**Blue-Velvet-Arena SW**

V02.01.002 / Petr Novák / novakpe@cvut.cz / 2022-09-20

Obsah

[1 Úvod 1](#_Toc114662691)

[2 Princip spojení částí aplikace 2](#_Toc114662692)

[2.1 Hlavní řízení přístupu / stavu externích zařízení 2](#_Toc114662693)

[2.2 Laserový modul 3](#_Toc114662694)

[2.3 Otáčení podlahy 3](#_Toc114662695)

[2.4 Nástěnné obrázky 4](#_Toc114662696)

[2.5 Dodatečné 4](#_Toc114662697)

[3 Projekty a jejich propojení / využití 5](#_Toc114662698)

# Úvod

V tomto dokumentu je popsána pouze část aplikace (pro PC) zajišťující ovládání externího HW. Celá aplikace pro BVA (testy) se skládá ze tří částí:

* (Miroslav Uller) Hlavního GUI a řízení / ovládání / průběh testu. Tato (hlavní) část není součástí tohoto dokumentu. (Simulována aplikací BVATest.)
* (Petr Novák) Část zajišťující komunikaci s externím HW na vyšší úrovni, tedy z pohledu hlavní aplikace. Tato vyšší část přistupuje ke každému externímu zařízení podle jeho symbolické / uživatelské činnosti v testu (motor, laser, obrázky). Tato část obsahuje vlastní GUI a lze ji spustit zcela samostatně pomocí **BVAService.exe** za účelem (pouze) testování činnosti připojených HW zařízení (nikoli však pro průběh testu). Případně lze její GUI vyvolat tlačítkem **Service** z hlavního GUI aplikace pro řízení testu (za účelem okamžitého / aktuálního testu některého připojeného HW).
* (Petr Novák) Část zajišťující komunikaci s externím zařízením na (nej)nižší úrovni, tedy z pohledu skutečného externě připojeného zařízení. Zde se ke každému zařízení přistupuje jako ke zcela univerzálnímu USB-HID (Human-Input-Device) zařízení a pouze se přenášejí data do / z externího zařízení (bez znalosti jejich významu). Tato část nerozeznává symbolickou / uživatelskou činnost externího zařízení, pouze přenos dat do / z externího zařízení podle jeho (načtené) identifikace a konfigurace (typy vstupů / výstupů). Tato část obsahuje zcela samostatné GUI a lze jej vyvolat tlačítkem **(Cmns)** na dialogu aplikace **BVAService.exe** (tedy z jejího GUI).

Druhá a třetí část jsou skutečně i samostatně spustitelné aplikace. V .NET Framework/Core v podstatě není rozdíl mezi aplikací a knihovnou. Z jedné aplikace lze volat / spouštět jinou aplikaci jako by to byla knihovna. Toto dává velkou schopnost vytvářet komplexní aplikace s využitím jiných aplikací jako externě volaných knihoven.

# Princip spojení částí aplikace

Aplikace **BVAService** poskytuje statickou třídu obsahující metody a data pro přístup ke všem (připojeným) HW zařízením. Vše se nachází v namespace nazvaném **BVAService.Devices** a třída má název **DevsCmns**. Její metody a vlastnosti jsou následující (před každou metodu a vlastností je tedy potřeba uvádět **BVAService.Devices.DevsCmns.,**  případně vytvořit **using static BVAService.Devices. DevsCmns** na začátku zdrojového souboru):

## Hlavní řízení přístupu / stavu externích zařízení

// zda je povolena / blokována komunikace s externím HW zařízením (výchozí je FALSE)

// vhodné pro testování aplikace bez připojených externích zařízení

(zápis / čtení) **bool EnableHW**

// nalezení všech připojených / dostupných USB-HID zařízeni

// návratová hodnota obsahuje seznam sériových čísel nalezených / připojených zařízení (pro kontrolu)

// (sériové číslo zařízení má 8 znaků)

// (samozřejmě mohou být nalezena i jiná než potřebná / požadovaná externí zařízení)

(volání) **string[] serialNumbers = FindAllDevice()**

// zobrazení dialogu obsahující stav nalezených / nenalezených zařízení

// řádky obsahují: ident zařízení, název, stav (OK / nenalezeno)

**Devices.DevsMain.ShowDevicesStatus();**

// pro test, zda zařízeni je požadované / potřebné lze využít vytvořený slovník

// pokud je sériové číslo v seznamu, jde o potřebné zařízení a je uveden jeho stručný název / typ

**Dictionary<string,string> DevsSerNumToName { get; set; } = new Dictionary<string, string>()**

**{**

**{ "00000011", "Laser" }, // Laser**

**{ "00000012", "Motor" }, // Motor**

**{ "00000021", "Anchor1" }, { "00000022", "Anchor2" }, { "00000023", "Anchor3" }, { "00000024", "Anchor4" }, { "00000025", "Anchor5" }, { "00000026", "Anchor6" }, { "00000027", "Anchor7" }, { "00000028", "Anchor8" } // Anchor(s)**

**};**

// například zjištění / výpisu pouze těch zařízení co nebyly nalezeny a jsou potřebná pro činnost

// seznam nenalezených / chybějících (ale potřebných) zařízení

**List<string> missingInfo = new List<string>()**

// projití skutečně všech potřebných (v programu evidovaných) zařízení

**foreach (string serialNumber in DevsSerNumToName.Keys)**

**{**

// pokud toto zařízení NENÍ mezi nalezenými, tak se jeho info vloží do seznamu nenalezených

**if (serialNumbers.Contains(serialNumber) == false)**

**{ missingInfo.Add(serialNumber + „ - “ + DevsSerNumToName[serialNumber]); }**

**}**

// **missingInfo** – obsahuje seznam jaká zařízení nebyla nalezena (sériové číslo + textový název)

// možno zobrazit pro obsluhu, aby zkontrolovala nenalezená zařízení (a spustila aplikaci znova)

// načtení všech dostupných obrázků

// vrátí pole obsahující názvy všech načtených obrázků

**string[] imagesNames = BVAService.Devices.DevsMain.LoadImages();**

// nastavení režimu (všech najednou) externích zařízení

// Disable – činnost je blokována (uvedeno do stavu jako po zapnutí) / Enable – činnost je povolena

// před využitím externích zařízení je potřeba je povolit (pomocí Enable)

// zda je při ukončení aplikace nastaveno Disable záleží zcela na požadavku programátora / uživatele

(zápis) **SetMode(EnMainAction.Enable / Disable)**

## Laserový modul

// zapnutí / vypnutí svitu laseru

(zápis) **LaserOnOff(EnLaserState.On / Off / Toggle)**

// nastavení pozice laseru na požadované cílové místo

// (využívají se pouze celá čísla, desetiny se zahazují)

(zápis) **LaserPosPercent(EnLaserAction.Position, (double)<-100;+100>, (double)<-100;+100>)**

// Pozor: po nastavení polohy serv/ laseru je potřeba počkat aspoň 300ms na nastavení polohy

// pro případnou možnost zápisu / čtení stavu laseru (zapnuto / vypnuto)

// čtení pouze poslední nastavený hodnoty aplikací, nikoli skutečný stav v externím HW

// (pokud možno nepoužívat, bude asi pouze private)

(zápis/čtení) **bool LaserOn**

// pro případnou možnost zápisu / čtení pozice X laseru

// čtení pouze poslední nastavený hodnoty aplikací, nikoli skutečný stav v externím HW

// (pokud možno nepoužívat, bude asi pouze private)

(čtení) **sbyte LaserPosX**

// pro případnou možnost zápisu / čtení pozice Y laseru

// čtení pouze poslední nastavený hodnoty aplikací, nikoli skutečný stav v externím HW

// (pokud možno nepoužívat, bude asi pouze private)

(čtení) **sbyte LaserPosY**

## Otáčení podlahy

// vykonávání pohybu podlahou nebo její zastavení

// Zero – nulování čísla z inkrementální čidla určující relativní otočení podlahy

(zápis) **FloorMove(EnFloorAction.Right / Left / Stop / Zero)**

// relativní otočení / natočení podlahy od posledního použití **FloorMove(EnFloorAction.Zero)**

// není ve stupních, ale v krocích inkrementálního snímače (aplikace převede na stupně po kalibraci)

// (pokud nebude stačit Int32, tak bude použito Int64)

(čtení) **Int32 FloorRotation**

// stav spínače nulové polohy při otáčení podlahy

(čtení) **bool FloorSwitchZero**

## Nástěnné obrázky

Obrázky musí být nejprve načteny.

// na HW zařízení s číslem (1-8) se zašlou požadovaná data obrázku se zadaným názvem

// (jde o název uložený v souboru obsahující obrázek, nikoli o název souboru s obrázkem)

// (vrací: 0 = OK / -1 = HW nepovolen / -2 = není takový obrázek / -3 = není takové zařízení)

(zápis) **int AnchorSetImage(int number, string imageName,**

**EnImageParms prms = EnImageParms.Set)**

public enum EnImageParms : byte

{

Set = 0 /\* původní obsah se vymaže, zobrazí se pouze tento celý obrázek \*/,

Or = 16 /\* pouze body nastavené v bitmapě na „1“ se zobrazí podle zadané barvy (přidají se k existujícímu obrázku), body nastavené na „0“ se zachovají \*/,

NonAnd = 32 /\* body nastavené v bitmapě na „1“ se vymažou (nebudou svítit), body nastavené na „0“ zachovají původní stav / barvu \*/,

NoShow = 128 /\* obrázek se ještě nyní nezobrazí (budou další data) \*/

}

// zápis na více / všechny „Anchor(s)“ současně

// přijímá **null** nebo pole **Tuple<number, imageName>**

// **anchorAndImage = null** – zhasnuti všech „Anchor(s)“

// **Tuple<number, imageName>** – na požadovaný „Anchor“ nastaví požadovaný obrázek

(zápis) **void AnchorSetImage(params Tuple<int, string>[] anchorAndImage)**

## Dodatečné

Z ovládací aplikace pro vykonání / průběh testu lze zobrazit tzv. servisní dialog pro možnost přímého testu / kontroly některého externího HW zařízení (v podstatě vyvolání aplikace **BVAService.exe**). Postup zobrazení tohoto dialogu je následující (BVA-Tests přestavuje hlavní aplikaci pro testy, BVA-Service představuje servisní část aplikace pro testování činnosti a komunikaci s externím HW zařízením):

// uchování odkazu na servisní / testovací dialog

// (globální proměnná v nějaké třídě hlavní aplikace)

**private BVAService.MainWindow bvaServiceWindow = null;**

// adresář kde jsou testovací / servisní data (například obrázky)

// umístěno někde v inicializaci hlavní aplikace

**Project.Global.GlobalConsts.PathDataAppBase = "HWTestData";**

// pokud dialog **BVA-Service** neexistuje, tak se vytvoří (bude se pouze skrývat, ne zavírat)

// umístěno v metodě pro vyvolání / zobrazení servisního dialogu

**if (bvaServiceWindow == null)**

// vytvoření dialogu „Service“, bude se pouze skrývat, je vytvářen z externí aplikace

// (FromExternalApp = true – nehledají se znova externí připojená zařízení)

**{ bvaServiceWindow = new BVAService.MainWindow() { NoCloseOnlyHide = true, FromExternalApp = true }; }**

// (znova) zobrazeni dialogu

**bvaServiceWindow.Show();**

// při uzavření **BVA-Test** se rovněž uzavře i **BVA-Service**

// umístěno v ukončení hlavní aplikace

**bvaServiceWindow.NoCloseOnlyHide = false;**

**if (bvaServiceWindow != null) { bvaServiceWindow.Close(); }**

# Projekty a jejich propojení / využití

Z hlediska programového využití jsou dostupné tři připravené projekty (všechny projekty jsou obsaženy v jednom „solution“ po názvem BVATests):

* **BVATests** – Minimalistická aplikace (náhrada za plnou hlavní aplikaci) demonstrující jak z hlavní aplikace spustit GUI pro **AppService**.
* **BVAService** – Aplikace sloužící pro testování externího HW a poskytující k němu přístup na vyšší úrovni. Rovněž obsahující všechny statické metody pro komunikaci s externím HW.
* **UniversalTestDebug** (bude přejmenováno) – Univerzální (v podstatě) aplikace komunikující s externím HW na (nej)nižší úrovni. Její GUI se zobrazí pomocí tlačítka **(Cmns)** v **AppService** (pokud je někdy potřeba, jde zobrazení uložených / přenášených dat).

Hlavní aplikace pro testy tedy bude **BVATests** (případně nějak jinak nazvaná).

Zdrojové soubory jsou v adresáři SRC. Struktura adresářů / projektů je následující:

**Projects-dotNETCore** – Hlavní adresář obsahující všechny projekty.

**BVA-BlueVelvetArena** – Podadresář obsahující projekty / aplikace **BVATests** a **BVAService**.

**BVATests** – Projekt pro (minimalistickou) aplikaci **BVATests**.

**BVAService** – Projekt pro aplikaci **BVAService**.

**911-UniversalTests** – Podadresář obsahující projekt **UniversalTestsDebug**.

Projekt **BVATests** je závislý na **BVAService**. Projekt **BVAService** je závislý na **UniversalTestsDebug**. Je vhodné mít všechny projekty v jednom „solution“ a nastavit tyto závislosti. Projekt **BVAService** lze nastavit jako hlavní / spustitelný a tím nepoužívat **BVATests**.

Při spuštění se vytvoří adresáře pro obrázky.

Změny

2023-04-05

* Přidána možnost posuvu / pohybu laseru po krocích ve směru vlevo/vpravo/nahoru/dolu.
* Přidány pojmenované pozice pro laser.

2022-09-21

* Možnost OR a AND pro obrázky (skládání obrázků).

2022-09-14

* „Init“ přejmenováno na „LoadImages“ (vrací pole obsahující jména načtených obrázků).
* Přidána metoda pro výpis stavu připojených / nepřipojených zařízení.
* Metoda **AnchorSetImage(…)** vrací chybu.
* Opraveno nastavení adresáře pro servisní data (nyní je použit „**HWTestData**“).