# Slow Control System

J. Demjančuková[1] and T.Poliak[2]

[1] Gymnázium Na Vítězné pláni, Praha
[2] Střední průmyslová škola, Trutnov

**Abstrakt**

V dnešní době existuje spousta způsobů, jak kontrolovat experimentální a průmyslová zařízení. Jedním z těchto způsobů je použití EPICS/Python Soft IOC, který v tomto projektu využíváme pro vytvoření aplikace na kontrolu a monitorování jednoduchého LED obvodu.

# 1 Úvod

Narozdíl od DAS (Data Acquisition System), který si zakládá na nahrávání a digitalizaci událostí v řádech milisekund, se Slow Control System používá pro kontrolu a monitorování technických zařízení v časových škálách v rozmezí několik sekund až po několik hodin. Často slouží ke kontrole a monitoringu komponent, jako jsou:

- vysokonapěťové moduly
- tlakoměry
- teplotní senzory
- RF detektory
- PID regulátory, atd.

# 2 Back-end

## 2.1 EPICS

Jednou z klíčových částí Slow Control System je **EPICS** (Experimental Physics and Industrial Control System)[2]. Jedná se o soubor softwarových nástrojů a aplikací, které poskytují infrastrukturu pro vybudování kontrolních systémů, a využívají ho například:

- částicové urychlovače (SLAC, iThemba, atd.)
- velké experimenty (Fermilab, LIGO, atd.)
- teleskopy (Gemini, Keckovy dalekohledy, atd.)

Takovéto systémy pak musí často spravovat desítky až stovky vzájemně propojených počítačů, mezi kterými probíhá komunikace. Tato komunikace může být monitorována a řízena z kontrolní místnosti nebo dokonce vzdáleně přes internetové připojení. Pro komunikaci mezi různými počítači používá EPICS techniku client/server a publish/subscribe, kdy jedna skupina počítačů sbírá data a takto posbíranou informaci pak předá jiné skupině přes Channel Access Network Protocol.

EPICS může interagovat se softwarem, který je schopný s tímto Channel Access protokolem komunikovat. V tom případě daný software většinou obsahuje rozšíření, která umožňují podporu Channel Access, což má např. Matlab, Python, Labview, Perl, atd. V našem případě jsme použili programovací jazyk Python, který obsahuje balíček **epics**. Tento balíček nabízí různé funkce, moduly a třídy pro interakci s Channel Access, stejně tak i PV objekt, který reprezentuje **Process Variable**.

## 2.2 Python Soft IOC

Problémem s balíčkem epics je fakt, že může s process variable jenom pracovat, ale není schopný ji vytvořit. Jedním z možných řešení je pak použít **Python Soft IOC**[3]. Jedná se o nástroj, který plně vytvoří a spustí EPICS IOC v rámci programovacího jazyka Python, a skládá se z:

- **pythonioc** (EPICS IOC provázaný s Python interpretem)
- **softioc** (přidružená knihovna)

Jednoduchá demonstrace Pythonu Soft IOC je znázorněna na obr.(1).

```
#!/home/lukas/Apps/epics/modules/pythonIoc/pythonIoc
# Import basic softioc framework
from softioc import softioc, builder

# Create PVs
builder.SetDeviceName('TEST')
builder.stringIn('01', initial_value = 'This is a test')

# Run the IOC.
# This is boilerplate, and must always be done in this order,
# and must always be done after creating all PVs.
builder.LoadDatabase()
softioc.iocInit()
softioc.interactive_ioc(globals())
```

Obrázek 1: Příklad kódu, který vytváří process variable

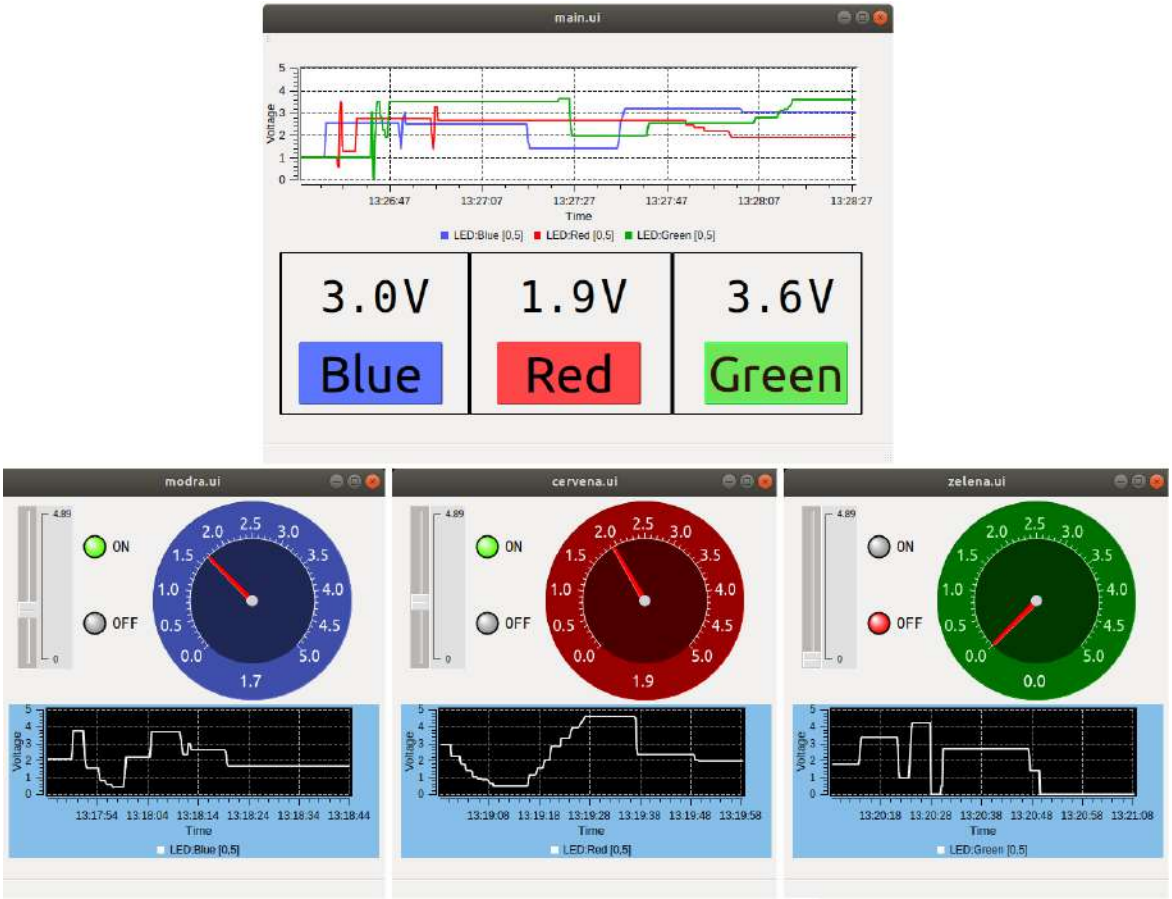
První řádek kódu obsahuje výše zmíněný pythonIoc, za kterým se skrývá EPICS IOC a prostředí Pythonu.

Na třetím řádku je vidět knihovnu softioc, pomocí které se vytváří daná process variable s názvem TEST:01.

# 3 Front-end

## 3.1 QTDesigner

Pro GUI (Graphic User Interface) jsme použili QTDesigner software. Jedná se o program ze skupiny Qt Tools, který pomocí QtWidgetů navrhuje a vybuduje GUI. Tento software pracuje na principu "what you see is what you get", kdy uživatel v podstatě nemusí psát žádný kód, a většinu aplikace vybuduje přetahováním a uspořádáváním widgetů, na které následně připojí už vytvořené process variable.

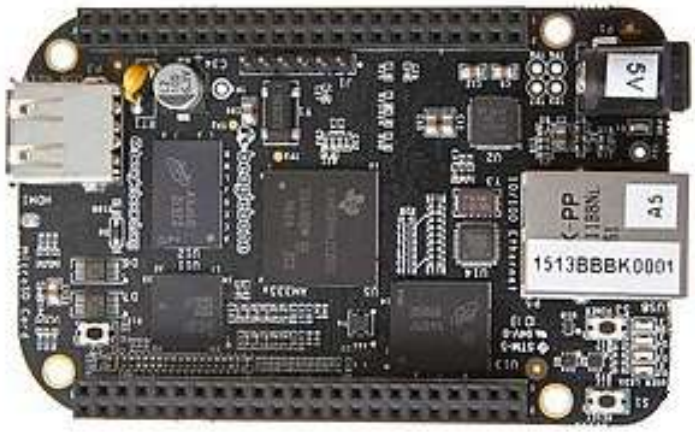



Obrázek 2: Front-end výsledné aplikace

# 4 BeagleBoard

BeagleBoard je nízkonapěťový open-source jednodeskový počítač od firmy Texas Instrument. Pracuje s open source softwarem v jeho OMAP3530 system-on-chipu a má stejné funkce jako základní počítač. OMAP3530 obsahuje ARM Cortex-A8 CPU, který může spustit Linux, Minix, FreeBSD, OpenBSD, RISC OS nebo Symbii. BeagleBoard má vestavěnou paměť 256 MB flash paměti na svém chipu a 256 MB paměti RAM. Vývojová deska může být napájená USB konektorem, nebo jiným zdrojem do 5 V. Protože je deska nízkonapěťová, nepotřebuje žádné chladiče. V našem projektu jsme použili pro ovládání LED diod vývojovou platformu Beaglebone Black,viz. obr.(3), což je jedna z verzí BeagleBoardu. Parametry BeagleBone Black jsou tyto:

- Frekvence(MHz) - 1000
- Pamět(SDRAM) - 512 MiB
- CPU - Cortex-A8 + Dual PRU
- GPU - PowerVR SGX530
- Rozměry - 86,4 x 54,3
- Audio výstup - Micro-HDMI
- Video výstup - Micro-HDMI
- USB port - 1 x Standard A host port
- Je schopný vyprodukovat napětí do 5 V a proud mezi 210 a 460 mA

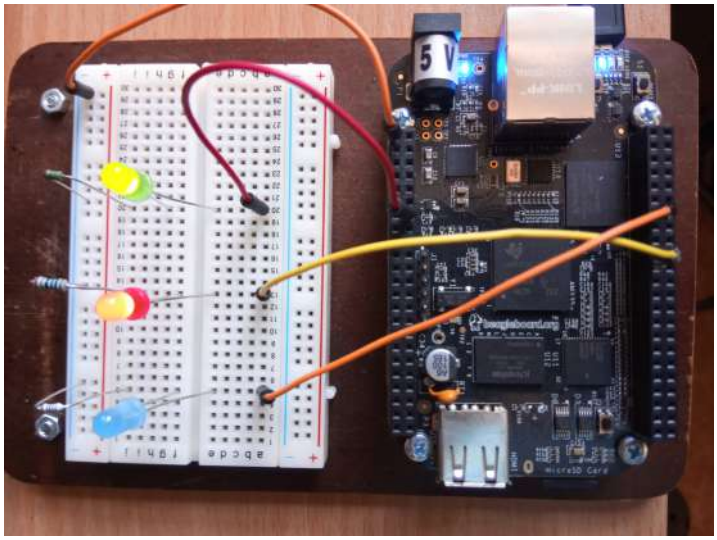



Obrázek 3: Beaglebone Black [1]

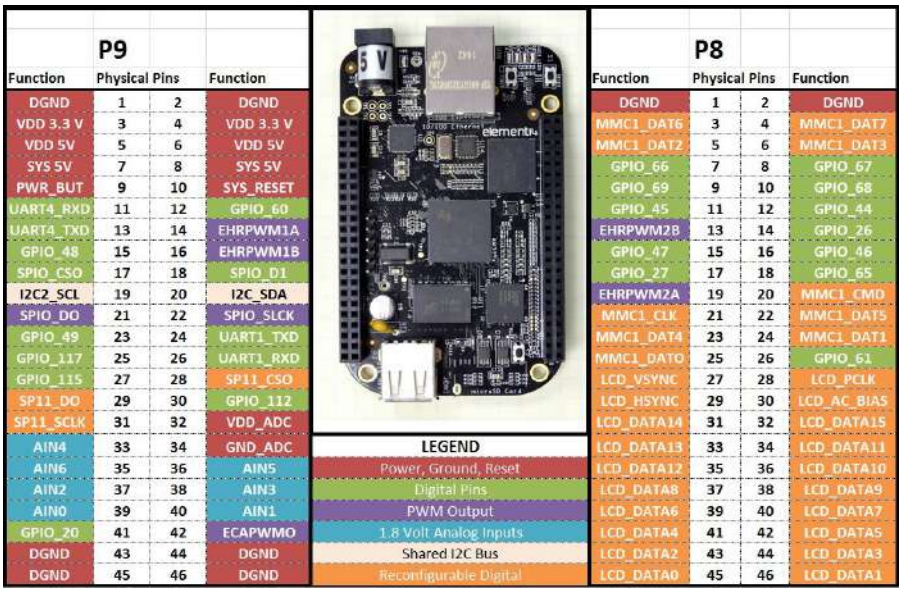
# 5 LED Obvod

Na obr.(4) je vidět zapojení tří LED diod do BeagleBone Black, se kterým jsme pracovali. Kladný pól LED diod jsme zapojili do PWM pinu v Beaglebone Black a záporný pól jsme zapojili přes rezistory v rozmezí od 51 Ω do 120 Ω do DGND (ground) pinu v Beaglebone Black. V obvodu koluje proud, který se chová podle Ohmova zákona

$$I = \frac{U}{R} \tag{1}$$



Obrázek 4: Zapojení obvodu LED diod a Beaglebone Black



| P9 | | | |
|---|---|---|---|
| Function | Physical Pins | | Function |
| DGND | 1 | 2 | DGND |
| VDD 3.3 V | 3 | 4 | VDD 3.3 V |
| VDD 5V | 5 | 6 | VDD 5V |
| SYS 5V | 7 | 8 | SYS 5V |
| PWR_BUT | 9 | 10 | SYS_RESET |
| UART4_RXD | 11 | 12 | GPIO_60 |
| UART4_TXD | 13 | 14 | EHRPWM1A |
| GPIO_48 | 15 | 16 | EHRPWM1B |
| SPIO_CSO | 17 | 18 | SPIO_D1 |
| I2C2_SCL | 19 | 20 | I2C_SDA |
| SPIO_DO | 21 | 22 | SPIO_SLCK |
| GPIO_49 | 23 | 24 | UART1_TXD |
| GPIO_117 | 25 | 26 | UART1_RXD |
| GPIO_115 | 27 | 28 | SP11_CSO |
| SP11_DO | 29 | 30 | GPIO_112 |
| SP11_SCLK | 31 | 32 | VDD_ADC |
| AIN4 | 33 | 34 | GND_ADC |
| AIN6 | 35 | 36 | AIN5 |
| AIN2 | 37 | 38 | AIN3 |
| AIN0 | 39 | 40 | AIN1 |
| GPIO_20 | 41 | 42 | ECAPWMO |
| DGND | 43 | 44 | DGND |
| DGND | 45 | 46 | DGND |

| LEGEND |
|---|
| Power, Ground, Reset |
| Digital Pins |
| PWM Output |
| 1.8 Volt Analog Inputs |
| Shared I2C Bus |
| Reconfigurable Digital |

| P8 | | | |
|---|---|---|---|
| Function | Physical Pins | | Function |
| DGND | 1 | 2 | DGND |
| MMC1_DAT6 | 3 | 4 | MMC1_DAT7 |
| MMC1_DAT2 | 5 | 6 | MMC1_DAT3 |
| GPIO_66 | 7 | 8 | GPIO_67 |
| GPIO_69 | 9 | 10 | GPIO_68 |
| GPIO_45 | 11 | 12 | GPIO_44 |
| EHRPWM2B | 13 | 14 | GPIO_26 |
| GPIO_47 | 15 | 16 | GPIO_46 |
| GPIO_27 | 17 | 18 | GPIO_65 |
| EHRPWM2A | 19 | 20 | MMC1_CMD |
| MMC1_CLK | 21 | 22 | MMC1_DAT5 |
| MMC1_DAT4 | 23 | 24 | MMC1_DAT1 |
| MMC1_DATO | 25 | 26 | GPIO_61 |
| LCD_VSYNC | 27 | 28 | LCD_PCLK |
| LCD_HSYNC | 29 | 30 | LCD_AC_BIAS |
| LCD_DATA14 | 31 | 32 | LCD_DATA15 |
| LCD_DATA13 | 33 | 34 | LCD_DATA11 |
| LCD_DATA12 | 35 | 36 | LCD_DATA10 |
| LCD_DATA8 | 37 | 38 | LCD_DATA9 |
| LCD_DATA6 | 39 | 40 | LCD_DATA7 |
| LCD_DATA4 | 41 | 42 | LCD_DATA5 |
| LCD_DATA2 | 43 | 44 | LCD_DATA3 |
| LCD_DATA0 | 45 | 46 | LCD_DATA1 |

Obrázek 5: Beaglebone Black pinout diagram[1]

# 6 Závěr

V rámci tohoto projektu jsme sestavili jednoduchý LED obvod, pro který jsme naprogramovali front-end a back-end aplikaci, díky které dokážeme tento LED obvod ovládat a monitorovat. Pro back-end jsme použili Python Soft IOC, který v sobě kloubí EPICS IOC a programovací jazyk Python. Front-end jsme vytvořili pomocí softwaru QtDesigner, který funguje na principu "what you see is what you get". Jako zdroj napětí pro LED obvod posloužil BeagleBone Black, který poskytuje napětí do 5 V.

# Poděkování

Rádi bychom poděkovali našemu odbornému garantovi Ing. Lukášovi Holubovi za předané znalosti, ochotu a trpělivost při vypracovávání tohoto projektu. Také bychom chtěli

poděkovat organizátorům TV@J 2019 za uskutečnění této akce.

# Reference

[1] *BeagleBone* beagleboard.org/

[2] *EPICS* https://epics.anl.gov/

[3] *Python Soft IOC* controls.diamond.ac.uk/downloads/python/pythonSoftIoc/2-10/html/pythonsoftioc.html