**Listing 1. Otworzenie pliku urządzenia /dev/i2c-1 oraz ustawienie adresu I2C\_SLAVE**

#define GYRO\_ADDR 0x6b

int main **(**void**)**

**{**

int i2c\_fd**,** ret**;**

/\* open i2c device \*/

i2c\_fd **=** open **(**"/dev/i2c-1"**,** O\_RDWR**);**

**if** **(**i2c\_fd **<** 0**)**

**{**

printf **(**"Failed to open the i2c bus\n"**);**

**return** EXIT\_FAILURE**;**

**}**

/\* set slave address \*/

ret **=** ioctl **(**i2c\_fd**,** I2C\_SLAVE**,** GYRO\_ADDR**);**

**if** **(**ret **<** 0**)**

**{**

printf **(**"Failed to acquire bus access and/or talk to slave\n"**);**

**goto** exit**;**

**}**

**}**

**Listing 2. Inicjalizacja układu L3GD20 poprzez zapis rejestrów CTRL\_REG[X]**

#define AUTO\_INCREMENT 0x80

static int gyro\_init **(**int i2c\_fd**)**

**{**

unsigned char init\_seq**[**6**];**

init\_seq**[**0**]** **=** **(**CTRL\_REG1 **|** AUTO\_INCREMENT**);**

init\_seq**[**1**]** **=** 0xCF**;** /\* CTRL\_REG1: normal mode, xyz enable \*/

init\_seq**[**2**]** **=** 0x01**;** /\* CTRL\_REG2: <default value> \*/

init\_seq**[**3**]** **=** 0x00**;** /\* CTRL\_REG3: <default value> \*/

init\_seq**[**4**]** **=** 0x80**;** /\* CTRL\_REG4: 250dps, Block Data Update \*/

init\_seq**[**5**]** **=** 0x02**;** /\* CTRL\_REG5: <default value> \*/

**if** **(**write **(**i2c\_fd**,** init\_seq**,** 6**)** **!=** 6**)** **return** **-**1**;**

**return** 0**;**

**}**

**Listing 3. Funkcja odczytu prędkości kątowych w osiach X, Y i Z**

static int gyro\_get\_xyz **(**int i2c\_fd**,** float **\***x**,** float **\***y**,** float **\***z**)**

**{**

unsigned char reg\_addr **=** OUT\_X\_L **|** AUTO\_INCREMENT**;**

unsigned char reg\_data**[**6**];**

int ret**;**

struct i2c\_msg messages**[]** **=**

**{**

**{**

GYRO\_ADDR**,**

0**,**

**sizeof(**reg\_addr**),**

**&**reg\_addr

**},**

**{**

GYRO\_ADDR**,**

I2C\_M\_RD**,**

**sizeof(**reg\_data**),**

reg\_data

**}**

**};**

struct i2c\_rdwr\_ioctl\_data packets **=**

**{**

messages**,**

**sizeof(**messages**)** **/** **sizeof(**struct i2c\_msg**)**

**};**

ret **=** ioctl **(**i2c\_fd**,** I2C\_RDWR**,** **&**packets**);**

**if** **(**ret **<** 0**)** **return** ret**;**

**\***x **=** **(**short**)** **(**reg\_data**[**0**]** **+** **((**short**)**reg\_data**[**1**]** **<<** 8**));**

**\***y **=** **(**short**)** **(**reg\_data**[**2**]** **+** **((**short**)**reg\_data**[**3**]** **<<** 8**));**

**\***z **=** **(**short**)** **(**reg\_data**[**4**]** **+** **((**short**)**reg\_data**[**5**]** **<<** 8**));**

**return** 0**;**

**}**

**Listing 4. Kompletny kod źródłowy aplikacji gyro-i2c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#include <linux/i2c-dev.h>

#include <linux/i2c.h>

#include <sys/time.h>

#define WHO\_AM\_I 0x0F

#define CTRL\_REG1 0x20

#define CTRL\_REG2 0x21

#define CTRL\_REG3 0x22

#define CTRL\_REG4 0x23

#define CTRL\_REG5 0x24

#define REFERENCE 0x25

#define OUT\_TEMP 0x26

#define STATUS\_REG 0x27

#define OUT\_X\_L 0x28

#define OUT\_X\_H 0x29

#define OUT\_Y\_L 0x2A

#define OUT\_Y\_H 0x2B

#define OUT\_Z\_L 0x2C

#define OUT\_Z\_H 0x2D

#define FIFO\_CTRL\_REG 0x2E

#define FIFO\_SRC\_REG 0x2F

#define INT1\_CFG 0x30

#define INT1\_SRC 0x31

#define INT1\_TSH\_XH 0x32

#define INT1\_TSH\_XL 0x33

#define INT1\_TSH\_YH 0x34

#define INT1\_TSH\_YL 0x35

#define INT1\_TSH\_ZH 0x36

#define INT1\_TSH\_ZL 0x37

#define INT1\_DURATION 0x38

#define GYRO\_ADDR 0x6b

#define AUTO\_INCREMENT 0x80

int x\_low **=** 0**,** y\_low **=** 0**,** z\_low **=** 0**;**

int x\_high **=** 0**,** y\_high **=** 0**,** z\_high **=** 0**;**

static unsigned long get\_timestamp **()**

**{**

struct timeval tv**;**

gettimeofday **(&**tv**,NULL);**

**return** tv**.**tv\_sec **\*** 1000000UL **+** tv**.**tv\_usec**;**

**}**

static int gyro\_init **(**int i2c\_fd**)**

**{**

unsigned char init\_seq**[**6**];**

init\_seq**[**0**]** **=** **(**CTRL\_REG1 **|** AUTO\_INCREMENT**);**

init\_seq**[**1**]** **=** 0xCF**;** /\* CTRL\_REG1: normal mode, xyz enable \*/

init\_seq**[**2**]** **=** 0x01**;** /\* CTRL\_REG2: <default value> \*/

init\_seq**[**3**]** **=** 0x00**;** /\* CTRL\_REG3: <default value> \*/

init\_seq**[**4**]** **=** 0x80**;** /\* CTRL\_REG4: 250dps, Block Data Update \*/

init\_seq**[**5**]** **=** 0x02**;** /\* CTRL\_REG5: <default value> \*/

**if** **(**write **(**i2c\_fd**,** init\_seq**,** 6**)** **!=** 6**)** **return** **-**1**;**

**return** 0**;**

**}**

static int gyro\_get\_status **(**int i2c\_fd**)**

**{**

unsigned char reg\_addr **=** STATUS\_REG**;**

unsigned char reg\_data**[**1**];**

int ret**;**

struct i2c\_msg messages**[]** **=**

**{**

**{**

GYRO\_ADDR**,**

0**,**

**sizeof(**reg\_addr**),**

**&**reg\_addr

**},**

**{**

GYRO\_ADDR**,**

I2C\_M\_RD**,**

**sizeof(**reg\_data**),**

reg\_data

**}**

**};**

struct i2c\_rdwr\_ioctl\_data packets **=**

**{**

messages**,**

**sizeof(**messages**)** **/** **sizeof(**struct i2c\_msg**)**

**};**

ret **=** ioctl **(**i2c\_fd**,** I2C\_RDWR**,** **&**packets**);**

**if** **(**ret **<** 0**)** **return** ret**;**

**return** **(**reg\_data**[**0**]** **&** **(**1 **<<** 3**));**

**}**

static int gyro\_get\_xyz **(**int i2c\_fd**,** float **\***x**,** float **\***y**,** float **\***z**)**

**{**

unsigned char reg\_addr **=** OUT\_X\_L **|** AUTO\_INCREMENT**;**

unsigned char reg\_data**[**6**];**

int ret**;**

struct i2c\_msg messages**[]** **=**

**{**

**{**

GYRO\_ADDR**,**

0**,**

**sizeof(**reg\_addr**),**

**&**reg\_addr

**},**

**{**

GYRO\_ADDR**,**

I2C\_M\_RD**,**

**sizeof(**reg\_data**),**

reg\_data

**}**

**};**

struct i2c\_rdwr\_ioctl\_data packets **=**

**{**

messages**,**

**sizeof(**messages**)** **/** **sizeof(**struct i2c\_msg**)**

**};**

ret **=** ioctl **(**i2c\_fd**,** I2C\_RDWR**,** **&**packets**);**

**if** **(**ret **<** 0**)** **return** ret**;**

**\***x **=** **(**short**)** **(**reg\_data**[**0**]** **+** **((**short**)**reg\_data**[**1**]** **<<** 8**));**

**\***y **=** **(**short**)** **(**reg\_data**[**2**]** **+** **((**short**)**reg\_data**[**3**]** **<<** 8**));**

**\***z **=** **(**short**)** **(**reg\_data**[**4**]** **+** **((**short**)**reg\_data**[**5**]** **<<** 8**));**

**return** 0**;**

**}**

static int gyro\_calib **(**int i2c\_fd**)**

**{**

float x\_raw**,** y\_raw**,** z\_raw**;**

int ret**;**

**for** **(**int i **=**0 **;** i **<** 200 **;** i**++)**

**{**

**while** **(!**gyro\_get\_status **(**i2c\_fd**));**

ret **=** gyro\_get\_xyz **(**i2c\_fd**,** **&**x\_raw**,** **&**y\_raw**,** **&**z\_raw**);**

**if** **(**ret **<** 0**)** **break;**

**if** **(**x\_raw **>** x\_high**)** x\_high **=** x\_raw**;**

**else** **if** **(**x\_raw **<** x\_low**)** x\_low **=** x\_raw**;**

**if** **(**y\_raw **>** y\_high**)** y\_high **=** y\_raw**;**

**else** **if** **(**y\_raw **<** y\_low**)** y\_low **=** y\_raw**;**

**if** **(**z\_raw **>** z\_high**)** z\_high **=** z\_raw**;**

**else** **if** **(**z\_raw **<** z\_low**)** z\_low **=** z\_raw**;**

**}**

**return** ret**;**

**}**

int main **(**void**)**

**{**

int i2c\_fd**,** ret**;**

float x\_raw**,** y\_raw**,** z\_raw**;**

unsigned long pt **=** 0**;**

/\* actual angles \*/

float angX **=** 0**;**

float angY **=** 0**;**

float angZ **=** 0**;**

/\* previous angles for calculation \*/

float p\_angX **=** 0**;**

float p\_angY **=** 0**;**

float p\_angZ **=** 0**;**

/\* open i2c device \*/

i2c\_fd **=** open **(**"/dev/i2c-1"**,** O\_RDWR**);**

**if** **(**i2c\_fd **<** 0**)**

**{**

printf **(**"Failed to open the i2c bus\n"**);**

**return** EXIT\_FAILURE**;**

**}**

/\* set slave address \*/

ret **=** ioctl **(**i2c\_fd**,** I2C\_SLAVE**,** GYRO\_ADDR**);**

**if** **(**ret **<** 0**)**

**{**

printf **(**"Failed to acquire bus access and/or talk to slave\n"**);**

**goto** exit**;**

**}**

/\* gyro init \*/

ret **=** gyro\_init **(**i2c\_fd**);**

**if** **(**ret **<** 0**)**

**{**

printf **(**"gyro\_init error!\n"**);**

**goto** exit**;**

**}**

/\* gyro calib \*/

puts **(**"Calibration..."**);**

ret **=** gyro\_calib **(**i2c\_fd**);**

**if** **(**ret **<** 0**)**

**{**

printf **(**"gyro\_calib error!\n"**);**

**goto** exit**;**

**}**

**while** **(**1**)**

**{**

**while** **(!**gyro\_get\_status **(**i2c\_fd**));**

/\* read xyz raw values \*/

gyro\_get\_xyz **(**i2c\_fd**,** **&**x\_raw**,** **&**y\_raw**,** **&**z\_raw**);**

/\* get timestamp \*/

unsigned long ct **=** get\_timestamp**();**

**if** **(**pt **==** 0**)**

**{**

pt **=** get\_timestamp**();**

**continue;**

**}**

float dt **=** **(**float**)** **(**ct **-** pt**)** **/** 1000000.0**;**

pt **=** get\_timestamp**();**

/\* x-axis \*/

**if** **(**x\_raw **>=** x\_high **||** x\_raw **<=** x\_low**)**

**{**

angX **+=** **((**p\_angX **+** **(**x\_raw **\*** 0.00875**))/**2**)** **\*** dt**;**

p\_angX **=** x\_raw **\*** 0.00875**;**

**}**

**else** p\_angX **=** 0**;**

/\* y-axis \*/

**if** **(**y\_raw **>=** y\_high **||** y\_raw **<=** y\_low**)**

**{**

angY **+=** **((**p\_angY **+** **(**y\_raw **\*** 0.00875**))/**2**)** **\*** dt**;**

p\_angY **=** y\_raw **\*** 0.00875**;**

**}**

**else** p\_angY **=** 0**;**

/\* z-axis \*/

**if** **(**z\_raw **>=** z\_high **||** z\_raw **<=** z\_low**)**

**{**

angZ **+=** **((**p\_angZ **+** **(**z\_raw **\*** 0.00875**))/**2**)** **\*** dt**;**

p\_angZ **=** z\_raw **\*** 0.00875**;**

**}**

**else** p\_angZ **=** 0**;**

printf **(**"%.1f %.1f %.1f\n"**,** angX**,** angY**,** angZ**);**

fflush **(**stdout**);**

**}**

exit**:**

**return** EXIT\_FAILURE**;**

**}**

**Listing 5. Budowanie sceny z wykorzystaniem biblioteki Three.js**

function init**()**

**{**

scene **=** new THREE**.**Scene**();**

camera **=** new THREE**.**PerspectiveCamera **(**70**,** 500**/**500**,** 0.01**,** 10**);**

camera**.**position**.**z **=** 0.5**;**

geometry **=** new THREE**.**BoxGeometry **(**0.2**,** 0.2**,** 0.2**);**

material **=** new THREE**.**MeshNormalMaterial**();**

mesh **=** new THREE**.**Mesh **(**geometry**,** material**);**

scene**.**add **(**mesh**);**

renderer **=** new THREE**.**WebGLRenderer **({** canvas**:** mycanvas**});**

renderer**.**setSize **(**500**,** 500**);**

document**.**body**.**appendChild **(**renderer**.**domElement**);**

**}**

**Listing 6. Kod funkcji wykonującej obrót obiektu**

function animate**()**

**{**

requestAnimationFrame **(**animate**);**

mesh**.**rotation**.**x **=** THREE**.**Math**.**degToRad**(**x**);**

mesh**.**rotation**.**y **=** THREE**.**Math**.**degToRad**(**y**);**

mesh**.**rotation**.**z **=** THREE**.**Math**.**degToRad**(**z**);**

renderer**.**render **(**scene**,** camera**);**

**}**

**Listing 7. Pełny kod źródłowy strony index.html po dodaniu elementów grafiki 3D**

**<!**DOCTYPE html**>**

**<**html**>**

**<**head**>**

**<**canvas id**=**"mycanvas" width**=**"500" height**=**"500"**></**canvas**>**

**<**style**>**

table**,** th**,** td **{**

border**:** 1px solid black**;**

**}**

th**,** td **{**

border**:** 1px solid black**;**

padding**:** 15px**;**

**}**

**</**style**>**

**<**script src**=**'/socket.io/socket.io.js'**></**script**>**

**<**script src**=**'three.min.js'**></**script**>**

**<**script**>**

var camera**,** scene**,** renderer**;**

var geometry**,** material**,** mesh**;**

var x**,** y**,** z**;**

function init**()** **{**

scene **=** new THREE**.**Scene**();**

camera **=** new THREE**.**PerspectiveCamera **(**70**,** 500**/**500**,** 0.01**,** 10**);**

camera**.**position**.**z **=** 0.5**;**

geometry **=** new THREE**.**BoxGeometry **(**0.2**,** 0.2**,** 0.2**);**

material **=** new THREE**.**MeshNormalMaterial**();**

mesh **=** new THREE**.**Mesh **(**geometry**,** material**);**

scene**.**add **(**mesh**);**

renderer **=** new THREE**.**WebGLRenderer **({** canvas**:** mycanvas**});**

renderer**.**setSize **(**500**,** 500**);**

document**.**body**.**appendChild **(**renderer**.**domElement**);**

**}**

function animate**()** **{**

requestAnimationFrame **(**animate**);**

mesh**.**rotation**.**x **=** THREE**.**Math**.**degToRad**(**x**);**

mesh**.**rotation**.**y **=** THREE**.**Math**.**degToRad**(**y**);**

mesh**.**rotation**.**z **=** THREE**.**Math**.**degToRad**(**z**);**

renderer**.**render **(**scene**,** camera**);**

**}**

init**();**

animate**();**

var socket **=** io**();**

socket**.**on **(**'xyz'**,** function **(**data**)** **{**

var arr **=** data**.**message**.**split**(**" "**);**

x **=** arr**[**0**];**

y **=** arr**[**1**];**

z **=** arr**[**2**];**

document**.**getElementById**(**"x\_val"**).**innerHTML **=** x**;**

document**.**getElementById**(**"y\_val"**).**innerHTML **=** y**;**

document**.**getElementById**(**"z\_val"**).**innerHTML **=** z**;**

**});**

**</**script**>**

**</**head**>**

**<**body**>**

**<**h1**>**Gyroscope I2C**</**h1**>**

**<**table**>**

**<**tr**>**

**<**th**>**X **[**deg**]</**th**>**

**<**td**><**p id**=**"x\_val"**>---</**p**></**td**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Y **[**deg**]</**th**>**

**<**td**><**p id**=**"y\_val"**>---</**p**></**td**>**

**</**tr**>**

**<**tr**>**

**<**th**>**Z **[**deg**]</**th**>**

**<**td**><**p id**=**"z\_val"**>---</**p**></**td**>**

**</**tr**>**

**</**table**>**

**</**body**>**

**</**html**>**

**Listing 8. Zmodyfikowany kod obsługi zapytania – skrypt main.js**

var url **=** require**(**'url'**);**

var server **=** http**.**createServer **(**function handler **(**request**,** response**)**

**{**

var pathname **=** url**.**parse**(**request**.**url**).**pathname**;**

console**.**log**(**"Request for " **+** pathname **+** " received."**);**

response**.**writeHead **(**200**,** **{**'Content-Type'**:** 'text/html'**});**

**if(**pathname **==** "/"**)**

**{**

var index **=** fs**.**readFileSync **(**\_\_dirname **+** '/index.html'**);**

response**.**write **(**index**);**

**}** **else** **if** **(**pathname **==** "/three.min.js"**)**

**{**

var script **=** fs**.**readFileSync **(**\_\_dirname **+** '/three.min.js'**);**

response**.**write **(**script**);**

**}**

response**.**end**();**

**});**