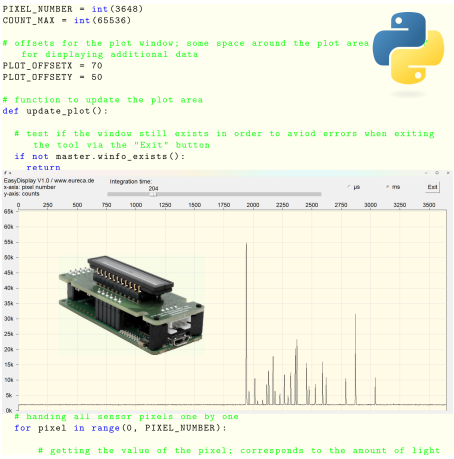
EasyDisplay-Code für unsere Zeilenkameras via Python

Das Programm *EasyDisplay* stellt eine Weiterentwicklung unseres bisherigen Programms *EasyAccess* dar. Es baut auf dem bestehenden System auf und integriert eine zusätzliche einfache grafische Darstellung der erfassten Sensordaten unserer Kameras vom Typ e9u-LSMD-TCD1304-STD. Durch die zweidimensionale Darstellung in einem Plot werden die Messdaten der jeweiligen Pixel übersichtlich visualisiert.



Besonders nützlich ist die zusätzliche Funktion, die es ermöglicht, die Integrationszeit flexibel zwischen Millisekunden und Mikrosekunden umzuschalten. Dadurch kann man die Erfassung der Messdaten präzise an die Anforderungen der jeweiligen Anwendung anpassen. Um eine bequeme und genaue Einstellung der Integrationszeit zu gewährleisten, wurde ein Schieberegler implementiert. Dieser erlaubt es dem Benutzer, die Integrationszeit relativ genau zu justieren und somit die gewünschte Aussteuerung des Sensors zu erzielen.

Das Programm wurde wieder in Python und mithilfe der externen DLL realisiert. Diese Kombination ermöglicht eine effiziente und benutzerfreundliche Programmierung, ist jedoch auch mit anderen Programmiersprachen umsetzbar. Das Programm ist mit Absicht so einfach wie möglich gehalten und soll lediglich die Funktionsweisen der verwendeten Elemente zeigen und so zur eigenen individuellen Weiterentwicklung anregen.

1 Vorbereitungen

1.1 Python installieren

Sie benötigen für dieses Programm eine funktionierende Python-Umgebung. Wie Sie diese gegebenenfalls installieren, entnehmen Sie bitte den Beschreibungen zu unserem Applikationsbeispiel zum *EasyAccess*-Programm.

1.2 Installation der verwendeten Python-Module

Das Programm nutzt das Python-Modul `Tkinter` für die Erzeugung der graphischen Elemente wie Buttons oder Schieberegler. Dieses muss gegebenenfalls noch nachinstalliert werden, z. B. von der Kommandozeile aus mit folgendem Befehl:

```
pip install tkinter
```

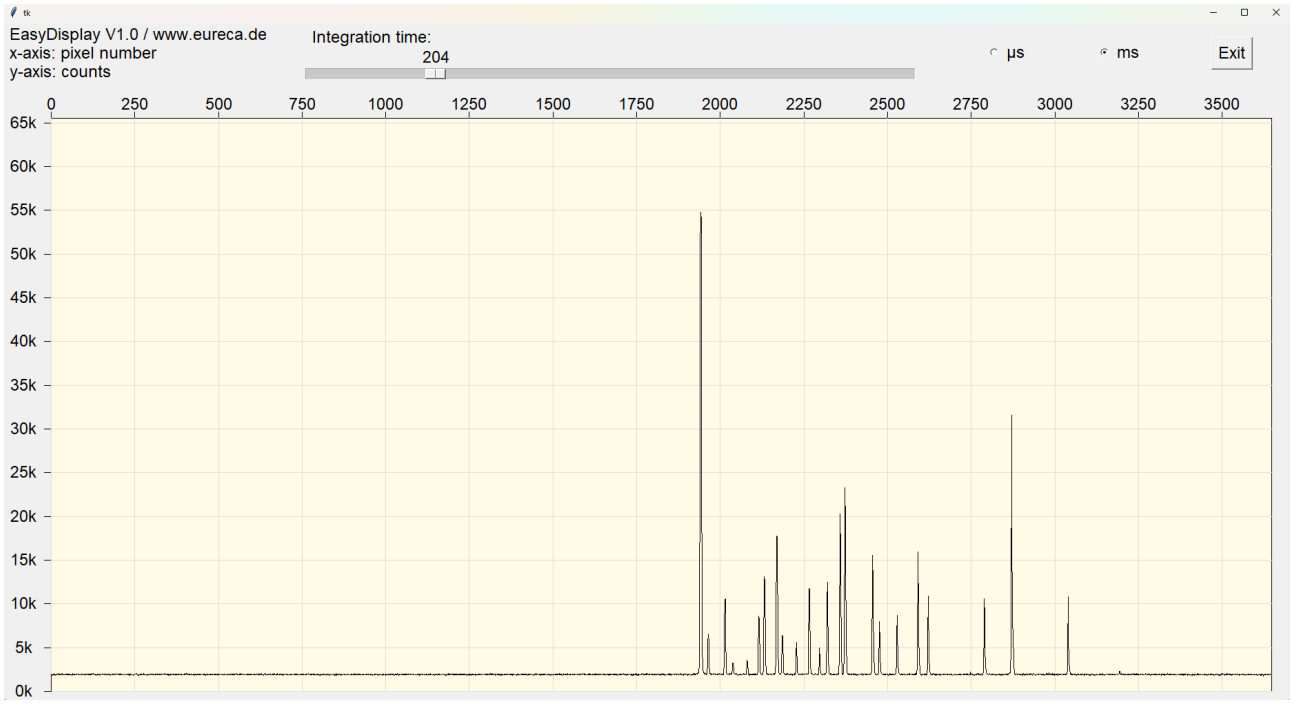
1.3 *EasyDisplay*-Code erhalten

Kopieren Sie den unten gezeigten *EasyDisplay*-Code von `https://www.eureca.de/EasyDisplay-de` und speichern ihn in eine neue Python-Datei mit z. B. dem Namen `EasyDisplay.py`.

2 Bedienung der Oberfläche

Nach Starten des Programms öffnet sich ein neues Fenster, welches fast den kompletten Bildschirm ausfüllt. Eine angeschlossene Kamera wird automatisch gesucht, initialisiert, das Auslesen in einer Endlos-Schleife gestartet und die erhaltenen Messwerte graphisch als x/y-Plot (Pixelmesswert/Pixelnummer) dargestellt. Als Startwert für die Integrationszeit sind 10 µs voreingestellt. Diese kann mittels eines Schiebereglers geändert werden. Mittels eines Radiobuttons kann umgeschaltet werden, ob der Wert des Schiebereglers als Mikro- oder Millisekunden interpretiert wird.

Die Integrationszeit ist somit für dieses Programm auf einen Maximalwert von 1000 ms beschränkt, obwohl die Kamera auch mit deutlich höheren Integrationszeiten betrieben werden kann. Bei Bedarf kann das Programm aber relativ einfach entsprechend erweitert werden. Zudem kann die Kamera auch mit unterschiedlichen Werten für `frame_time` und `exposure_time` arbeiten. Um das Progamm hier so einfach wie möglich zu halten wird für beide Werte der gleiche Wert des Schieberegles genutzt und allgemein von Integrationszeit gesprochen. Details hierzu entnehmen Sie bitte der ausführlichen Dokumentation unserer Kameras. Auch hier ist somit noch viel Potential für eine individuelle Erweiterung des Programms vorhanden.



Screenshot der graphischen Ausgabe für das Spektrum einer Glimmlampe, aufgenommen mit unserem Czerny-Turner-Spektrometer

3 *EasyDisplay* Python-Code

```
# EasyDisplay.py V1.1
#
# Python display tool as example code for line scan cameras of the e9u
   series by EURECA Messtechnik GmbH
# - detects and starts an attached camera in asychrone mode
# - reads out the camera and displays the recorded sensor data
# - the integration time can be adjusted between 1µs and 1000ms
# - extension of our EasyAccess example code
#
# For details please refer to: www.eureca.de

# import library for widgets
from tkinter import *

# import library for handling external camera DLL
import ctypes

# basic values of the used linear sensor; e.g. for e9u-LSMD-TCD1304-STD:
   3648 pixel; 16bit data
PIXEL_NUMBER = int(3648)
COUNT_MAX = int(65536)

# offsets for the plot window; some space around the plot area is needed
   for displaying additional data
PLOT_OFFSETX = 70
PLOT_OFFSETY = 50

# function to update the plot area
def update_plot():

  # test if the window still exists in order to aviod errors when exiting
     the tool via the "Exit" button
  if not master.winfo_exists():
    return

  # getting the integration time from the slider and using this value for
     the exposure time as well as for the frame time
  exposure_time = int(slider_exp_time.get()) * faktor.get()
  frame_time = exposure_time

  # reading out the camera; for details refer to the EasyAccess
     documentation
  libe9u.e9u_LSMD_set_times_us (0, exposure_time, frame_time)
  libe9u.e9u_LSMD_get_next_frame (0)

  # auxiliary variables for displaying the sensor data
  x_old = 0
  y_old = 0

  # deleting the old data points
  plot.delete("data")

  # handing all sensor pixels one by one
  for pixel in range(0, PIXEL_NUMBER):
```

```
        # getting the value of the pixel; corresponds to the amount of light
            by this pixel collected during the integration time
        counts = pointer[pixel]

        # scaling the sensor data to fit into the plot region
        x = int(pixel / PIXEL_NUMBER * plot_width)
        y = int(counts / COUNT_MAX * plot_height)

        # ploting the data point via connecting the current data with the
            last one
        plot.create_line(x_old + PLOT_OFFSETX,plot_height + PLOT_OFFSETY -
            y_old,x + PLOT_OFFSETX,plot_height + PLOT_OFFSETY - y, tags="data")

        x_old = x
        y_old = y

    # displaying the made changes in the plot area
    plot.update()

    # calling this funtion again after 1ms delay
    master.after(1, update_plot)


print("EasyCalibration V1.1\nSearching for camera: ")

# open external DLL
libe9u = ctypes.WinDLL('./libe9u_LSMD_x64.dll')

# define argument and return types for the used functions
libe9u.e9u_LSMD_search_for_camera.argtype = ctypes.c_uint
libe9u.e9u_LSMD_search_for_camera.restype = ctypes.c_int

libe9u.e9u_LSMD_start_camera_async.argtype = ctypes.c_uint
libe9u.e9u_LSMD_start_camera_async.restype = ctypes.c_int

libe9u.e9u_LSMD_set_times_us.argtypes = (ctypes.c_uint, ctypes.c_uint,
    ctypes.c_uint)
libe9u.e9u_LSMD_set_times_us.restype = ctypes.c_int

libe9u.e9u_LSMD_get_next_frame.argtype = ctypes.c_uint
libe9u.e9u_LSMD_get_next_frame.restype = ctypes.c_int

libe9u.e9u_LSMD_get_pixel_pointer.argtypes = (ctypes.c_uint, ctypes.c_uint)
libe9u.e9u_LSMD_get_pixel_pointer.restype = ctypes.POINTER(ctypes.c_uint16)

# Seach for a suitable camera on all USB ports and quit with returning the
    error code, if no camera is found
i_status = libe9u.e9u_LSMD_search_for_camera(0)
if i_status != 0:
    print("No camera found! Error Code: " + str(i_status))
    exit(1)

print("Starting camera: ", end='')
libe9u.e9u_LSMD_start_camera_async(0)
```

```
# getting the pointer to the array containing the sensor data
pointer = libe9u.e9u_LSMD_get_pixel_pointer(0, 0)

# defining the master window for graphical output
master = Tk()

# getting the size of the master window
screen_width = master.winfo_screenwidth()
screen_height = master.winfo_screenheight()

# setting the size of the display window to cover nearly the complete screen
master.geometry(str(screen_width - 50) + "x" + str(screen_height - 100) +
    "+10+20")

# defining the dimensions for the plot area
plot_width = screen_width - 150
plot_height = screen_height - 250

# defining and packing the control/output widgets
statusline = Frame(master)
statusline.pack(side='top')
plot = Canvas(master)
plot.pack(side ='bottom', fill=BOTH, expand=YES)

# output label for program name and version number
output_peak_width = Label(statusline, text="EasyDisplay V1.1 /
    www.eureca.de\nx-axis: pixel number\ny-axis: counts", justify=LEFT,
    font=("Arial Bold", 18))
output_peak_width.pack(side='left', padx=0)

# defining slider for integration time and setting it to 10
faktor = IntVar()
slider_exp_time = Scale(statusline, from_=1, to=1000, length=plot_width/2,
    orient=HORIZONTAL, label="Integration time:", font=("Arial Bold", 18))
slider_exp_time.pack(side='left', padx=50)
slider_exp_time.set(100)

# defining two radio buttons for switching the integration time between µs
    and ms
Radiobutton_us = Radiobutton(statusline, text="µs", font=("Arial Bold",
    18), variable=faktor, value=1)
Radiobutton_us.pack(side='left', padx=50)
Radiobutton_us.invoke()
Radiobutton_ms = Radiobutton(statusline, text="ms", font=("Arial Bold",
    18), variable=faktor, value=1000)
Radiobutton_ms.pack(side='left', padx=50)

# defining the exit button with the respective closing function
def close_window():
    master.destroy()
exit_button = Button(statusline, text="Exit", font=("Arial Bold", 18),
    command=close_window)
exit_button.pack(side='left', padx=50)

# drawing a rectangular frame for the plot area
Sensor_Plot = plot.create_rectangle(PLOT_OFFSETX, PLOT_OFFSETY, plot_width
```

```
    + PLOT_OFFSETX, plot_height + PLOT_OFFSETY, fill="#fffbe6")

# x-axis marking with pixel numbers
for x in range(15):
  plot.create_line(PLOT_OFFSETX + x*250*plot_width/PIXEL_NUMBER,
     PLOT_OFFSETY, PLOT_OFFSETX + x*250*plot_width/PIXEL_NUMBER,
     PLOT_OFFSETY - 10)
  plot.create_text(PLOT_OFFSETX + x*250*plot_width/PIXEL_NUMBER,
     PLOT_OFFSETY - 20,font=("Arial Bold", 18),text=int(x*250),fill='black')

  plot.create_line(PLOT_OFFSETX + x*250*plot_width/PIXEL_NUMBER,
     PLOT_OFFSETY, PLOT_OFFSETX + x*250*plot_width/PIXEL_NUMBER,
     PLOT_OFFSETY + plot_height, fill="#dddddd")

# y-axis marking with count numbers
for y in range(14):
  plot.create_line(PLOT_OFFSETX-10, PLOT_OFFSETY + plot_height -
     y*5000*plot_height/COUNT_MAX,
                   PLOT_OFFSETX, PLOT_OFFSETY + plot_height -
                      y*5000*plot_height/COUNT_MAX)
  text_output = str(y*5) + "k"
  plot.create_text(PLOT_OFFSETX-40, plot_height + PLOT_OFFSETY -
     y*5000*plot_height/COUNT_MAX,font=("Arial Bold",
     18),text=text_output,fill='black')

  plot.create_line(PLOT_OFFSETX + plot_width, PLOT_OFFSETY + plot_height -
     y*5000*plot_height/COUNT_MAX,
                   PLOT_OFFSETX, PLOT_OFFSETY + plot_height -
                      y*5000*plot_height/COUNT_MAX, fill="#dddddd")

# starting the first readout of the sensor and displaying the collected data
update_plot()

master.mainloop()
```

4 Abfangen von Fehlern

Aus Gründen der Übersichtlichkeit enthält der obige *EasyDisplay*-Code oft keine zusätzlichen Zeilen zur Fehlerbehandlung. Es wird jedoch empfohlen, z. B. nach jedem Aufruf einer DLL-Routine den zurückgegebenen Wert auszuwerten und eine Fehlerbehandlung durchzuführen, um den korrekten Betrieb der Kamera und/oder des Computersystems sicherzustellen. Ein Beispiel hierfür finden Sie wieder in unserer Beschreibung des Programms *EasyAccess*.

Tel: +49 (0) 221 / 952629 - 0
info@eureca.de
www.eureca.de

EURECA Messtechnik GmbH
Deutz-Kalker Str. 35
50679 Cologne, Germany