Listing 1

#include <QApplication>

#include <QThread>

#include <QMutex>

#include <QMessageBox>

#include <QColor>

#include <QLabel>

#include <QtDebug>

#include <QString>

#include <QPushButton>

#include "LeptonThread.h"

#include "MyLabel.h"

int main**(** int argc**,** char **\*\***argv **)**

**{**

// utworzenie nowej aplikacji

QApplication a**(** argc**,** argv **);**

QWidget **\***myWidget **=** **new** Qwidget**;**

// ustawienie pozycji (400, 300) i wielkości (340, 290) okna

myWidget**->**setGeometry**(**400**,** 300**,** 340**,** 290**);**

// utworzenie miejsca na obraz

QImage myImage**;**

myImage **=** QImage**(**320**,** 240**,** QImage**::**Format\_RGB888**);**

QRgb red **=** qRgb**(**255**,**0**,**0**);** // definicja koloru czerwonego w RGB

// górny lewy róg aplikacji wypełniony jest na czerwono

**for(**int i**=**0**;**i**<**80**;**i**++)** **{**

**for(**int j**=**0**;**j**<**60**;**j**++)** **{**

myImage**.**setPixel**(**i**,** j**,** red**);**

**}**

**}**

// utworzenie nowej etykiety i ustawienie jej obrazu w oknie

MyLabel myLabel**(**myWidget**);**

// ustawienie pozycji (10, 10) i wielkości (320, 240) w oknie

myLabel**.**setGeometry**(**10**,** 10**,** 320**,** 240**);**

myLabel**.**setPixmap**(**QPixmap**::**fromImage**(**myImage**));**

// utworzenie przycisku FFC

QPushButton **\***button1 **=** **new** QPushButton**(**"Perform FFC"**,** myWidget**);**

button1**->**setGeometry**(**320**/**2**-**50**,** 290**-**35**,** 100**,** 30**);**

// utworzenie osobnego wątku do zbierania danych po SPI

// gdy wątek zada sygnał updateImage, etykieta zaktualizuje się

LeptonThread **\***thread **=** **new** LeptonThread**();**

QObject**::**connect**(**thread**,** SIGNAL**(**updateImage**(**QImage**)),** **&**myLabel**,** SLOT**(**setImage**(**QImage**)));**

// połączenie przycisku FFC z wątkiem wykonującym akcję

QObject**::**connect**(**button1**,** SIGNAL**(**clicked**()),** thread**,** SLOT**(**performFFC**()));**

thread**->**start**();**

myWidget**->**show**();**

**return** a**.**exec**();**

**}**

Listing 2

void LeptonThread**::**run**()**

**{**

// utwórz początkowy obraz

myImage **=** QImage**(**80**,** 60**,** QImage**::**Format\_RGB888**);**

// otwórz port SPI

SpiOpenPort**(**0**);**

**while(true)** **{**

// odczytywanie pakietów danych z kamery poprzez SPI

int resets **=** 0**;**

**for(**int j**=**0**;**j**<**PACKETS\_PER\_FRAME**;**j**++)** **{**

// jeżeli odebrany pakiet jest typu "drop" musimy zresetować komunikację oraz

// ustawić j = -1, aby w kolejnej iteracji po inkrementacji było znowu zerem

read**(**spi\_cs0\_fd**,** result**+sizeof(**uint8\_t**)\***PACKET\_SIZE**\***j**,** **sizeof(**uint8\_t**)\***PACKET\_SIZE**);**

int packetNumber **=** result**[**j**\***PACKET\_SIZE**+**1**];**

**if(**packetNumber **!=** j**)** **{**

j **=** **-**1**;**

resets **+=** 1**;**

usleep**(**1000**);**

// 750 resetów to arbitralny limit, jako że nie powinno się nidgy wydarzyć, aby

// przerwa w transmisji była tak długa dla tej częstotliwości czytania z kamery

// Czytając z kamery częściej, okazać się może, że limit ten zostanie jednakże

// łatwo przekroczony, co zostanie od razu oznaczone, jako utrata synchronizacji

**if(**resets **==** 750**)** **{**

SpiClosePort**(**0**);**

usleep**(**750000**);**

SpiOpenPort**(**0**);**

**}**

**}**

**}**

**if(**resets **>=** 30**)** **{**

qDebug**()** **<<** "Zakończono odczyt, resetuje: " **<<** resets**;**

**}**

frameBuffer **=** **(**uint16\_t **\*)**result**;**

int row**,** column**;**

uint16\_t value**;**

uint16\_t minValue **=** 65535**;**

uint16\_t maxValue **=** 0**;**

**for(**int i**=**0**;**i**<**FRAME\_SIZE\_UINT16**;**i**++)** **{**

// pomijamy pierwsze 4 bajty każdej ramki (to nagłówekj) - są to dwiue wartości typu uint16\_t

**if(**i **%** PACKET\_SIZE\_UINT16 **<** 2**)** **{**

**continue;**

**}**

// zamiana MSB z LSB

int temp **=** result**[**i**\***2**];**

result**[**i**\***2**]** **=** result**[**i**\***2**+**1**];**

result**[**i**\***2**+**1**]** **=** temp**;**

value **=** frameBuffer**[**i**];**

**if(**value **>** maxValue**)** **{**

maxValue **=** value**;**

**}**

**if(**value **<** minValue**)** **{**

minValue **=** value**;**

**}**

column **=** i **%** PACKET\_SIZE\_UINT16 **-** 2**;**

row **=** i **/** PACKET\_SIZE\_UINT16 **;**

**}**

float diff **=** maxValue **-** minValue**;** // rozpiętość wartości

float scale **=** 255**/**diff**;** // skala do normalizacji danych na obrazie

QRgb color**;**

**for(**int i**=**0**;**i**<**FRAME\_SIZE\_UINT16**;**i**++)** **{**

**if(**i **%** PACKET\_SIZE\_UINT16 **<** 2**)** **{**

**continue;**

**}**

value **=** **(**frameBuffer**[**i**]** **-** minValue**)** **\*** scale**;**

const int **\***colormap **=** colormap\_ironblack**;** // mapa kolorów wykorzystana do obrazka

color **=** qRgb**(**colormap**[**3**\***value**],** colormap**[**3**\***value**+**1**],** colormap**[**3**\***value**+**2**]);**

column **=** **(**i **%** PACKET\_SIZE\_UINT16 **)** **-** 2**;**

row **=** i **/** PACKET\_SIZE\_UINT16**;**

myImage**.**setPixel**(**column**,** row**,** color**);**

**}**

// emisja sygnału dla interfejsu w Qt, który mówi, że można aktualizować obrazek

emit updateImage**(**myImage**);**

**}**

// finalnie - zamknięcie portu SPI

SpiClosePort**(**0**);**

**}**

Listing 3

#include "Lepton\_I2C.h"

// blbioteki oficjalnego SDK do kamer Lepton

#include "leptonSDKEmb32PUB/LEPTON\_SDK.h"

#include "leptonSDKEmb32PUB/LEPTON\_SYS.h"

#include "leptonSDKEmb32PUB/LEPTON\_Types.h"

bool \_connected**;**

LEP\_CAMERA\_PORT\_DESC\_T \_port**;**

// funkcja do połączenia się z kamerą

int lepton\_connect**()** **{**

LEP\_OpenPort**(**1**,** LEP\_CCI\_TWI**,** 400**,** **&**\_port**);**

\_connected **=** **true;**

**return** 0**;**

**}**

// funkcja uruchamiająca normalizację FFC w kamerze

void lepton\_perform\_ffc**()** **{**

**if(!**\_connected**)** **{**

lepton\_connect**();**

**}**

LEP\_RunSysFFCNormalization**(&**\_port**);**

**}**

// tutaj dodać można więcej funkcji z SDK