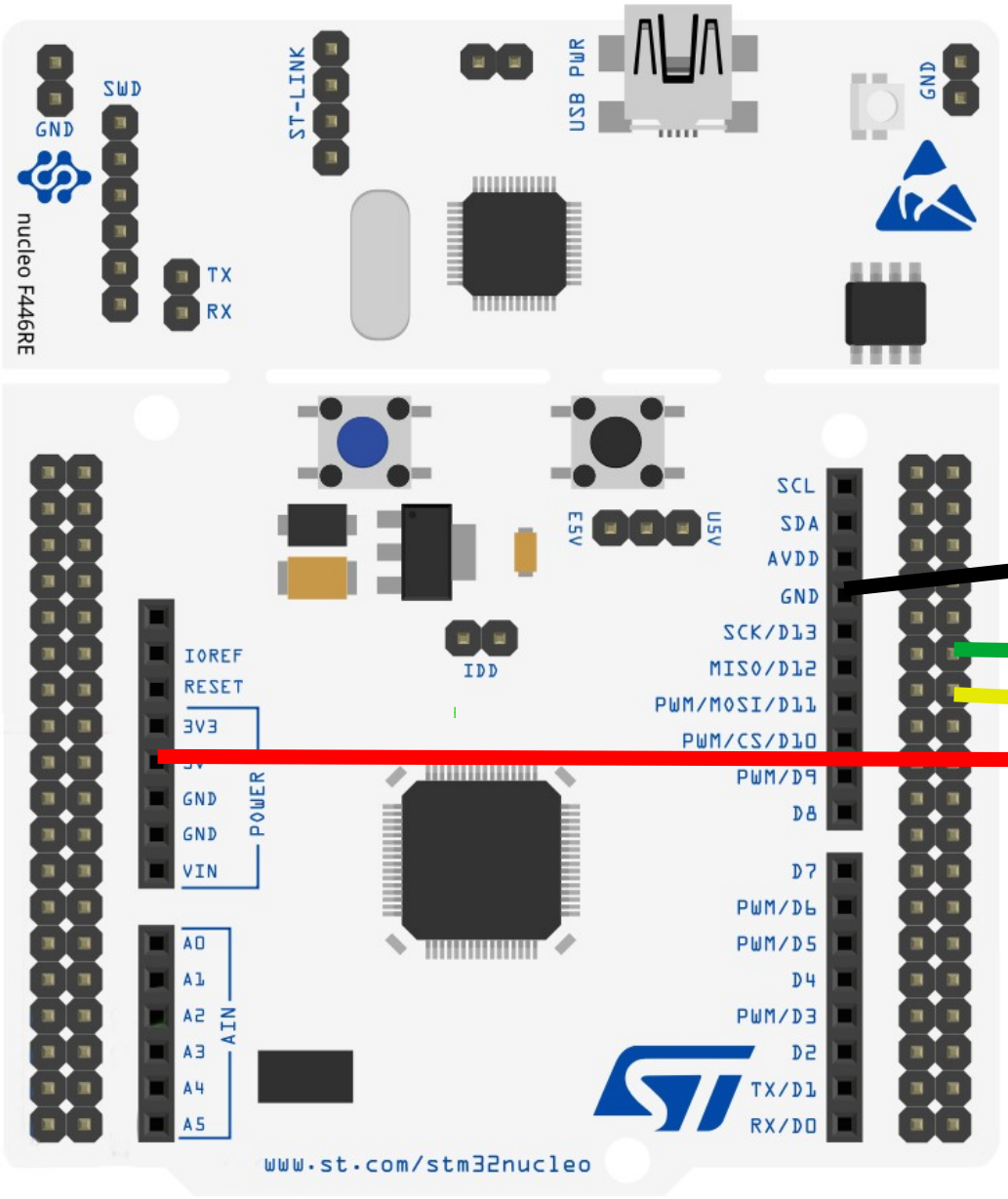
mbed6_usbHID_PnP – egyedi HID eszköz
usbhid_pnp.py – adatküldés és adatfogadás

mbed6_usbHID_PWM – egyedi HID eszköz LED PWM vezérléssel bővítve
usbhid_pwm.py – adatküldés és adatfogadás

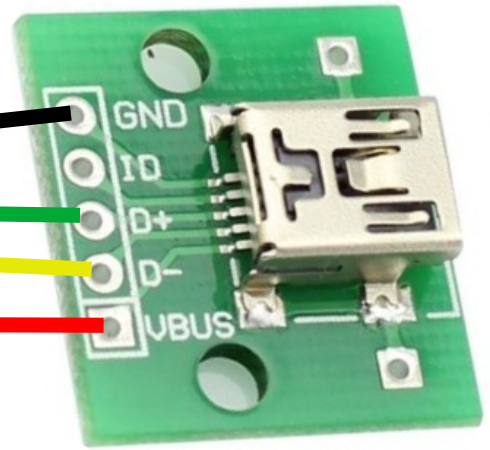


A mintaprogramok az https://github.com/cspista oldalon is megtalálhatók

Az USB_FS csatorna használata



PA12 USB D+
PA11 USB D-



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

NUCLEO-F446RE mbed_app.json

- A **NUCLEO-F446RE** kártyán az **USB_OTG_FS** csatornát használjuk, ennek konfigurálásához az **mbed_app.json** fájlban az alábbi beállítás szükséges (**"*"** helyett valójában **"NUCLEO_F446RE"**-et kellene írni)
- Ha más eszközt is konfigurálnunk kell (SD kártya, soros terminál, akkor az alábbiakat természetesen ki kell majd egészíteni a szükséges sorokkal

```
{
  "config": {
    "usb_speed": {
      "help": "USE_USB_OTG_FS or USE_USB_OTG_HS or USE_USB_HS_IN_FS",
      "value": "USE_USB_OTG_FS"
    }
  },
  "target_overrides": {
    "*": {
      "target.device_has_add": ["USBDEVICE"]
    }
  }
}
```

Az USB nem 'component', hanem 'device'

Egyedi USB HID eszköz

- 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 miként szabványos, az adatok felhasználása azonban egyedi, gyártóspecifikus.
- Az **USBHID** osztály az előzőekben említett *"Általános HID Eszközt"* (Generic HID Device) valósítja meg
- A **számítógépen kell olyan alkalmazás,** amely képes kezelni az általunk készített egyedi eszköz **USBHID** adatforgalmát.
  Ez most egy szkript lesz, amihez a Python értelmező **pywinusb** kiegészítését használjuk (Linux környezetben a **pyusb** csomag használható, de annak más a szintaktikája)
- Egy további 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 az említett könyvtárak használatára

mbed6_usbHID_demo/main.cpp

A program forrása: Mbed 6 Documentation (USBHID API)

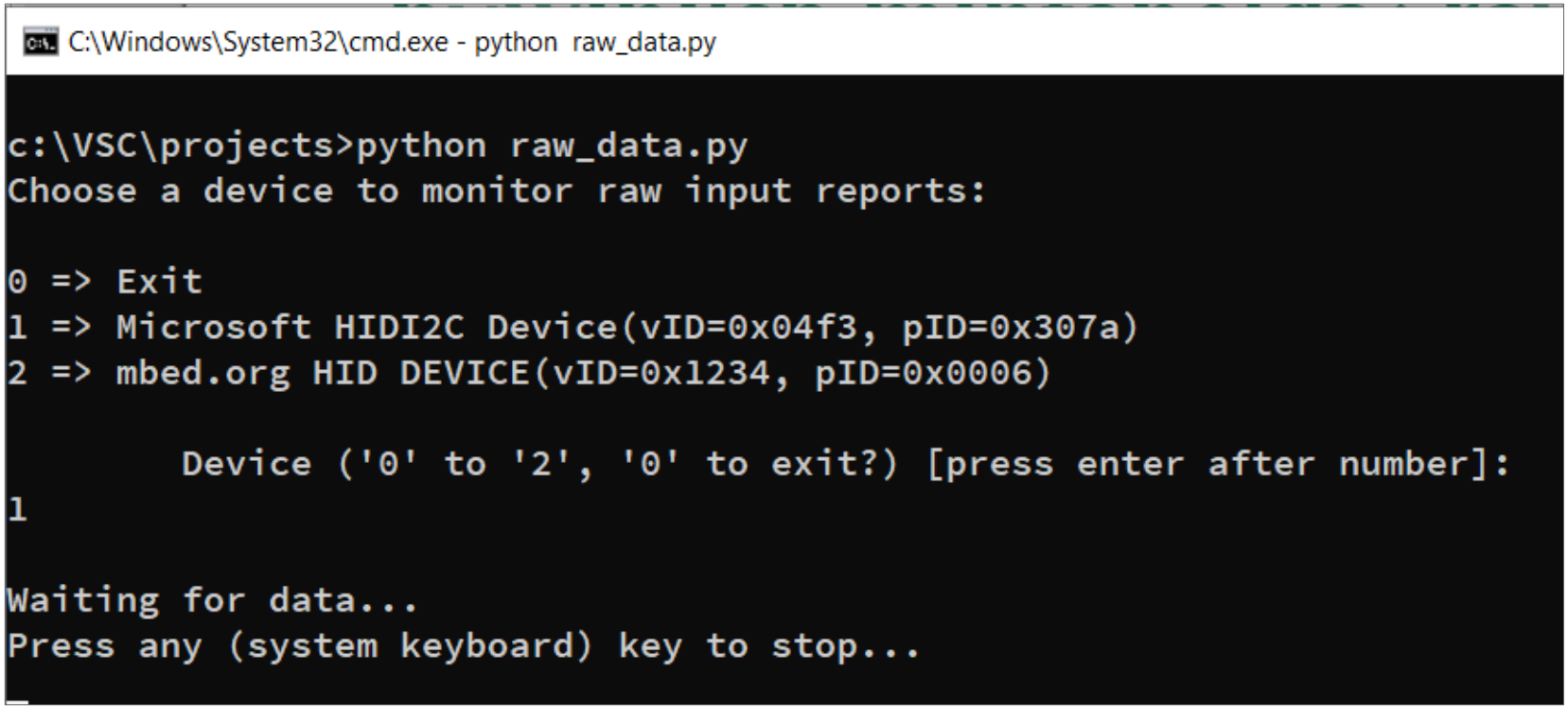
```
#include <stdio.h>
#include "mbed.h"
#include "USBHID.h"
USBHID HID(true, 8, 8, 0x1234, 0x0006, 0x0001);
HID_REPORT output_report = {.length = 8, .data = {0} };
HID_REPORT input_report = {.length = 0, .data = {0} };

DigitalOut led_out(LED1);
int main(void) {
    while (1) {
        for (int i = 0; i < output_report.length; i++) {
            output_report.data[i] = rand() & UINT8_MAX;
        }
        HID.send(&output_report);           // Send the report
        if (HID.read_nb(&input_report)) {   // Try to read a msg
            led_out = !led_out;             // Toggle LED state
            for (int i = 0; i < input_report.length; i++) {
                printf("%d ", input_report.data[i]);
            }
            printf("\r\n");
        }
    }
}
```

pywinusb

- **René Aguirre: pywinusb: USB/HID windows helper library**
- Legutolsó kiadása: **ver 0.4.2** (2016. december 12)
  egyelőre csak Windows alathasználható
- Telepítése (ha a Python már telepítve van): `pip install pywinusb`
- Importálása: `from pywinusb import hid`
- Kompatibilitás: **Python 2.7,** illetve **Python 3.5** (a mai előadásban bemutatott programok **Python 3.10** alatt is működnek)
- Leírás: szűkszavú leírás a **pywinusb wiki oldalán** található
- Mintapéldák: a **github.com/rene-aguirre/pywinusb** címen elérhető programcsomag **examples** mappája tartalmaz néhány mintapéldát, azonban kompatibilitási okoból ezek többségét módosítások nélkül nem tudjuk használni

pywinusb mintapélda: **raw_data.py**

- A **pywinusb** ezen mintapéldája először kilistázza az összes kívülről csatlakoztatott USB HID eszközt, majd lehetővé teszi, hogy egy Enterrel lezárt számjegy beírásával kiválasszuk valamelyiket
- Ezután a kiválasztott eszközről érkező üzeneteket végtelen ciklusban kilistázza, mindaddig, amíg az eszköz csatlakoztatott állapotban van és amíg le nem ütünk a billentyűzeten egy gombot



pywinusb mintapélda: raw_data.py

```
from time import sleep
from msvcrt import kbhit
import pywinusb.hid as hid

def sample_handler(data):
    print("Raw data: {0}".format(data))

def raw_test():
    all_hids = hid.find_all_hid_devices()       # browse devices...
    if all_hids:
      while True:
        print("Choose a device to monitor raw input reports:\n")
        print("0 => Exit")
        for index, device in enumerate(all_hids):
          device_name = unicode("{0.vendor_name} {0.product_name}" \
                  "(vID=0x{1:04x}, pID=0x{2:04x})"\
                  "".format(device, device.vendor_id, device.product_id))
          print("{0} => {1}".format(index+1, device_name))
        print("\n\tDevice ('0' to '%d', '0' to exit?) " \
              "[press enter after number]:" % len(all_hids))
        index_option = raw_input()
        if index_option.isdigit() and int(index_option) <= len(all_hids):
                break;
```

pywinusb mintapélda: raw_data.py

```
    int_option = int(index_option)
    if int_option:
      device = all_hids[int_option-1]
      try:
        device.open()
        #set custom raw data handler
        device.set_raw_data_handler(sample_handler)
        print("\nWaiting for data...\nPress any key to stop...")
        while not kbhit() and device.is_plugged():
           #just keep the device opened to receive events
           sleep(0.5)
           return
      finally:
        device.close()
    else:
      print("There's not any non system HID class device available")

if __name__ == '__main__':
  import sys
  if sys.version_info >= (3,):
    unicode = str
    raw_input = input
  raw_test()
```

pywinusb mintapélda: raw_data.py

```
C:\Windows\System32\cmd.exe
c:\VSC\projects>python raw_data.py
Choose a device to monitor raw input reports:

0 => Exit
1 => Microsoft HIDI2C Device(vID=0x04f3, pID=0x307a)
2 => mbed.org HID DEVICE(vID=0x1234, pID=0x0006)

        Device ('0' to '2', '0' to exit?) [press enter after number]:
2

Waiting for data...
Press any (system keyboard) key to stop...
Raw data: [0, 50, 46, 210, 209, 80, 101, 84, 0]
Raw data: [0, 200, 103, 242, 126, 147, 5, 133, 58]
Raw data: [0, 81, 76, 73, 135, 122, 158, 156, 117]
Raw data: [0, 235, 245, 247, 188, 139, 162, 188, 249]
Raw data: [0, 140, 141, 79, 249, 22, 83, 184, 145]
Raw data: [0, 73, 110, 181, 198, 40, 34, 140, 232]
Raw data: [0, 31, 152, 165, 192, 241, 34, 197, 43]
Raw data: [0, 186, 95, 32, 73, 123, 167, 184, 89]
Raw data: [0, 176, 96, 51, 89, 45, 167, 198, 57]
Raw data: [0, 107, 225, 44, 58, 143, 190, 58, 22]
Raw data: [0, 175, 86, 220, 212, 29, 99, 177, 229]
```

NUCLEO-F446RE

Véletlen számokból álló üzenetcsomagok

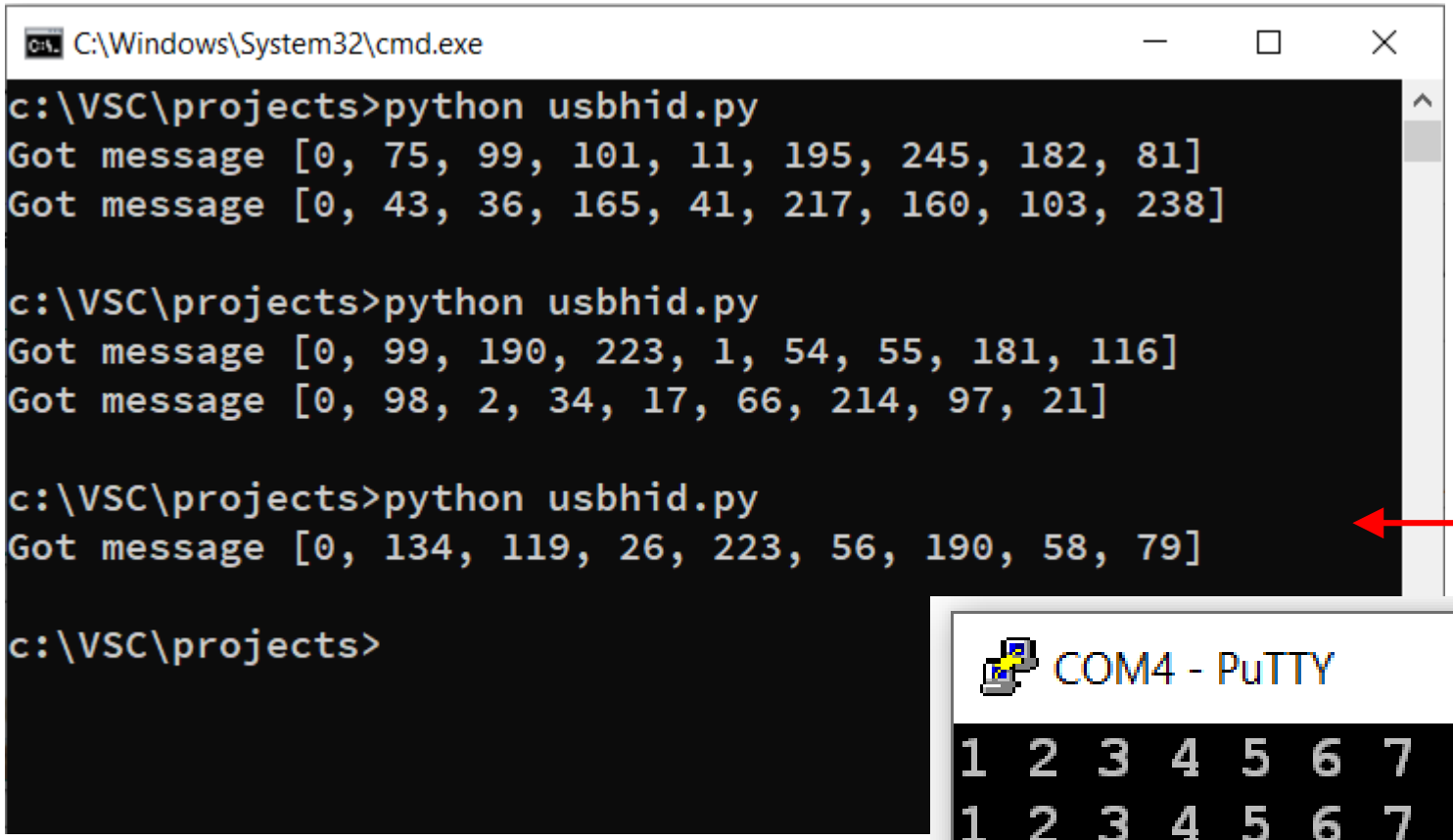
usbhid.py

- Az itt bemutatott **usbhid.py** program az Mbed OS 6 dokumentáció **USBHID.py** mintapéldája, néhány jelentéktelen módosítással
- Az előző példaprogramtól eltérően itt automatikusan választjuk ki az első olyan USB HID eszközt, amelynél a gyártó neve tartalmazza az 'mbed' karaktersorozatot
- Az eszköz megnyitása után egy visszahívási függvényt rendelünk az `on_data` eseményhez (ez csak kilistázza a beérkező adatokat)
- A következő lépésben összeállítunk egy 9 bájtos csomagot (a kezdő 0. bájt a jelentés azonosítója, kötelezően nulla), s ezt a csomagot üzenetként elküldjük az eszköznek, majd lezárjuk a kapcsolatot
- Az USB HID eszköz (a NUCLEO-F446RE kártya) minden üzenetcsomag vételekor átbillenti a beépített LED állapotát, az üzenetcsomag tartalmát pedig kilistázza a soros terminálon (és az ST-Link v2-1 USB-soros átalakítóján) keresztül

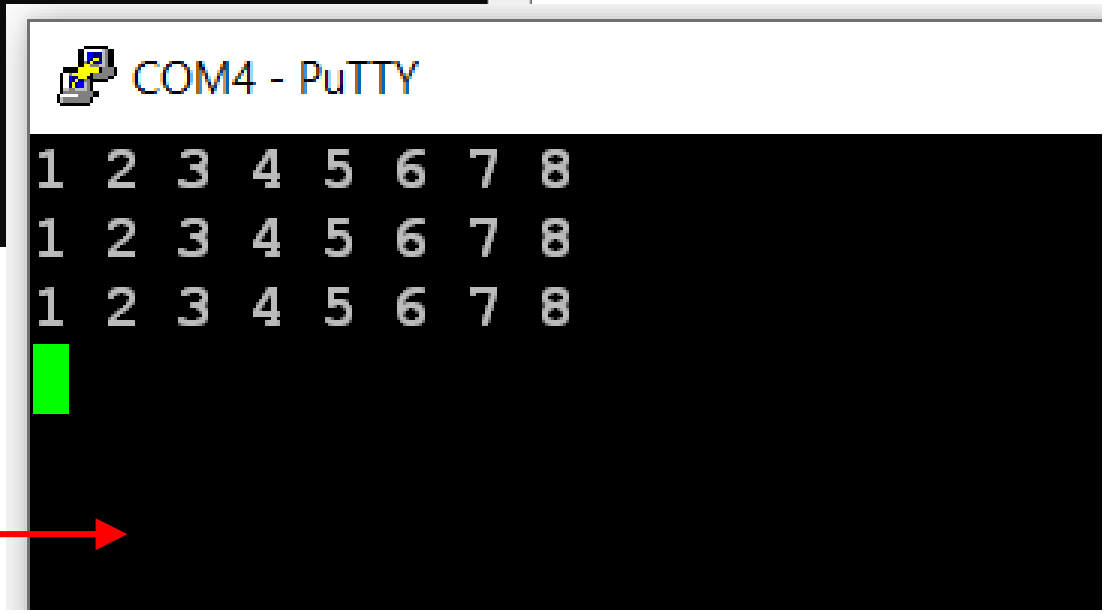
usbhid.py

```
from pywinusb import hid
# Whenever the host computer receives data from the
# Mbed board, the received data is printed
def on_data(data):
    print(f"Got message {data}")

# Gets all HIDs currently connected to host computer,
# and sets the first device with string "mbed" in its vendor name
all_hid_devices = hid.find_all_hid_devices()
mbed_devices = [d for d in all_hid_devices if "mbed" in d.vendor_name]
if mbed_devices is None:
    raise ValueError("No HID devices found")

# A buffer of bytes representing the message to be sent
Buffer = [0, 1, 2, 3, 4, 5, 6, 7, 8]      # The first byte is the report ID
mbed_devices[0].open()
mbed_devices[0].set_raw_data_handler(on_data) # Set custom raw data handler

# Send the message to the Mbed board
out_report = mbed_devices[0].find_output_reports()
out_report[0].set_raw_data(buffer)
out_report[0].send()
mbed_devices[0].close()
```

usbhid.py és mbed6_usbHID_demo



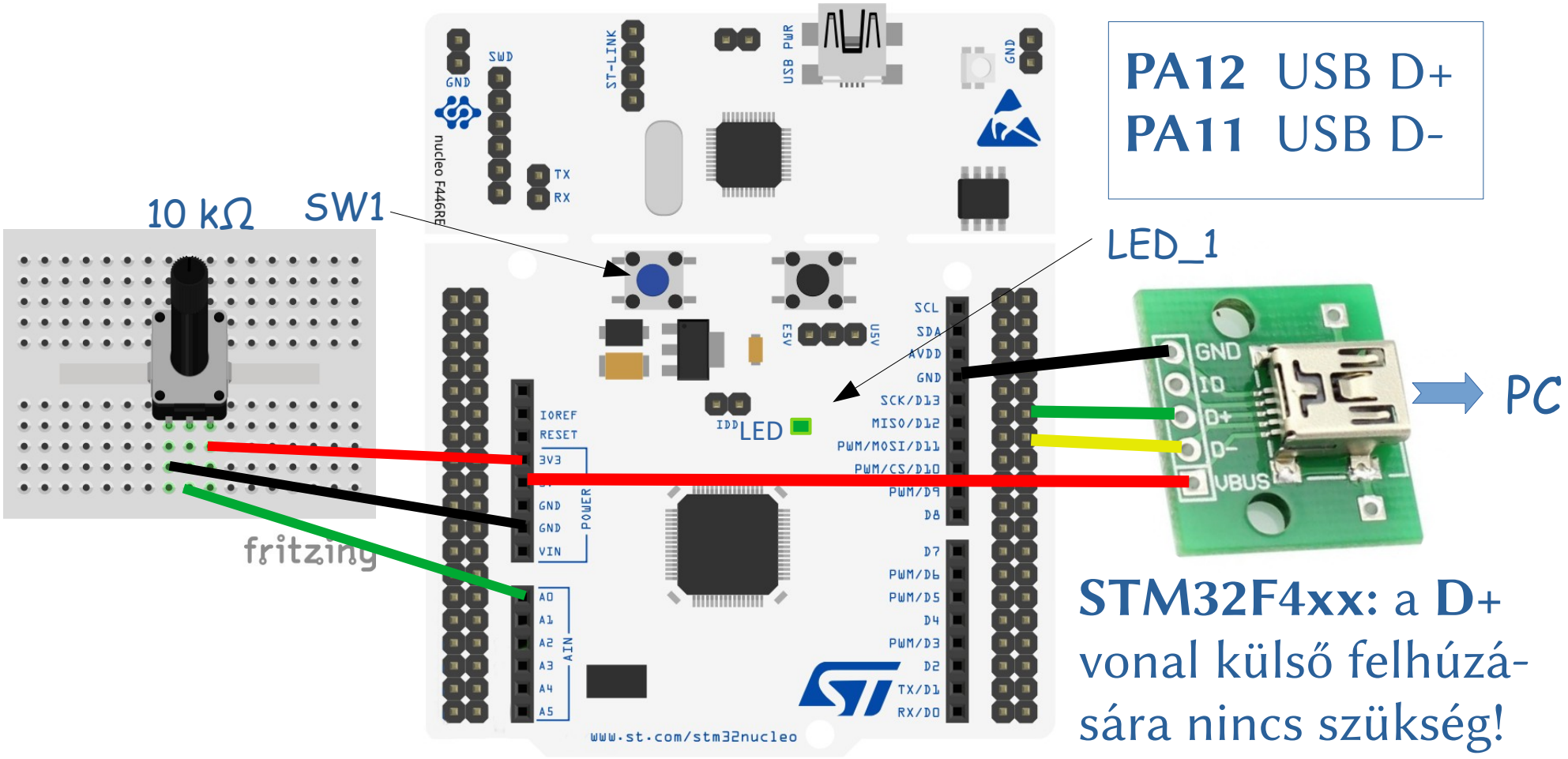
- Az eszköz felől érkező csomagok első bájtja a packet ID, ez esetünkben mindig nulla



- A soros porton és az ST-Linken keresztül, a PC felől az USB porton át beérkező csomagok tartalmát íratjuk ki

mbed6_usbHID_PnP

- A 2021. április 8-i előadás **F446RE_USB_HID_PnP** projektjét írtuk át **Mbed-OS v6** környezetre (az előző előadásban bemutattuk)
- A kapcsolási vázlat az alábbi ábrán látható



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

mbed6_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)

mbed6_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 "usb/USBHID.h"

//We declare a USBHID device. Here the input and output reports are 64 bytes long.
USBHID hid(true,64,64,0x0483,0x5750); // Vid/Pid: STM32 Custom Human Interface Device

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
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) {
```

mbed6_usbHID_PnP/main.cpp – 2/2.

```
        if(hid.read_nb(&recv_report)) {       // read a msg (it was readNB formerly)
            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
                }
            }
        }
    }
}
```

usbhid_pnp.py

- Az előzőleg bemutatott Python szkriptekből hellyel-közzel összeollózható egy egyszerű program, amellyel kapcsolódhatunk az mbed eszközünkhöz
- Az eszköz automatikus megnyitásához most a Vid/Pid páros megadásával választjuk ki a HID eszközök közül a sajátunkat

```
all_devices = hid.HidDeviceFilter(vendor_id = 0x0483, product_id = 0x5750).get_devices()
```

- Ha az így kapott lista üres, akkor nincs csatlakoztatva az eszköz, egyébként pedig a lista 0. elemét vesszük elő
- A parancsok kiválasztását egy egyszerű menürendszer kezeli:
  - **0. Exit** (kilépés)
  - **1. Toggle LED state** (LED1 állapot átbillentése)
  - **2. Read Button state** (BUTTON1 állapotának lekérdezése)
  - **3. Read ADC value** (ADC konverzió eredményének lekérdezése)
- A menüpont sorszámának beírását **Enter**-rel kell lezárni

usbhid_pnp.py – 3/1.

```
from time import sleep
from msvcrt import kbhit
import pywinusb.hid as hid

def sample_handler(data):                # Receive message from the Mbed board
    print("Raw data: {0}".format(data))
    if data[1] == 0x81:
       if data[2] == 0:
          print("Button is pressed")
       else:
          print("Button is not pressed")
    elif data[1] == 0x37:
       adc = data[3]*256 + data[2]
       voltage = int(adc*3300/4096)
       print("ADC = {0} = {1} mV".format(adc,voltage))

def send_command(cmd):                   # Send the message to the Mbed board
    # The first byte is the report ID which must be 0
    buffer = [0 for i in range(65)]  # Array of 65 elements
    buffer[1] = cmd
    out_report = device.find_output_reports()
    out_report[0].set_raw_data(buffer)
    out_report[0].send()
```

usbhid_pnp.py – 3/2.

```
all_devices = hid.HidDeviceFilter(vendor_id = 0x0483,\
                  product_id = 0x5750).get_devices()

if not all_devices:
    raise ValueError("HID device not found")

device = all_devices[0]
print("Device connected!\r\nSelect option:")
print("0. Exit")
print("1. Toggle LED state")
print("2. Read Button state")
print("3. Read ADC value")

device.open()
device.set_raw_data_handler(sample_handler) # Set custom raw data handler

while device.is_plugged():
    print("\nCommand ('0' to '3') [press Enter after number]: ")
    while not kbhit() and device.is_plugged():
        index_option = input()
        if index_option.isdigit() and int(index_option) < 4:
            break;
    ix = int(index_option)
```

Vid = 0x0483

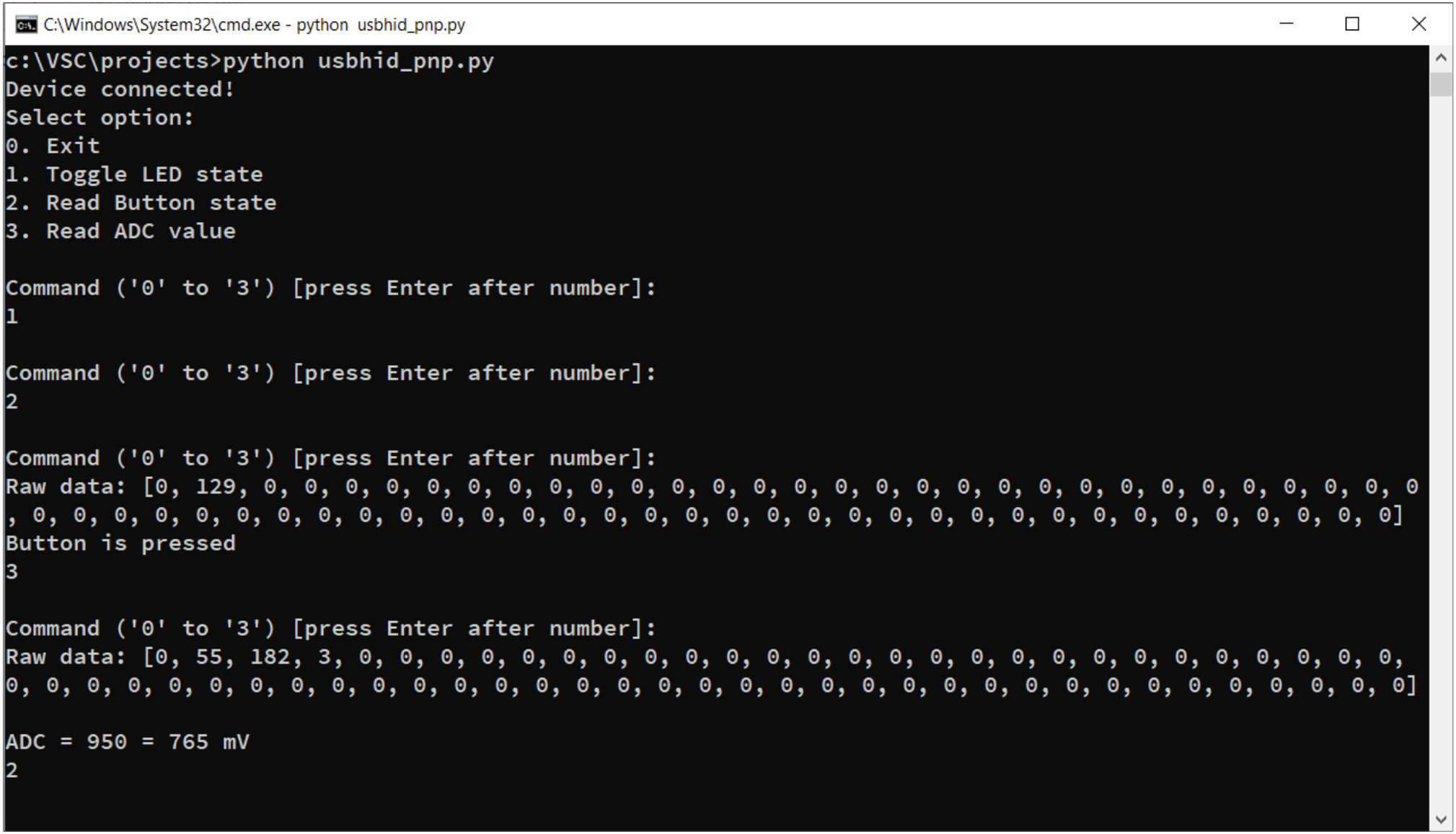
Pid = 0x5750

usbhid_pnp.py – 3/3.

```
    if ix == 0:
        device.close()
        exit()
        pass
    elif ix == 1:
        send_command(0x80)
        pass
    elif ix == 2:
        send_command(0x81)
        pass
    elif ix == 3:
        send_command(0x37)
        pass
```

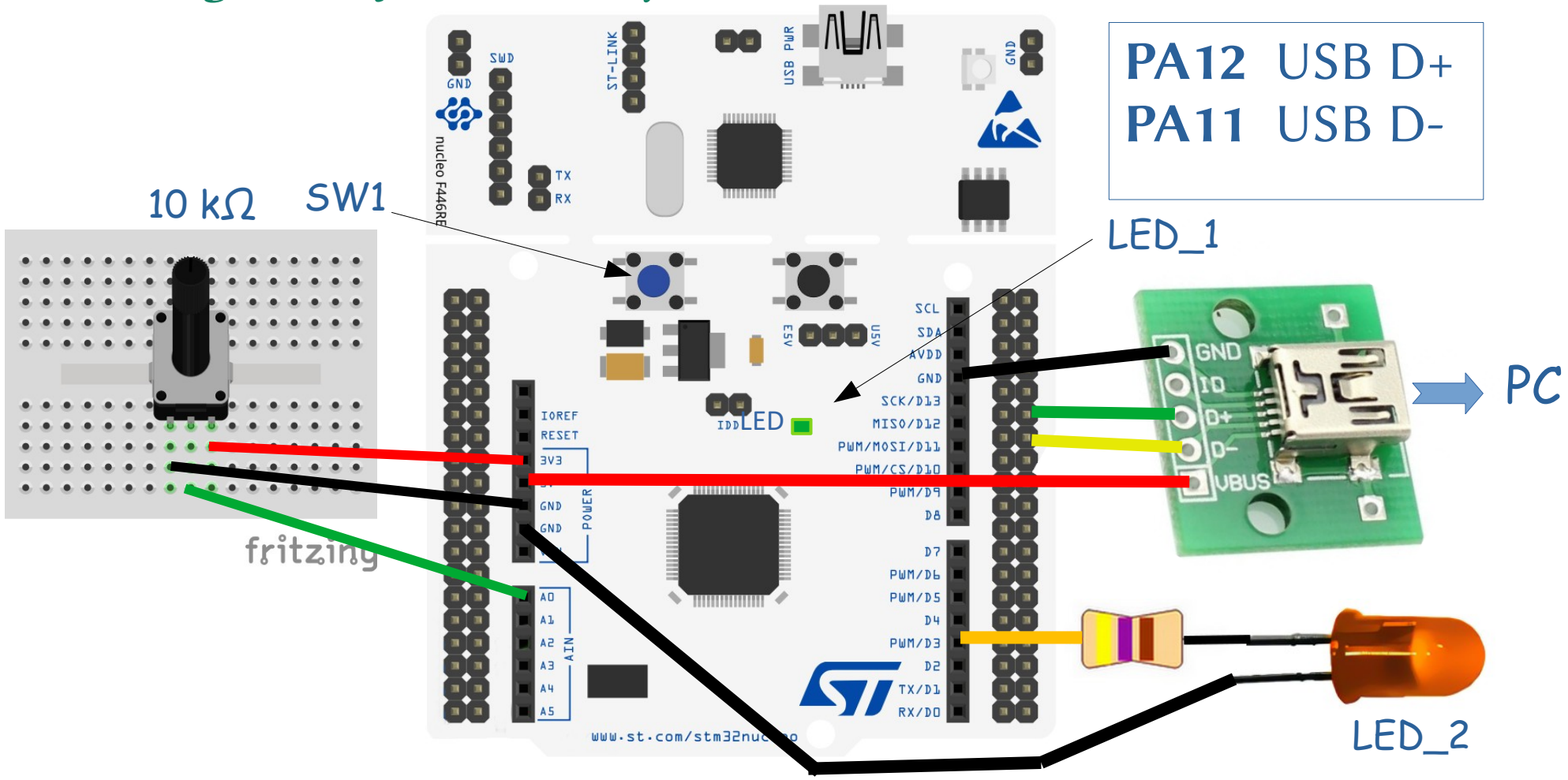
usbhid_pnp.py futási eredmény

- A kártya csatlakoztatása után indítsuk el a programot



mbed6_usbHID_PWM

- Egészítsük ki az előző kapcsolást egy LED-del, melynek anódját a **D3** Arduino kompatibilis kivezetésre kötjük!
- Ezt a LED-et PWM-mel, *„analóg”* módban vezéreljük, az **mbed-os AnalogOut** objektumosztályának felhasználásával



mbed6_usbHID_PWM

- Ezt a programot a 2022. június 16-i előadásban mutattuk be. Akkor **mbed 2** környezetben, a Cypress AN82072 alkalmazási mintapéldájához igazítva készítettük el, hogy annak a letölthető grafikus PC alkalmazását használni tudjuk. Most ezt a programot dolgozzuk át **Mbed Os 6** környezetre
- **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 a **D3** kimenetre kötött LED-et vezérel (1 - 100 közötti érték), 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...

Input Report Data

Button Status	:Byte 1
~~ADC_Data[31..24]~~	:Byte 2
~~ADC_Data[23..16]~~	:Byte 3
ADC_Data[15..8]	:Byte 4
ADC_Data[7..0]	:Byte 5
Unused (0x00)	:Byte 6
Unused (0x00)	:Byte 7
Unused (0x00)	:Byte 8

Output Report Data

LED Control	:Byte 1
PWM Duty Cycle	:Byte 2
Unused (0x00)	:Byte 3
Unused (0x00)	:Byte 4
Unused (0x00)	:Byte 5
Unused (0x00)	:Byte 6
Unused (0x00)	:Byte 7
Unused (0x00)	:Byte 8

mbed6_usbHID_PWM/main.cpp 2/1

- Változás: az **USBHID** konstruktor első paraméter itt egy 'true'

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

//We declare a USBHID device. Input and output reports are 8 bytes long.
USBHID hid(true, 8, 8, 0x1234, 0x0006, 0x0001);    //Mbed Custom HID

HID_REPORT send_report = {
    .length = 8,
    .data = {0}
};
HID_REPORT recv_report = {
    .length = 8,
    .data = {0}
};

DigitalOut LED_1(LED1);          //Builtin LED at PA5
PwmOut     LED_2(D3);            //External LED anode at D3 (PB3)
DigitalOut myGND(D4);            //Just for lazyness: LED cathode
DigitalIn  SW1(BUTTON1,PullUp); //Builtin button at PC13
AnalogIn   adc(A0);              //Analog input at A0 (PA0)
```

mbed6_usbHID_PWM/main.cpp 2/2

- Változások: **hid.readNB** helyett itt **hid.read_nb** írandó, továbbá az USB csatornát az **mbed_app.json** állományban engedélyezni és konfigurálni kell

```
int main(void) {
  LED_1 = 0;
  LED_2.period_ms(20);
  myGND = 0;
  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.read_nb(&recv_report)) {                 //try to read a msg
      LED_1 = recv_report.data[0];
      LED_2.write(recv_report.data[1]*0.01f);
    }
  }
}
```

usbhid_pwm.py

- Az eszköz megnyitását és a menükezelést az **usbhid_pnp.py** program mintájára készítjük el, azonban az érkező csomagokat csak lekérdező parancs (4. menüpont) esetén íratjuk, mert folyamatosan érkeznek a csomagok
- A választható menüpontok:
  - 0. Kilépés
  - 1. LED1 állapot átbillentése
  - 2. LED2 fényerő növelése (+30 %)
  - 3. LED2 fényerő csökkentése (–30%)
  - 4. BUTTON1 állapotának és az ADC konverzió eredményének a lekérdezése
- A beérkező csomagokat kezelő **sample_handler()** függvénynél ügyeljünk rá, hogy globális változókba (sw1 és adc) tároljuk el a beérkező adatokat!

usbhid_pwm.py – 3/1.

```
from time import sleep
from msvcrt import kbhit
import pywinusb.hid as hid

sw1, adc, voltage, led1, led2 = 0, 0, 0, 0, 10

def sample_handler(data):              # Callback function for receiving
    global sw1
    global adc
    sw1 = data[1]
    adc = data[4]*256 + data[5]

def send_command(led1,led2):           # Send the message to the Mbed board
    buffer = [0 for i in range(9)]     # Array of 9 elements
    buffer[1] = led1
    buffer[2] = led2
    out_report = device.find_output_reports()
    out_report[0].set_raw_data(buffer)
    out_report[0].send()

all_devices = hid.HidDeviceFilter(vendor_id = 0x1234,\
              product_id = 0x0006).get_devices()
```

Vid = 0x1234

Pid = 0x0006

usbhid_pwm.py – 3/2.

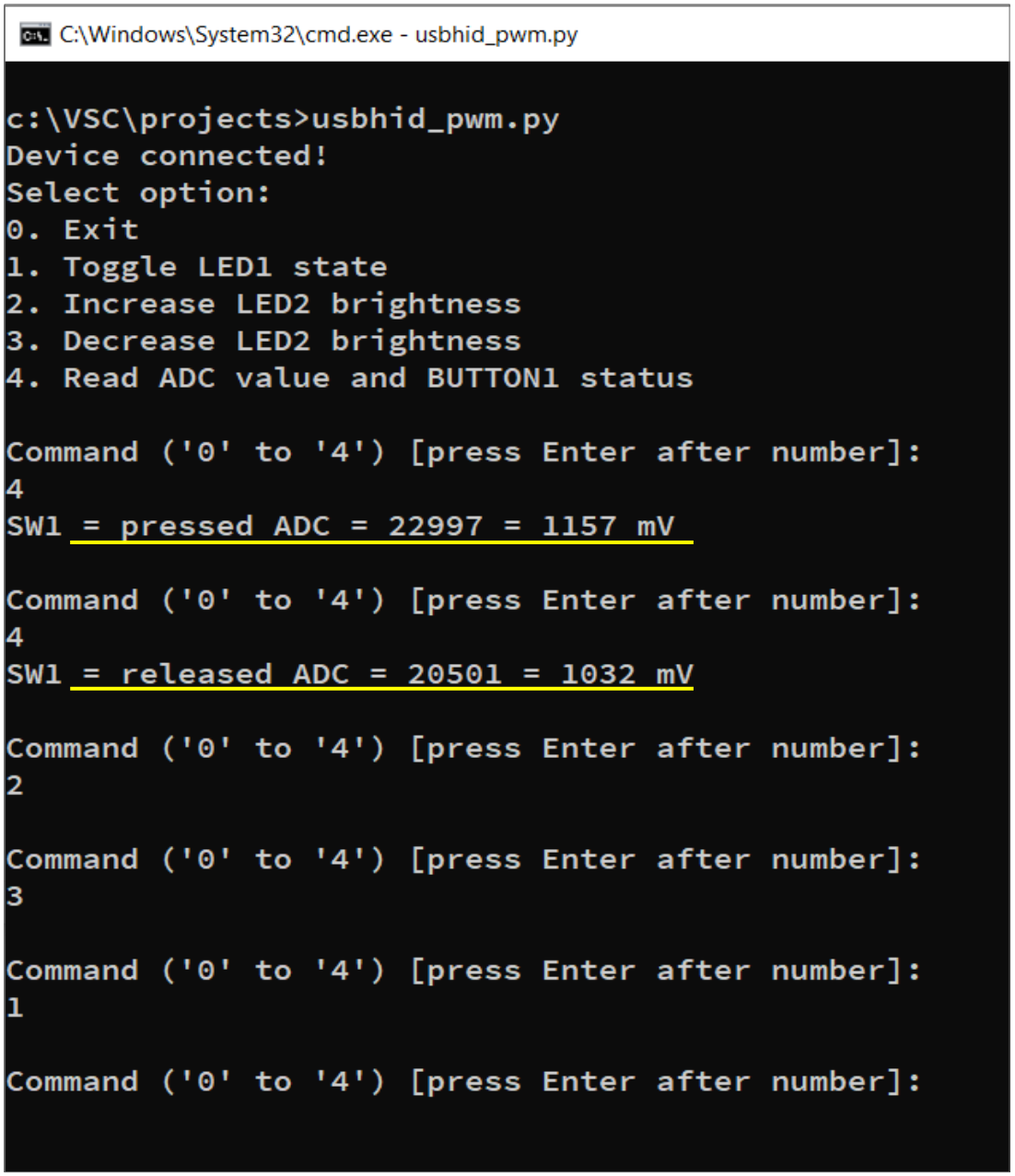
```
if not all_devices:
    raise ValueError("HID device not found")

device = all_devices[0]    # Take the first element of the list
print("Device connected!\r\nSelect option:")
print("0. Exit")
print("1. Toggle LED1 state")
print("2. Increase LED2 brightness")
print("3. Decrease LED2 brightness")
print("4. Read ADC value and BUTTON1 status")

device.open()
# Set custom raw data handler
device.set_raw_data_handler(sample_handler)
while device.is_plugged():
    print("\nCommand ('0' to '4') [press Enter after number]: ")
    while not kbhit() and device.is_plugged():
        index_option = input()
        if index_option.isdigit() and int(index_option) < 5:
            break;
    ix = int(index_option)
```

usbhid_pwm.py – 3/3.

```
if ix == 0:                         # Exit
    device.close()
    exit()
    pass
elif ix == 1:                       # Toggle LED1 state
    led1 = not led1
    send_command(led1,led2)
    pass
elif ix == 2:                       # Increase LED2 brightness
    led2 = led2 + 30
    if led2 > 100: led2 = 100
    send_command(led1,led2)
    pass
 elif ix == 3:                      # Decrease LED2 brightness
    led2 = led2 - 30
    if led2 < 1: led2 = 1
    send_command(led1,led2)
    pass
elif ix == 4:                       # Query SW1 state and ADC conversion result
    if sw1:
       btn = "pressed"
    else:
       btn = "released"
    voltage = int(adc*3300/65536)
    print("SW1 = {0} ADC = {1} = {2} mV".format(btn,adc,voltage))
    pass
```

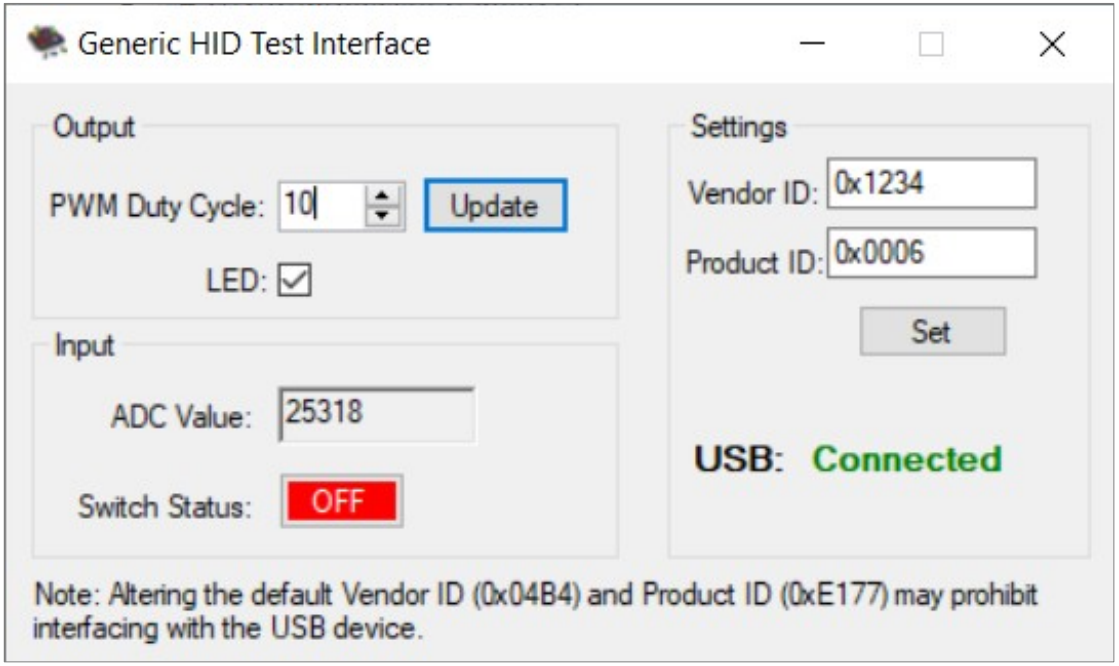
usbhid_pwm.py

- A kártya csatlakoztatása után indítva, a programunk automatikusan felismeri az eszközt
- A menüpontok kiválasztásához a számjegy után egy Enter-t is nyomjunk!
- Kiírás csak a 4. menüpontnál történik
- Kilépéshez a 0. menüpontot válasszuk!



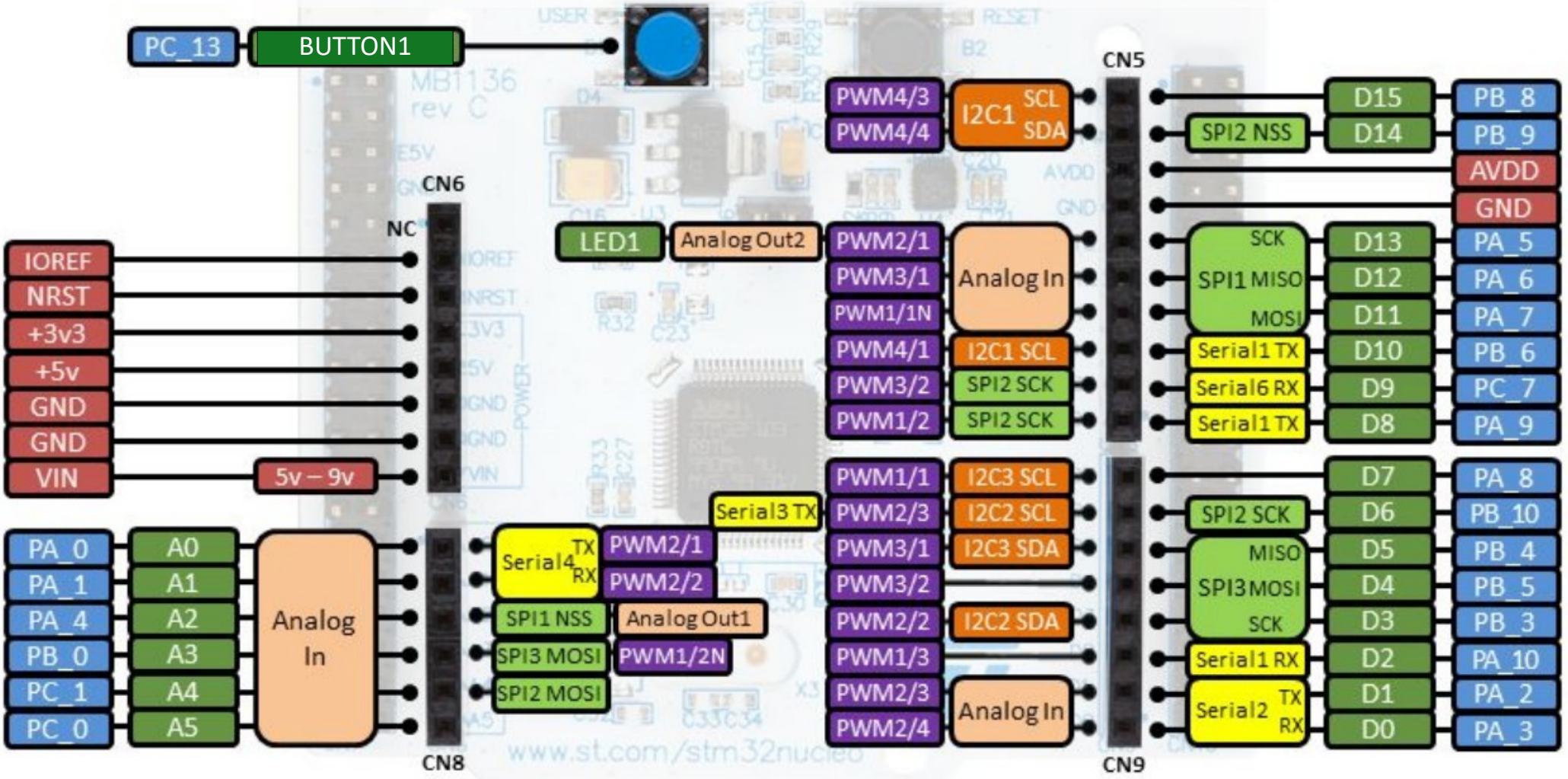
mbed6_usbHID_PWM

- Az **mbed6_usbHID_PWM** programot a 2022. június 16-i előadásban **Lab10_USBHID_demo** néven mutattuk be
- Akkor **mbed 2** környezetben, a Cypress AN82072 alkalmazási mintapéldájához igazítva készítettük el, hogy annak a letölthető grafikus PC alkalmazását használni tudjuk
- Természetesen az mbed 6-hoz igazított **mbed6_usbHID_PWM** programmal is használható a grafikus alkalmazás



Arduino kompatibilis kivezetések

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



Morpho kivezetések

Nucleo F446RE

Morpho Headers