**Manuál balíku PyIRCA**

# Obsah

[Obsah 2](#_Toc106612336)

[1 Instalace 3](#_Toc106612337)

[1. Programovací manuál 4](#_Toc106612338)

[1.1. PyIRCA.IRCA2 a PyIRCA.IRCA3 4](#_Toc106612339)

[1.1.1. PyIRCA.IRCA3.Camera 4](#_Toc106612340)

[1.1.2. PyIRCA.IRCA3.CameraSettings 4](#_Toc106612341)

[1.1.3. PyIRCA.IRCA3.Header 4](#_Toc106612342)

[1.2. PyIRCA.Connections 5](#_Toc106612343)

[1.2.1. PyIRCA.Connections.WheelFW102 5](#_Toc106612344)

[1.2.2. PyIRCA.Connections.Mirror 5](#_Toc106612345)

[1.2.3. PyIRCA.Connections.BlackBodyFluke 5](#_Toc106612346)

[1.3. PyIRCA.Corrections 6](#_Toc106612347)

[1.4. PyIRCA.Hyper 6](#_Toc106612348)

[1.5. PyIRCA.Utils 6](#_Toc106612349)

# Instalace

Instalace pomocí balíku „[Anaconda](https://www.anaconda.com)“ („Jupyter notebook“ nebo „JupyterLab“)

Doplňkový balík „PyIRCA“ pro ovládání kamery. Další potřebné balíky jsou: scipy, scikit, numpy, matplotlib

Instalace prostředí IRCA:

conda env create -f irca.yml

Odkazování na balík PyIRCA je prozatím manuální na jednotlivé soubory, později bude vytvořen instalovatelný balík.

# Programovací manuál

## PyIRCA.IRCA2 a PyIRCA.IRCA3

### PyIRCA.IRCA3.Camera

Stejné funkce pro IRCA2 a IRCA3.

* Connect – Připojí se ke kameře na dané IP adrese
* Disconnect – Bezpečně se odpojí od kamery
* Write – Zapíše bitové pole do kamery
* Read – Přešte bitovou odpověď z kamery
* SimpleCommand – Provede jednoduchý příkaz (přenos hodnoty)
* ComplexCommand – Provede komplexní příkaz (přenos bytového pole)
* Calibration – provede kalibraci kamery
* SetUp – nastaví kameru podle zadaného CammeraSettings
* StartStream – zapne asynchronní stream z kamery
* StopStream – ukončí asynchronní stream z kamery
* GetImg – nasnímá zadaný počet snímků
* GetImgSubArea – nasnímá zadaný počet snímků a průběžně je ořezává
* GetImgSubPoint – nasnímá zadaný počet snímků a vybírá z nich pouze daný bod

### PyIRCA.IRCA3.CameraSettings

Nastavení GSK, GFID, INT, VBUS, VDET, AVG, OFF, GMS

### PyIRCA.IRCA3.Header

Dekódování XML hlavičky snímku (funkce DecodeXML)

* CameraID – označení kamery
* DNA – unikátní číslo kamery
* Firmware – verze FW
* HWResolutionX – rozlišení senzoru X
* HWResolutionY – rozlišení senzoru Y
* Minimum – minimální hodnota v obraze
* Maximum – maximální hodnota v obraze
* Sum – součet hodnot snímku
* ColoringMinimum – nastavené obarvovací minimum
* ColoringMaximum – nastavené obarvovací maximum
* FrameNumber – číslo snímku (od zapnutí kamery)
* TemperatureADCBol – teplota senzoru (RAW)
* BitsPerPixel – kódování hodnoty pixelu
* PixelByteStride – kódování snímku
* StartX – první pixel snímku, souřadnice X
* StartY – první pixel snímku, souřadnice Y
* Width – šířka snímku
* Height – výška snímku
* ByteSize – počet bytů na pixel
* ImageFlags – informace o zpracování snímku
* TriggerA – identifikace trigru A
* TriggerB – identifikace trigru B
* ResolvedTrigger – vyhodnocení trigrů,

## PyIRCA.Connections

### PyIRCA.Connections.WheelFW102

Ovládání otočného kola filtrů (6 pozic), komunikace COM,115200,8n1. Při inicializaci se definuje „home“ pozice, výchozí pozice 0).

* Connect – připojení na daný COM port
* GetID – získání identifikačního označení zařízení
* GetPosition – vrátí aktuální pozici kola filtrů (0-5)
* Close – odpojí se od COM portu
* SetPosition – nastaví pozici kola filtrů (0-5)

### PyIRCA.Connections.Mirror

Ovládání zrcadla hyperspektrální kamery, komunikace COM,115200,8n1. Při inicializaci se definují krajní pozice PosA a PosB a pozice černého tělesa PosH).

* Connect – připojení na daný COM port
* Close – odpojí se od COM portu
* SetSpeed – nastavuje rychlost pohybu zrcátka (0.3f)
* SetValue – nastavuje absolutní polohu zrcátka ve stupních (0.3f)
* GetValue – vrací aktuální pozici zrcátka
* GetSettedValue – vrací anstavenou pozici zrcátka
* SetValueProc – nastavuje procentuální pozici zrcátka (mezi PosA a posB)
* SetHome – nastaví zrcátko do kalibrační pozice PosH
* SetStart – nastaví zrcátko do výchozí pozice snímání (PosA, nebo PosB)

### PyIRCA.Connections.BlackBodyFluke

Ovládání kalibračního černého tělesa Fluke 4180, 4181

* Connect – připojení na daný COM port
* GetID – získání identifikačního označení zařízení
* SetEnable – zapíná/vypíná regulaci černého tělesa
* GetEnable – vrací stav regulace černého tělesa
* SetTemp – nastavuje žádanou teplotu tělesa
* GetTemp – vrací nastavenou žádanou teplotu tělesa
* GetActualTemp – vrací aktuální teplotu tělesa
* SetStableLim – nastavuje toleranci stability tělesa
* GetStableLim – vrací nastavenou toleranci stability tělesa
* IsStable – vrací informace o stabilitě tělesa (prodleva 30s od nastavení regulované hodnoty)
* WaitForStable – čeká na stabilizaci tělesa
* Close – odpojí se od COM portu

## PyIRCA.Corrections

Funkce pro práci s vadnými pixely a výpočet šumu senzoru.

* NETD\_MAP – výpočet mapy šumu senzoru
* DataIP – identifikace vadných pixelů mezi dvěma snímky
* CorrIP – korekce vadných pixelů nad daným snímkem
* Noise – výpočet a zobrazení šumu 1D, 2D a 3D metrik

## PyIRCA.Hyper

Funkce pro načítání uložené hyperspektrální kostky a metody pro výběr jednotlivých řezů s korekcí (GetLine, GetData).

## PyIRCA.Utils

Přidružené obecné pomocné funkce.

* Convolve – konvoluce dat zadanou maticí
* ShowStats – zobrazení 2D pole a jeho parametrů Min, Max, Mean, Var, Std …
* ShowGraphs – zobrazení 2D pole s osami
* Norm – normování 1D průběhu
* Decime – nahrazení N za sebou jdoucích prvků jejich průměrem
* LoadBin – načítání vektorů (vícerozměrných) z binárního formátu
* Linearize – vyrovná 1D vektor pomocí lineární funkce mezi prvním a posledním vzorkem
* MyHist – primitivní funkce výpočtu histogramu 2D pole
* recursive\_print – výpis elementů XML struktury
* ConvMatrix – databáze předdefinovaných konvolučních matic pro IRCA (Gauss24, Gauss8, Gauss16, Sharpen, Sharpen2, Sharpen3, Sharpen4, Edge, Shape, Robinson, Sobel, Kirsch, Prewitt, Nothing)

# Ukázky kódu

## Init

%run ..\\..\\..\\..\\PyIRCA\\PyIRCA\\Connections.py

%run ..\\..\\..\\..\\PyIRCA\\PyIRCA\\IRCA3.py

import os.path

import os

import numpy as np

import matplotlib.pyplot as plt

## Připojení

cam = Camera()

cam.Connect(Cam\_IP)

assert cam.Connected

bb = BlackBodyFluke(FlukePort)

bb.SetEnable(True)

## Záznam snímku

im,he = cam.GetImg(10,1)

im = np.array(im)

plt.figure(dpi=150)

plt.imshow(im.mean(0))

plt.show()

plt.figure(dpi=150)

plt.plot(MyHist(im.mean(0)))

plt.show()

## Měření černého tělesa s kontrolou saturace

MyPrint("Start at " + time.strftime("%H:%M:%S",time.localtime()))

for t in T:

bb.SetTemp(t)

time.sleep(10)

limT=abs(t)/400

if limT<0.01:

limT=0.01

MyPrint(str(t) + " °C +- " + str(limT))

MyPrint("Start at " + time.strftime("%H:%M:%S",time.localtime()))

bb.SetStableLim(limT)

IsStable=bb.WaitForStable(30)

MyPrint("Acquisition starts at " + time.strftime("%H:%M:%S",time.localtime()) + " Tact "+ str(bb.GetActualTemp()))

for cs in range(OldCS,len(CSs)):

SavePath=SaveDir+"\\" + GetName(cs,CSs[cs],t)

Images = []

Headers = []

it = 0

while len(Headers) != N and it <=maxIter:

if it > 0:

MyPrint("Repeat for " + GetName(cs,CSs[cs],t)+",cause N="+str(len(Headers))+",Nset="+str(N))

try:

Images,Headers = cam.GetImg(N,1)

except:

cam.ClearCommand()

MyPrint("Clear command for " + GetName(cs,CSs[cs],t))

it += 1

Images = np.array(Images)

np.save(SavePath,Images)

ImgXML = ET.Element("ImgXML")

tree = ET.ElementTree(ImgXML)

data = ET.SubElement(ImgXML, "data")

subel = ET.SubElement(data, 'Count')

subel.text = str(len(Headers))

MyPrint("Measured images for " + GetName(cs,CSs[cs],t) + ", N=" + subel.text)

for i in range(len(Headers)):

he = ET.fromstring(Headers[i].Source)

data.append(he)

config = ET.SubElement(ImgXML, "CameraSettings")

CSs[cs].EncodeXML(config)

tepl = ET.SubElement(ImgXML, "TeplotaBB")

tepl.text = str(t)

ET.indent(ImgXML)

configxml = ET.tostring(ImgXML)

with open(SavePath+".xml","wb") as f:

tree.write(f,xml\_declaration=True)

OldCS=cs

OldCS=0

MyPrint("Done at " + time.strftime("%H:%M:%S",time.localtime()))