POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka (INF)
SPECJALNOŚĆ: Internet Engineering (INE)

PRACA DYPLOMOWA MAGISTERSKA

Smartfon z systemem Android
jako wysokopoziomowy sterownik robota

Android smartphone
as a high-level controller of a robot

AUTOR:
Michał Kowalski

PROWADZĄCY PRACĘ:
dr inż. Marek Woda

OCENA PRACY:

WROCŁAW 2016

Contents

Chapter 1

Introduction

1.1 Description of problem

Nowadays, popularity of robots is on the raise. It's not hard to built a simple one, and Internet is full of tutorials how to build them. They are built using specially programmed microcontrollers (MCUs), and simplest ones even without any. However, MCUs have some limitations:

1. They have limited memory and computational capability.
2. They require a lot of low-level configuration and programming.
3. Each MCU model requires (at least) slightly different configuration.
4. It's hard to look for help (e.g. on Stack Overflow, [7]) for specific MCU.

Therefore, usage of Android smartphones as high-level controllers, sending commands to MCU as low-level one, should be worth considering, because of:

1. Lot of memory and powerful processors.
2. Many built-in sensors.
3. Many ways to communicate with surroundings, especially - with MCU.
4. Popularity of Android platform:
   - tutorials,
   - devices,
   - solutions on Stack Overflow,
   - external libraries.
5. Compatibility between smartphones and Android versions.
6. High-level programming and reduced low-level configuration.

1.2 Goal of a project

Goal of this project is to check, if Android smartphone:

- can communicate with microcontroller,
- can extend functionality of robots using its built-in sensors.

Found solutions should be analyzed with attention to:

- compatibility,
- performance,
- difficulty of implementation.

Two Android smartphones (with different performance and Android version) will be used: Sony Ericsson Xperia Neo and Motorola Moto G LTE. They will communicate with MCU through USB cable, and face detection using built-in camera will be used as an example of extending MCU's capabilities - it requires both computing power and sensors not available in MCU, and there exists several ways to implement this.

As MCU, a Freescale FRDM KL26Z will be used. [1] is a blog dedicated to development on Freescale platform (mostly KL25Z, predecessor of KL26Z), and even contains an article how to built a mobile robot on that platform (img. 1.1). It has articles how to use most of those MCUs features, however from smartphone's point of view, only communication using USB port (KL25Z and KL26Z have two of them) is required.

Because of already working solutions on Freescale's MCUs (described in State-of-Art section), there is no need to build an actual, working robot - it's proved, that those MCUs can be used as a controlling unit of a robot. Therefore, this thesis focuses only on making use of smartphone for detecting position of face (as in 1.5, but with Android instead of PC) and sending text command to MCU (also as in 1.5, but on Android, not PC).

1.3 Content of thesis

This thesis contains:

1. Introduction - this chapter.
2. Platforms - chapter describing MCU, Android, and used equipment.
3. Communication - chapter about communication between smartphone and MCU.
4. Sensors - chapter about extending robot's capabilities by using one of smartphone's sensors - camera.
5. Summary - chapter with conclusions, whether Android smartphone can be used as high-level controller.
6. Bibliography.

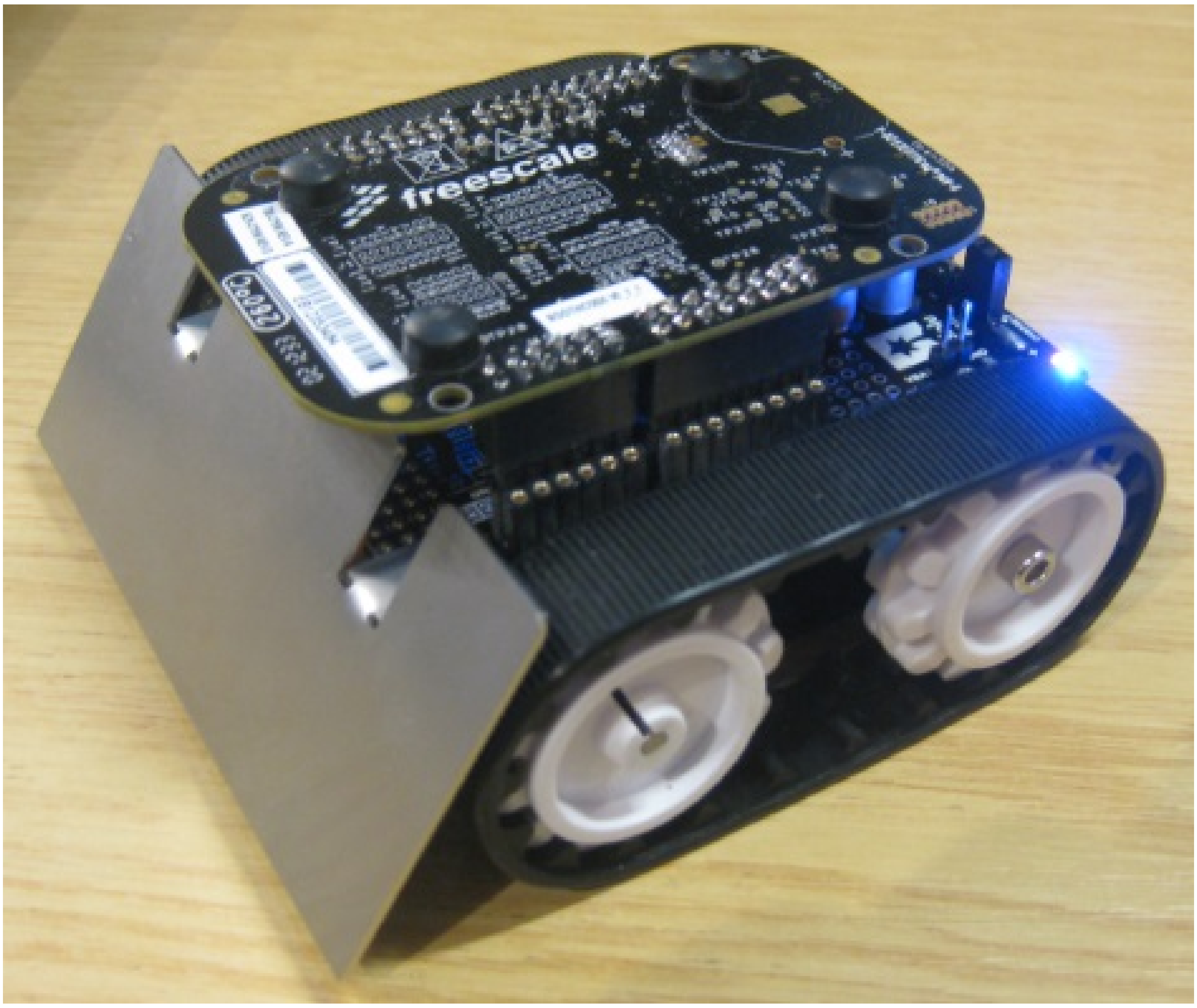

Figure 1.1 FRDM Zumo Robot, [1]

1.4 State of Art

State of Art can be divided into three different areas:

1. Robots working on Freescale's MCU.

2. Robots controlled by mobile phones.

3. "Face Followers".

1.4.1 Robots working on Freescale's MCU

Good example of such robot can be Freedom Zumo Robot, described in one of posts on [1] (img. 1.1). It's built on FRDM KL25Z, and Zumo Robot Kit for Arduino - KL25Z (and KL26Z probably too) is compatible with it.

Another example could be a robot built for one of previous projects - it will be described later. Therefore, it's proven, that this MCU can be used for building (at least simple) robots.

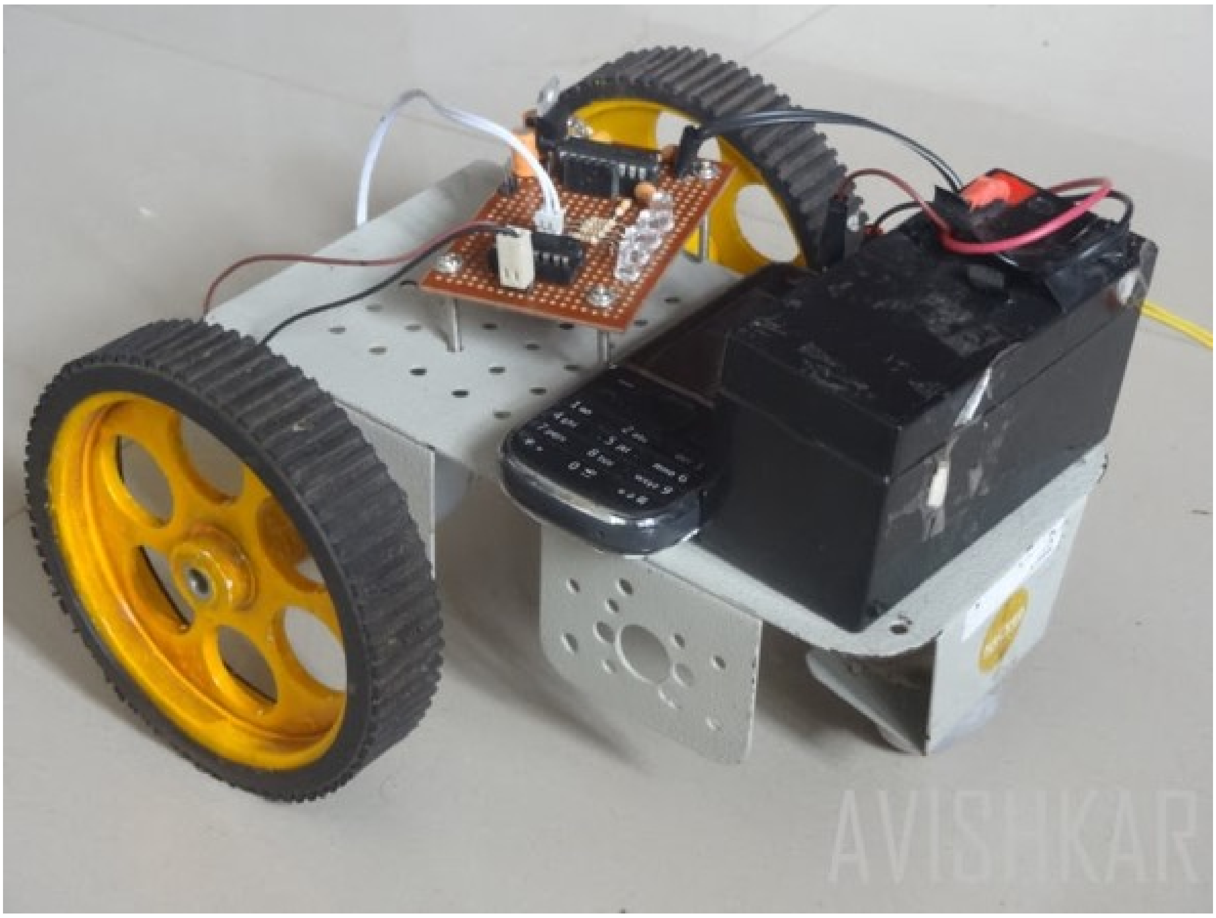

Figure 1.2 Mobile Controlled Robot by Ganeev Singh, [2]

1.4.2 Robots controlled by mobile phones

Next category are robots controlled by mobile phones, like ones shown on fig. 1.2, 1.3 and 1.4. It seems, that all of them all remotely-controlled using another phone - clicking on button during phone call with receiver generates a sound, which is received and transformed into signal in headphones' port. None of examples have any (additional) logic in phone.

Some examples of robots controlled by additional program on smartphones still exist, but most of them either are poorly documented, or are using smartphone only as a remote, not part of the robot.

The best example was line-following robot developed by co-students Krzysztof Taborski and Kamil Szyc, but it wasn't published anywhere. Part of their code was even used as one of methods to communicate between Android and MCU.

1.4.3 "Face Followers"

Figure 1.3 Mobile Controlled Robot by Robotics Bible, [3]

Figure 1.4 Mobile Controlled Robot by Mayoogh Girish, [4]

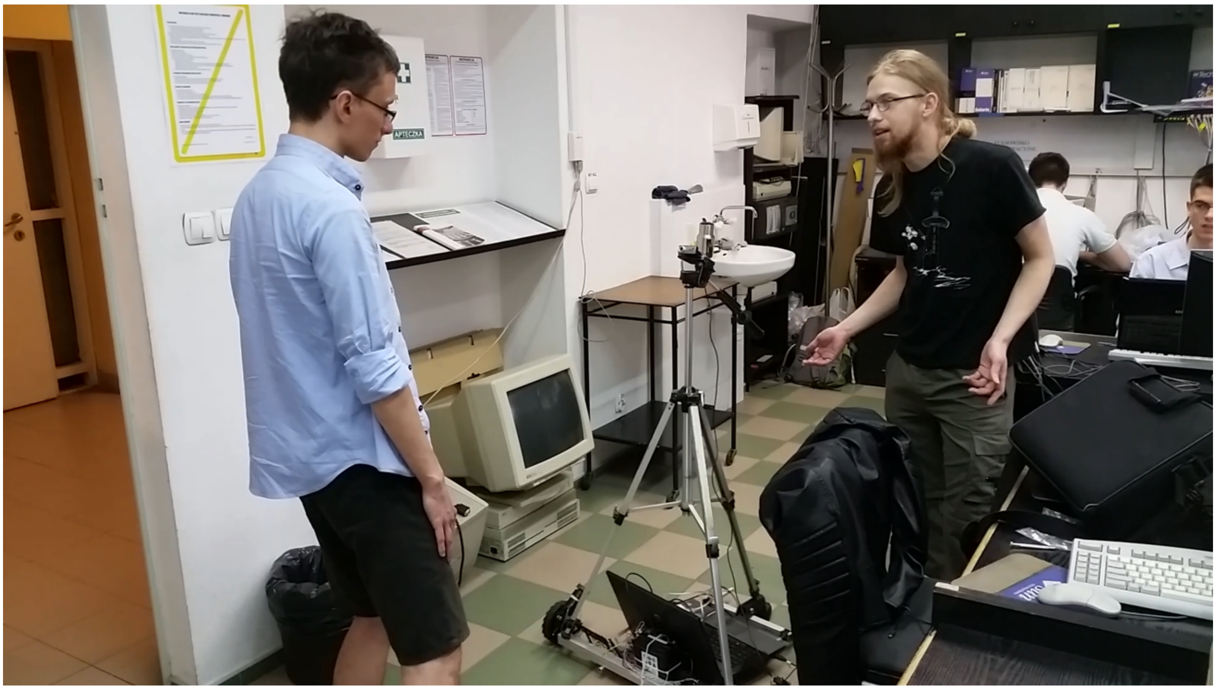

Figure 1.5 FaceFollower by Michał Kowalski and Adam Ćwik

Chapter 2

Platforms

2.1 Android

2.2 MCU

2.2.1 General info

2.2.2 Communication through UART

2.2.3 Communication through CDC

Figure 2.1 Sony Ericsson Xperia Neo

Figure 2.2 Motorola Moto G LTE

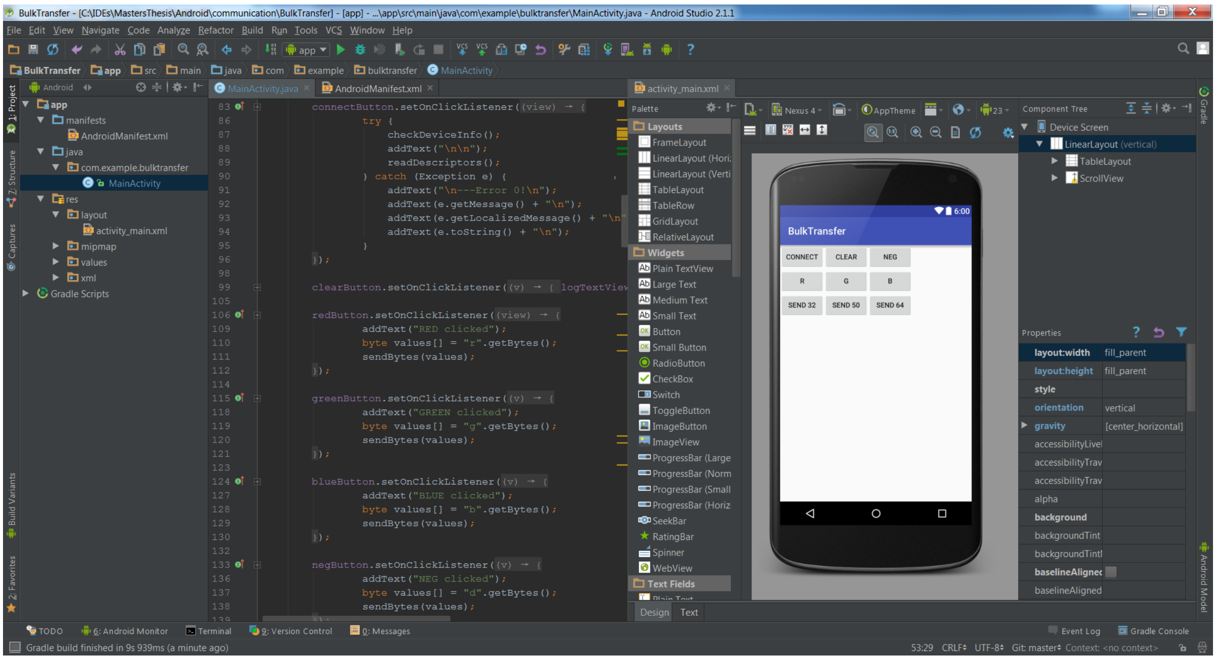

Figure 2.3 Android Studio

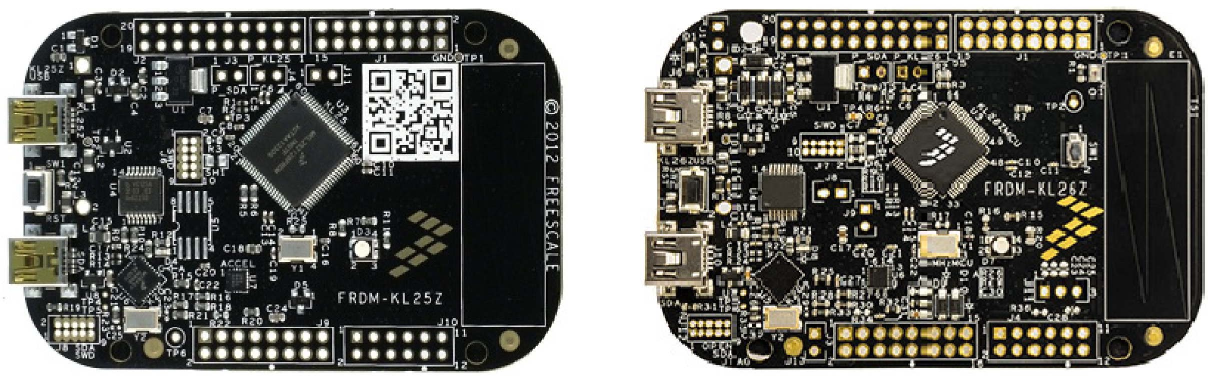

Figure 2.4 Freescale's FRDM KL25Z (left) and KL26Z (right)

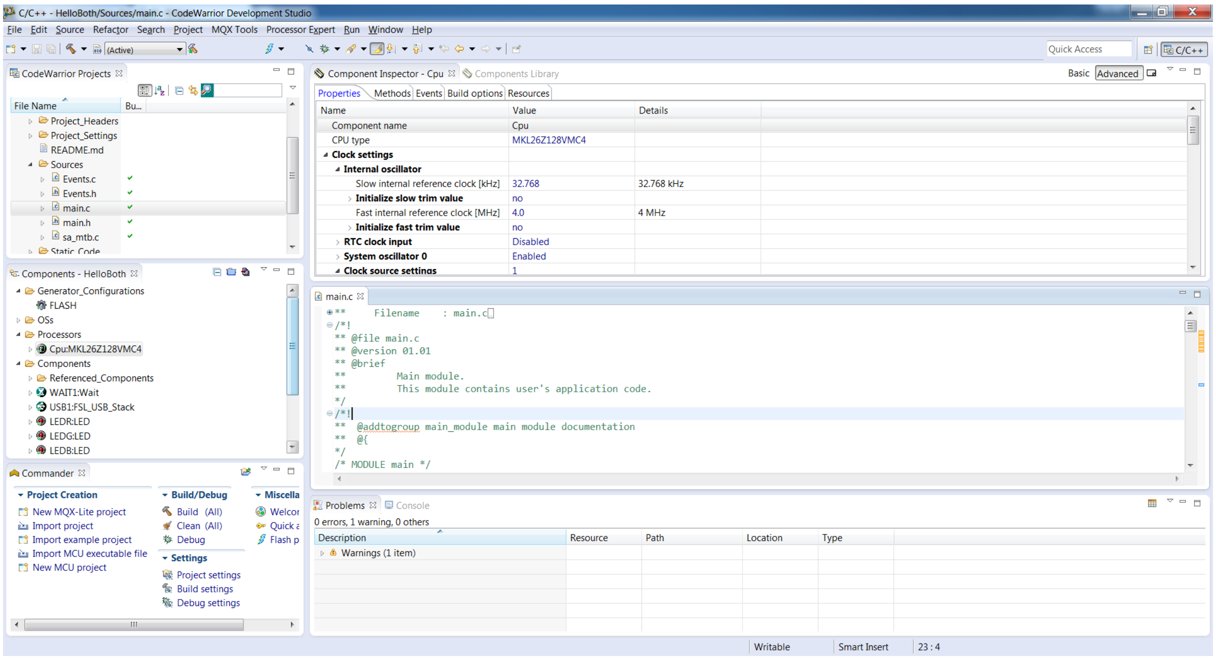

Figure 2.5 CodeWarrior

Chapter 3

Communication

3.1 Introduction

Three ways to communicate over USB were found:

- USB Host API [5],
- usb-serial-for-android library by mik3y [8],
- UsbSerial by felHR85 [9].

Because of similar names of projects, they will be referenced as Host API, mik3y and felHR85.

3.2 USB Host API

3.3 USB Serial by mik3y

3.4 USB Serial by felHR85

3.5 Summary

Chapter 4

Sensors

4.1 Introduction

Modern smartphones have many sensors, and most of them can extend robot's functionality. Sensors differ between phones, and new (or more advanced) ones can be connected using possible connections (mostly USB and Bluetooth). Most popular ones are:

- touch screen,
- accelerometer,
- gyroscope,
- microphone(s),
- front and rear camera(s),
- position sensors:
  - GPS,
  - multilateration based on GSM and/or WiFi,
- magnetometer,
- light sensor,
- proximity sensor.

Some (mostly high-end, or specialized ones) have also sensors like electronic compass, humidity/temperature sensors, fingerprint scanner, or even thermal camera.

Available implementations of face detection includes:

- FaceDetector API,
- Camera API,
- openCV for Android,
- openCV NDK.

4.2 FaceDetector API

```
void detectFaces(Bitmap image) {
        // detection
        FaceDetector face_detector = new FaceDetector(image.
            getWidth(), image.getHeight(), MAX_FACES);
        FaceDetector.Face[] faces = new FaceDetector.Face[
            MAX_FACES];
        int faceCount = face_detector.findFaces(image, faces);
        // drawing
        Canvas canvas = new Canvas(image);
        canvas.drawBitmap(image, 0, 0, null);
        PointF tmp_point = new PointF();
        Paint tmp_paint = new Paint();
        tmp_paint.setColor(COLOR);
        tmp_paint.setAlpha(ALPHA);
        for (int i = 0; i < facesCount; i++) {
                FaceDetector.Face face = faces[i];
                face.getMidPoint(tmp_point);
                canvas.drawCircle(tmp_point.x, tmp_point.y,
                    face.eyesDistance(), tmp_paint);
        }
}
```

4.3 Camera API

```
mCamera.startPreview();
mCamera.setFaceDetectionListener(new Camera.
    FaceDetectionListener() {
        @Override
        public void onFaceDetection(Camera.Face[] faces, Camera
            camera) {
                // no easy way to display it
                for(Camera.Face face : faces) {
                        Log.i(FACE, duration + " ms: " +face.
                            rect.flattenToString());
                }
        }
});
mCamera.startFaceDetection();
```

4.4 openCV for Android

```
@Override // from CameraBridgeViewBase.CvCameraViewListener
public Mat onCameraFrame(final Mat aInputFrame) {
```

```
        // convert to grayscale
        Mat grayscaleImage;
        Imgproc.cvtColor(aInputFrame, grayscaleImage, Imgproc.
           COLOR_RGBA2RGB);
        MatOfRect faces = new MatOfRect();
        // detect faces
        cascadeClassifier.detectMultiScale(grayscaleImage,
           faces);
        // mark faces on frame from camera
        for (Rect faceRect : faces.toArray()) {
                Imgproc.rectangle(aInputFrame, faceRect.tl(),
                   faceRect.br(), COLOR, THICKNESS);
        }
        return aInputFrame;
}
```

4.5 openCV NDK

4.6 Summary

Chapter 5

Summary

Bibliography

[1] Erich Styger, MCU on Eclipse, mcuoneclipse.com/, 29.04.2016

[2] Ganeev Singh, Mobile Controlled Robot, engineersgarage.com/contribution/mobile-controlled-robot, 29.05.2016

[3] Robotics Bible, Mobile Controlled Robot, roboticsbible.com/project-mobile-controlled-robot-without-microcontroller.html, 29.05.2016

[4] Mayoogh Girish, Mobile Controlled Robot, diyhacking.com/mobile-controlled-robot, 29.05.2016

[5] Google Inc., Android Reference, developer.android.com

[6] Itseez Inc., openCV Reference, http://opencv.org/

[7] Stack Overflow, http://stackoverflow.com/

[8] Mike Wakerly, usb-serial-for-android, github.com/mik3y/usb-serial-for-android, 20.05.2016

[9] Felipe Herranz, UsbSerial, github.com/felHR85/UsbSerial, 20.05.2016