Ocean Optics USB4000 with *usb4java*

Bernard Panneton
December 2019

Introduction

The usb4java library[1] is used to communicate with a USB peripheral. It used to interact with an Ocean Optics Inc. USB4000 spectrometer.

This library was used to build a collection of R functions stored in ***playWith_usbjava.R*** file. The commands required to communicate with the spectrometer via the USB port are detailed in the device technical manual[2].

Functions in playWith_usb4java.R

init_usb()

Initialize usb device:

1. load required library
2. init JVM
3. Set Java class paths
4. Define some objects
5. Define a C helper function for converting littleEndian byte vector to integer

RETURN: a list of 4 components:

1. context: a Java object of class org.usb4java.Context
2. dlist: a Java object of class org.usb4java.DeviceList
3. libusb: a Java object of class org.usb4java.LibUsb
4. bufutils: a Java object of class org.usb4java.BufferUtils

find_usb <- function(product,vendor,usbObjects, silent=TRUE)

Given a vendor and a product ID number, find the corresponding USB device.

INPUTS:

- product: product ID number
- vendor: vendor ID number
- usbObjects: list returned by init_usb
- silent: when TRUE, no output at console.

OUTPUTS:

- a list with 2 components
    1. usbDevice: the device as obtained with dlist$get(as.integer(dev_no)) where dlist was defined in init_usb
    2. usbDescription: obtained by libusb$DeviceDescriptor(usbDevice, usbDescription)

get_OO_name_n_serial(usbObjects, usbDevice)

Function to get device name and version, device serial number and company names.

INPUTS:

[1] http://usb4java.org/

[2] USB4000-OEM-Data-Sheet.pdf stored in the **Doc** of the RSTudio project **OceanOptics_with_usb4java_in_R**.

- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- list with 3 components:
  1. name: name of the USB device
  2. version: name of device with version number
  3. serialno: serial number

getWavelengths(usbObjects, usbDevice)

To get the wavelength vector by reading the wavelength calibration coefficients

INPUTS:

- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- a vector of wavelengths

getMaxSatLevel(usbObjects, usbDevice){

Read the maximum saturation level from register

INPUTS:

- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- an integer giving the maximum saturation level.

jbyte_2_uint(x)

Takes a vector of Java bytes and interprets and 0:255. Required as Java bytes are signed.

INPUTS:

- x: vector of Java bytes as seen in R

OUTPUTS:

- a vector of value in the range 0:255, same length as input.

queryStatus <- function(usbObjects, usbDevice)

Query the USB device status.

INPUTS:

- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- a list of 5 elements:
  1. nb_pix: number of pixels in spectrum
  2. int_time: current integration time
  3. pack_in_spectra: number of data packets per spectrum

4. pack_count:
5. usb_speed: speed of USB transfer (“full” or “high”)

setIntegrationTime(temps,usbObjects, usbDevice)

Function to set spectrometer integration time.

INPUTS:

- temps: integration time in msec.
- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- none

boxcar(x, n = 5)

INPUTS:

- x: vector to smooth
- n: half width of averaging window. n elements on each side of middle value.

OUTPUTS:

- a smoothed vector.

getSpectrumusbObjects, usbDevice)

Function to retrieve a spectrum.

INPUTS:

- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- a spectrum as a numeric vector.

get_N_Spectrum(nspectra=2, usbObjects, usbDevice)

Function to retrieve a spectrum made as an average over a number of spectra.

INPUTS:

- nspectra: number of spectrum to average over.
- usbObjects: the list returned by init_usb()
- usbDevice: the list returned by c(find_usb(product,vendor,usbObjects,TRUE), get_command_set(product))

OUTPUTS:

- a spectrum as a numeric vector.

free_Device(usbObjects)

Function to free device. Required to cleanly end.

INPUTS:

- – usbObjects: the list returned by init_usb()

Example

Connect a USB4000 to a USB port and copy the following lines to the R console.

source("R/playWith_usb4java.R")

usbObjects <- init_usb()

product=0x1022 vendor=0x2457 usbDevice <- find_usb(product,vendor,usbObjects,TRUE) cmdList <- get_command_set(product) usbDevice <- c(usbDevice, cmdList)

name_serial <- get_OO_name_n_serial(usbObjects, usbDevice) lapply(name_serial, print)

wv <- getWavelengths(usbObjects, usbDevice)

statut <- queryStatus(usbObjects, usbDevice)

(getMaxSatLevel(usbObjects, usbDevice))

setIntegrationTime(100,usbObjects,usbDevice) (statut <- queryStatus(usbObjects, usbDevice))

dum <- getSpectrum(usbObjects, usbDevice)

plot(wv, dum[22:3669],type="l",col="lightgreen",lwd=5) lines(wv, boxcar(dum[22:3669],10), col="black",lwd=1) legend("topleft",legend=c("Raw","Boxcar"),lty=c(1,1),col=c("green","black"), inset = c(0.05,0.05))

windows() { ptm=proc.time() for (k in 1:20){ setIntegrationTime(10+k*5,usbObjects,usbDevice) dum <- getSpectrum(usbObjects, usbDevice) plot(wv, boxcar(dum[22:3669],5),type="l",col="red",lwd=2,ylim=c(0,50000)) } (proc.time()-ptm) }

dev.off()

plot(wv, dum[22:3669],type="l",col="red",lwd=2)

setIntegrationTime(100,usbObjects,usbDevice) { ptm=proc.time() sp <- get_N_Spectrum(nspectra=20, usbObjects, usbDevice) (proc.time()-ptm) }

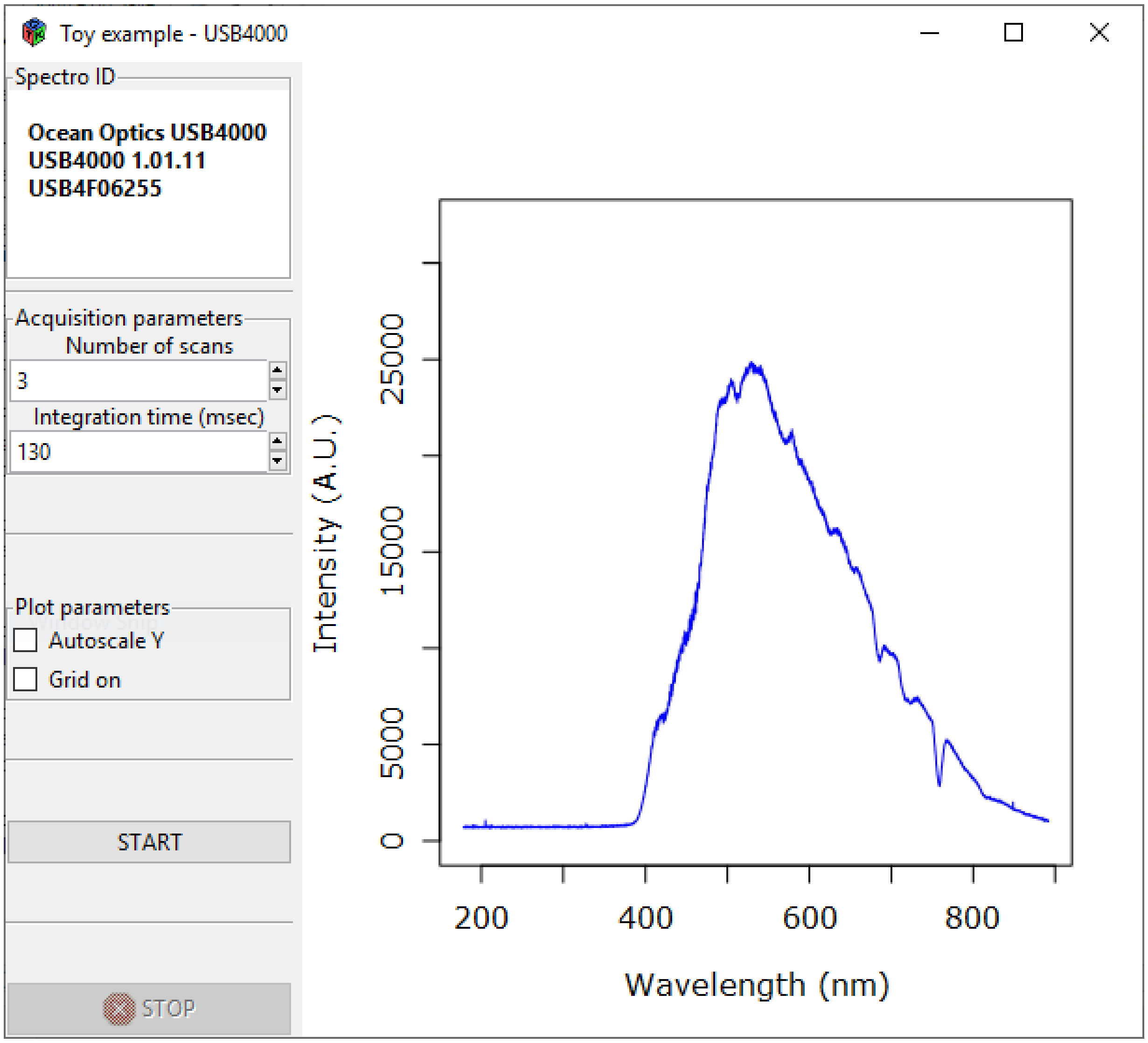
plot(wv,sp[22:3669],type="l",col="red",lwd=2, main = paste0(name_serial$name, " - Serial number : ", $name_serial$serialno), xlab = "Wavelength [nm]", ylab = "Intensity [A.U.]")

free_Device(usbObjects)

rm(list=ls())

Toy GUI

A little toy GUI implementing some of the functions is available in the file toGUI.R. Just source the file and run toyGUI with a USB4000 spectrometer plugged into a USB port.



Toy example - USB4000

Spectro ID

Ocean Optics USB4000
USB4000 1.01.11
USB4F06255

Acquisition parameters

Number of scans: 3

Integration time (msec): 130

Plot parameters

☐ Autoscale Y

☐ Grid on

Final words

The code in the R project works with a USB4000. Adaptation to other spectrometer should be fairly straigthforward. One needs to check the product ID and the exact syntax of device specific commands and parameters (the number of pixels in a spectrum, the number of packets transmitted over the USB. . .).