**Listing 1. Utworzenie wątku na potrzeby obsługi modułu kamery**

public class MainActivity **extends** Activity

**{**

**/.../**

private Handler cameraHandler**;**

private HandlerThread cameraThread**;**

@Override

protected void onCreate**(**Bundle savedInstanceState**)**

**{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**activity\_main**);**

**/.../**

/\* create new handler and associated thread for camera \*/

cameraThread **=** **new** HandlerThread**(**"CameraBackground"**);**

cameraThread**.**start**();**

cameraHandler **=** **new** Handler**(**cameraThread**.**getLooper**());**

**/.../**

**}**

/\*

\* onDestroy

\*/

@Override

protected void onDestroy**()** **{**

**super.**onDestroy**();**

cameraThread**.**quitSafely**();**

**}**

**}**

**Listing 2. Inicjalizacja modułu kamery**

private static final int IMAGE\_WIDTH **=** 640**;**

private static final int IMAGE\_HEIGHT **=** 480**;**

private CameraDevice mCameraDevice**;**

private CameraCaptureSession mCaptureSession**;**

private ImageReader mImageReader**;**

public void initializeCamera**(**Context context**,**

Handler backgroundHandler**,**

ImageReader**.**OnImageAvailableListener imageAvailableListener**)** **{**

/\* [A] Discover the camera instance \*/

CameraManager manager **=** **(**CameraManager**)** context**.**getSystemService**(**CAMERA\_SERVICE**);**

String**[]** camIds **=** **{};**

**try** **{**

camIds **=** manager**.**getCameraIdList**();**

**}** **catch** **(**CameraAccessException e**)** **{**

Log**.**d**(**TAG**,** "Cam access exception getting IDs"**,** e**);**

**}**

**if** **(**camIds**.**length **<** 1**)** **{**

Log**.**d**(**TAG**,** "No cameras found"**);**

**return;**

**}**

String id **=** camIds**[**0**];**

Log**.**d**(**TAG**,** "Using camera id " **+** id**);**

/\* (B) Initialize the image processor \*/

mImageReader **=** ImageReader**.**newInstance**(**IMAGE\_WIDTH**,** IMAGE\_HEIGHT**,** ImageFormat**.**JPEG**,** 1**);**

mImageReader**.**setOnImageAvailableListener**(**imageAvailableListener**,** backgroundHandler**);**

/\* (C) Open the camera resource \*/

**try** **{**

manager**.**openCamera**(**id**,** mStateCallback**,** backgroundHandler**);**

**}** **catch** **(**CameraAccessException cae**)** **{**

Log**.**d**(**TAG**,** "Camera access exception"**,** cae**);**

**}**

**}**

**Listing 3. Obiekt wywołań zwrotnych do monitorowania stanu modułu kamery**

private final CameraDevice**.**StateCallback mStateCallback **=** **new** CameraDevice**.**StateCallback**()** **{**

@Override

public void onOpened**(**CameraDevice cameraDevice**)** **{**

Log**.**d**(**TAG**,** "Opened camera."**);**

mCameraDevice **=** cameraDevice**;**

**}**

@Override

public void onDisconnected**(**CameraDevice cameraDevice**)** **{**

Log**.**d**(**TAG**,** "Camera disconnected, closing."**);**

cameraDevice**.**close**();**

**}**

@Override

public void onError**(**CameraDevice cameraDevice**,** int i**)** **{**

Log**.**d**(**TAG**,** "Camera device error, closing."**);**

cameraDevice**.**close**();**

**}**

@Override

public void onClosed**(**CameraDevice cameraDevice**)** **{**

Log**.**d**(**TAG**,** "Closed camera, releasing"**);**

mCameraDevice **=** **null;**

**}**

**};**

**Listing 4. Obiekt wywołań zwrotnych do monitorowania stanu modułu kamery**

public void takePicture**()** **{**

**if** **(**mCameraDevice **==** **null)** **{**

Log**.**w**(**TAG**,** "Cannot capture image. Camera not initialized."**);**

**return;**

**}**

**try** **{**

mCameraDevice**.**createCaptureSession**(**

Collections**.**singletonList**(**mImageReader**.**getSurface**()),**

mSessionCallback**,** **null);**

**}** **catch** **(**CameraAccessException cae**)** **{**

Log**.**d**(**TAG**,** "access exception while preparing pic"**,** cae**);**

**}**

**}**

**Listing 5. Kod funkcji triggerImageCapture()**

private void triggerImageCapture**()** **{**

**try** **{**

final CaptureRequest**.**Builder captureBuilder **=**

mCameraDevice**.**createCaptureRequest**(**CameraDevice**.**TEMPLATE\_STILL\_CAPTURE**);**

captureBuilder**.**addTarget**(**mImageReader**.**getSurface**());**

captureBuilder**.**set**(**CaptureRequest**.**CONTROL\_AE\_MODE**,** CaptureRequest**.**CONTROL\_AE\_MODE\_ON**);**

Log**.**d**(**TAG**,** "Session initialized."**);**

mCaptureSession**.**capture**(**captureBuilder**.**build**(),** mCaptureCallback**,** **null);**

**}** **catch** **(**CameraAccessException cae**)** **{**

Log**.**d**(**TAG**,** "camera capture exception"**);**

**}**

**}**

**Listing 6. Kod klasy MainActivity uzupełniony o obsługę modułu kamery**

public class MainActivity **extends** Activity

**{**

private Gpio button**;**

private RPiCamera camera**;**

private Handler cameraHandler**;**

private HandlerThread cameraThread**;**

**/.../**

@Override

protected void onCreate**(**Bundle savedInstanceState**)**

**{**

**super.**onCreate**(**savedInstanceState**);**

setContentView**(**R**.**layout**.**activity\_main**);**

**/.../**

/\* configure button \*/

PeripheralManagerService service **=** **new** PeripheralManagerService**();**

**try** **{**

button **=** service**.**openGpio**(**"BCM23"**);**

button**.**setDirection**(**Gpio**.**DIRECTION\_IN**);**

button**.**setEdgeTriggerType**(**Gpio**.**EDGE\_FALLING**);**

button**.**registerGpioCallback**(**ButtonCallback**);**

**}** **catch** **(**IOException e**)** **{**

Log**.**e**(**TAG**,** "PeripheralIO API ERROR"**,** e**);**

**}**

**/.../**

/\* create new handler and associated thread for camera \*/

cameraThread **=** **new** HandlerThread**(**"CameraBackground"**);**

cameraThread**.**start**();**

cameraHandler **=** **new** Handler**(**cameraThread**.**getLooper**());**

/\* initialize camera \*/

camera **=** RPiCamera**.**getInstance**();**

camera**.**initializeCamera**(this,** cameraHandler**,** mOnImageAvailableListener**);**

**}**

/\*

\* Button Callback

\*/

private GpioCallback ButtonCallback **=** **new** GpioCallback**()** **{**

@Override

public boolean onGpioEdge**(**Gpio gpio**)** **{**

camera**.**takePicture**();**

**return** **true;**

**}**

**};**

/\*

\* Listener for new camera images

\*/

private ImageReader**.**OnImageAvailableListener mOnImageAvailableListener **=**

**new** ImageReader**.**OnImageAvailableListener**()** **{**

@Override

public void onImageAvailable**(**ImageReader reader**)** **{**

Image image **=** reader**.**acquireLatestImage**();**

ByteBuffer imageBuf **=** image**.**getPlanes**()[**0**].**getBuffer**();**

final byte**[]** imageBytes **=** **new** byte**[**imageBuf**.**remaining**()];**

imageBuf**.**get**(**imageBytes**);**

image**.**close**();**

updateGUI**(**imageBytes**);**

**}**

**};**

**/.../**

**}**