Extending Lego Mindstorms NXT Functionality Using Internet-Connected Android Device
Abstract
Toys always have a meaning in people’s life. Nowadays toys become more usable, more functional with the technology. Lego Mindstorms NXT is one of the smartest toys in this industry. It can be controlled with other devices via Bluetooth. Smartphones are chosen as a device to control Lego Mindstorms NXT because they are used so common in the world. Android smartphones have a big slice in smartphones market so Android and NXT are used in this bachelor degree project.

This thesis shows a new approach where an Android smartphone provides more functionality by serving its own sensors and internet connection. The internet is used for connecting user and devices. New technologies are used for real time communication and a base application is developed with the communication infrastructure.

Keywords: Lego Mindstorms NXT, smartphones, Android, real-time communication.
Preface

Extending Lego Mindstorms NXT functionality was a degree project suggestion from my supervisor Oleg Danylenko in Linnaeus University. I would like to thank Oleg, for the support he gave me whole year and the opportunity to work with him.

I would like to thank Mathias Hedenborg for his great understanding during my Erasmus year.

I would like to thank Osman Nuri Homek for the great ideas and support he gave me.

I also would like to thank all my friends and family for the moral support they gave me. I would like to say “Tack” to my friends in Sweden: Alper Gundogdu, M. Akif Akben, Hossein Kazemi, Aboozar Kalantari, Pouya Khast, M. Mustafa Zora and Gokay Yildirim.

I would like to say “Hurmetler” to my friends in Turkey: Mustafa Aydin, Cihat Cengiz, Abdurrahim Sengul, Gokhan Gonultas, Kerim Koroglu, Yasin Altinsoy, Enes Agirman and Cihan Baylan.

Vaxjö, 2012-11-24

Ömer Baylan
# Contents

1. Introduction .......................................................................................................................... 1
   1.1. Overview .......................................................................................................................... 1
   1.2. Problem .......................................................................................................................... 2
   1.3. Motivation ....................................................................................................................... 2
   1.4. Goals ............................................................................................................................... 2
   1.5. Contributions ................................................................................................................ 3
   1.6. Report Structure ............................................................................................................ 3

2. Background .......................................................................................................................... 4
   2.1. Overview ........................................................................................................................ 4
   2.2. Technologies .................................................................................................................. 4
       2.2.1. Transmission Control Protocol (TCP) Client/Server ............................................. 4
       2.2.2. Android Cloud to Device Messaging Framework (C2DM) ...................................... 5
       2.2.3. Pusher API .............................................................................................................. 6
       2.2.4. PubNub API ............................................................................................................ 7
       2.2.5. Discussion ............................................................................................................... 8
       2.2.6. Conclusion ............................................................................................................. 8
   2.3. Required Knowledge ....................................................................................................... 8
       2.3.1. HTML (Hyper Text Markup Language) ................................................................. 8
       2.3.2. CSS (Cascading Style Sheets) .............................................................................. 9
       2.3.3. JavaScript ............................................................................................................ 9
       2.3.4. PHP (Hyper Text Preprocessor) .......................................................................... 9
       2.3.5. MySQL ................................................................................................................ 9
       2.3.6. AJAX (Asynchronous JavaScript and XML) ......................................................... 9
       2.3.7. XML (Extensible Markup Language) .................................................................... 9
       2.3.8. JSON (JavaScript Object Notation) ..................................................................... 10
       2.3.9. Imgur API ........................................................................................................... 10
       2.3.10. PubNub API ..................................................................................................... 10
       2.3.11. Android .............................................................................................................. 10

3. Solution ............................................................................................................................... 12
   3.1. Analysis .......................................................................................................................... 12
   3.2. Structure and Design ...................................................................................................... 13
   3.3. Implementation ............................................................................................................. 15
       3.3.1. Android Application ............................................................................................ 15
       3.3.2. Design and Implementation of the Web application ............................................. 19
**Table of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Lego Mindstorms NXT</td>
<td>1</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>TCP Server/Client</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Three-way Handshake</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>C2DM a Message’s Life</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Understanding Pusher</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>PubNub Simple Structure</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>PubNub Methods</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Real-time APIs</td>
<td>8</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>JSON Message</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Demonstration of Architecture</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Use Cases</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Class Diagram</td>
<td>13</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Photo Sequence Diagram</td>
<td>14</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>GPS Sequence Diagram</td>
<td>14</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Movement Sequence Diagram</td>
<td>15</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Paired Devices</td>
<td>16</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>Bluetooth Permission</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Main View</td>
<td>16</td>
</tr>
<tr>
<td>Figure 3.10</td>
<td>Internet Error</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.11</td>
<td>Location Error</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.12</td>
<td>Services View</td>
<td>18</td>
</tr>
<tr>
<td>Figure 3.13</td>
<td>Interactive part of website</td>
<td>19</td>
</tr>
<tr>
<td>Figure 3.14</td>
<td>Sample command</td>
<td>20</td>
</tr>
<tr>
<td>Figure 3.15</td>
<td>Demonstration of responses</td>
<td>20</td>
</tr>
<tr>
<td>Figure 3.16</td>
<td>Admin Login</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3.17</td>
<td>Responses and Commands Table</td>
<td>21</td>
</tr>
<tr>
<td>Figure 3.18</td>
<td>PubNub Developer Console</td>
<td>22</td>
</tr>
</tbody>
</table>
1. Introduction
In this chapter an overview about the Lego Mindstorms NXT and Android smartphones will be given firstly then readers will be introduced with the problem. Eventually, the motivation to solve the problem will be explained and goals will be decided in this chapter.

1.1. Overview
Lego Mindstorms NXT is a toy which has an intelligent brick that makes it a smart robot. The Lego Mindstorms is attractive for people of different ages. Lego Mindstorms NXT can be used for different purposes depending on the skills of its user. It can be programmed by the user easily or can communicate with different devices via Bluetooth. These devices can be computers or smartphones but nowadays smartphones become one of the fundamental needs of the people. Almost 50 percent of the phone users prefer using smartphones and Android takes the big slice of the cake in this market. According to report of the Canalys Company, 48 percent of the smartphones have Android operating system [25]. Especially Google, Samsung, HTC, Sony Ericsson and NVIDIA support Android OS and it makes Android very attractive for the developers. Afterwards, NXT can be an attractive robot for Android phone users.

Figure 1.1 Lego Mindstorms NXT
This thesis will present a prototype of controlling NXT with an Android application by using different technologies.

1.2. Problem
As previously stated, Lego Mindstorms NXT is a smart and programmable robot and it can be controlled by other devices via Bluetooth. The device for this degree project is an Android smartphone. The Bluetooth technology has one limitation with distance that restricts the movement area of the Lego Mindstorms NXT. The solution for this problem is to apply different technologies for communicating with a robot, like Internet, for example. However, Lego Mindstorms NXT does not have Internet connection so for the connection Android smartphone’s internet connection can be used. It also gives the advantage of computational power of Android device and using Android device’s sensors in addition to sensors of Lego Mindstorms NXT.

1.3. Motivation
Lego Mindstorms NXT can be used as a watchdog in an office, baby room or house as a simple security machine as well. The idea of controlling Lego Mindstorms NXT over the Internet can cause different usage of this simple robot. This project should attract people to use their Android smartphones with Lego Mindstorms NXT as a security kit and fun for their children.

1.4. Goals
The focus of this degree project is developing one Android application that provides users to control Lego Mindstorms NXT and the phone’s sensors over the Internet. The user should send commands over the Internet and the application should interpret and handle these commands. We can list goals as below:

- Develop an application that can be base to build different application with Lego Mindstorms NXT.
- Allow using camera and location sensors of the phone and serve motors of the Lego Mindstorms NXT.
- Provide an interface that is simple and easy to send commands for end users.
- Establish a real time communication between user and application.

According to these goals we decided to develop a prototype which is a kind of watchdog robot that can move forward/back, turn it to the right/left, take photo and get location coordinates then send the results to the user. An interface will be developed for users to communicate with this prototype. End of the project the functionalities will be tested by users. The communication between user and product will be evaluated for real timeliness according to time between sending commands and receiving responses.
1.5. Contributions
The product of this thesis, an NXT watchdog, will be the first application which is controlling Lego Mindstorms NXT over the Internet in the Google Play if we upload it there.

This project may lead developers to focus to new approaches according to the technologies which are used in this project like Real-time Push messaging service, photo uploading application programming interface (API) and using phone as a bridge between the Internet and Lego Mindstorms NXT.

1.6. Report Structure
The report is structured on four chapters as follows. Second chapter explains the background of the decisions of structure, architecture and technologies which will be used for real-time communication. Solution will be analyzed and implemented in third chapter. Additionally, the product of the degree project will be demonstrated with the help of UML diagrams and screenshots a validation of the solution and a comparison with the goals will be made in the last chapter.
2. Background
In this chapter a definition of technologies, required knowledge and related works is given. It should explain the architecture and the structure decisions which affect the thesis development process.

2.1. Overview
This degree project has two main components: one of them is Android application that behaves as a server and the other one is user interface on the Internet side that behaves as a client. The server side, Android application, has two main obligations: one is communicating with client and the other one is communicating with Lego Mindstorms NXT.

Users will send commands from client to the server. This is the main responsibility of the client. The server will receive these commands then it will interpret and handle them. According to the commands photo or location information can be taken by phone or Lego Mindstorms NXT can be moved forward, back, right or left. After the commands are performed, as last action server will send the results of the action to the client. The other responsibility of the client is receiving and displaying responses to user in proper form.

2.2. Technologies
Android application needs to get commands from user via client application over the Internet. There are two main approaches to get data from a server [1]. One approach for getting data is that the application has to poll a server for new data periodically. But polling technology uses unnecessary bandwidth and consumes the battery of mobile phone significantly even if there is no new data. It does not provide a real time communication, because of the time between every poll. The other alternative and the effective approach is Push technology. In this approach, server contacts the application if new data is available then application may fetch data from server. This approach provides the real time communication also. Google Contacts, Calendar, Gmail and more applications use this method. As mentioned before, real time communication is one of the criteria of the goal.

As a result, the possible technologies that include some real-time APIs, network protocols will be explained and discussed in this section and according to decision the architecture and structure will be designed.

2.2.1. Transmission Control Protocol (TCP) Client/Server
TCP is one of the essential protocols of the Internet [2]. TCP provides reliable, ordered delivery of the streams of bytes from an application/program on a device to another application on another device. TCP is a sliding window protocol that provides handling for timeouts and retransmissions for both server and client sides [3]. It also provides real time communication because in TCP there is a synchronized point to point connection between sockets (Figure 2.1). Server socket always listens to a port and the client has to connect this port for
communication and this connection is established by Three-way handshake manner that provides the synchronization between server and client before sending data to each other (Figure 2.2). The client application can be a desktop, Web or Android application which provides an interface for getting commands from users, but implementing a Web application needs hard working. The server application will be always an Android application which is a kind of bridge between client and Lego Mindstorms NXT.

2.2.2. Android Cloud to Device Messaging Framework (C2DM)
Android Cloud to Device Messaging is a service that provides sending data from servers to the applications on Android devices. C2DM framework is based on Extensible Messaging and Presence Protocol (XMPP) which is an open Extensible Markup Language (XML) technology for real time communication. There are three parties in C2DM: Android mobile device which runs an application that uses C2DM, third-party application server which sends to data to the application via C2DM servers, and C2DM servers which routes the data to the application (Figure 2.3).

Android has a connection to the Google Play and C2DM uses this existing connection and it provides optimization of bandwidth usage and battery consumption. The mobile device which runs the application requires Google Play and one Google account to use this API and also user’s mobile phones have to have Android 2.2 version or higher versions. The limitations of C2DM API are message size and number of the messages. The message size has to be 1024 bytes at most, and approximately 200.000 messages can be sent and received per day. C2DM offers desktop, Web or Android application options as client application but there are no official libraries for desktop and Web applications. The main disadvantage is that C2DM does not guarantee delivering messages. Handling timeouts and retransmissions should be implemented by developer.
2.2.3. Pusher API

Pusher is a service for web developers that handles the real-time updates to the applications via Web Sockets (Figure 2.4). It has Publish/Subscribe model that allows the users to filter and control how messages are received. Pusher offers to the developers some libraries for Java, PHP, Ruby, Perl, JavaScript, iOS and Android. It allows 100,000 messages per day and 20 connection for free usage. The message receiving time is nearly 5ms. In order to implement the degree project, different solutions can be used easily by this API. Pusher provides flexibility and different options with easy libraries for client like Android application or a Web application. Pusher also has the same disadvantage that C2DM - no guarantee of delivering messages. Developer has to find the way to handle timeouts, connection lost problems, and retransmissions.
2.2.4. PubNub API

PubNub is a cloud service based on the standard Hypertext Transfer Protocol (HTTP) 1.1 protocol and it delivers real-time communications for mobile, web, and server. PubNub is similar API to Pusher on Publish/Subscribe model. PubNub API provides large scaled client libraries for all major programming languages and mobile platforms. These are the some of the platforms and programming languages that have libraries provided by PubNub: IPhone, Android, Blackberry, Symbian, Windows Mobile, Firefox, IE, Opera, Safari, Chrome, PHP, Ruby, Python, Perl, .NET, C#, Node JS, Twisted and HTTP Environments. PubNub has a lot of advantages that are explained below:

- Does not require any knowledge or expertise about system management, monitoring and managing servers, network, and administration.
- The communication in real time from server to client, server to server, client to client is possible.
- Subscribers which receive messages can be publishers which send messages at the same time (Figure 2.5).
- Provides 250 connections and 1 million messages per month for free usage and each message size can be 1800 bytes at most.
- Message delivery speed is the fastest one compared to other APIs that “is measured in nanoseconds” [7].
- Prices of PubNub are cheaper than prices of Pusher.
- Very simple usage with two basic methods. In every platform and programming languages understanding just two methods is enough to use PubNub (Figure 2.6).
- PubNub guarantees message delivery. Even if a network drops, messages are delivered automatically on reconnect.

```
// send events
PUBNUB.publish ({
    channel : "hello_world",
    message : "data"
})

// listen to events
PUBNUB.subscribe ({
    channel : "hello_world",
    callback : alert
})
```

Figure 2.5 [8] PubNub Simple Structure Figure 2.6 [8] PubNub Methods
2.2.5. Discussion
In this section, technologies for real-time communication will be discussed according to three factors: programming effort, reliability on message delivery, and availability on different platforms e.g. desktop, web and Android.

TCP server/client structure is the most reliable service for communication. However this approach takes much more programming efforts comparing to the other options.

C2DM, Pusher and PubNub are easier then TCP to use for users and to implement for developers, but C2DM and Pusher does not guarantee message delivery. C2DM does not have official libraries for desktop and Web applications. Pusher and PubNub offer wide range of libraries and easy usage for developers. C2DM is XML based, Pusher and PubNub are JavaScript Object Notation (JSON) based and JSON is generally light weight data interchange format in comparison to XML.

Pusher works on Web Sockets protocol and Web Sockets protocol is blocked by proxy servers which are used by many networks (e.g. school, government, and corporation). PubNub does not have this kind of problem because its requests appear as normal HTTP requests. After all, the most important advantage of PubNub is that it guarantees message delivery.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Programming Effort</th>
<th>Reliability</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>C2DM</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pusher</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>PubNub</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Figure 2.7 Real-time APIs

2.2.6. Conclusion
Eventually we decided to use PubNub for real-time communication after all comparisons (Figure 2.7). It provides to developer very simple usage with basic two methods and large libraries in all platforms. Client application can be desktop, Android or Web application by flexibility of PubNub. Developing Web client is the best option in order to be accessible from everywhere, so a website will be developed as a client application.

2.3. Required Knowledge
The required knowledge for designing a website as client application and Android application as server application will be given in this section.

2.3.1. HTML (Hyper Text Markup Language)
“HTML is a language for describing web pages” by using markup tags. HTML pages are text files end with “.html” extension. HTML will be used when designing website’s page as the client’s interface. Hyper Text is method that provides users to move around on the web freely by clicking the special text on
the page which is called hyperlinks. Markup is the html tags effects for example bold text.

2.3.2. **CSS (Cascading Style Sheets)**
“CSS defines how to display HTML elements”[^10] and allows developers to avoid dealing with lots of tags. It helps configuring easily the fonts, colors, spacing as shortly layouts and styles of the HTML elements.

2.3.3. **JavaScript**
JavaScript is a scripting language that adds interactivity to the HTML pages. It can be executed directly in browsers. JavaScript allows developers to react to events, for example when a user clicks a button, manipulate the HTML elements e.g. changing the content of the element. JavaScript will be used for interaction with users and validating the input data.

2.3.4. **PHP (Hyper Text Preprocessor)**
“PHP is a server-side scripting language”[^11] that supports many databases (e.g. Oracle, MySQL, PostgreSQL, and Informix). PHP runs on different platforms like Windows, Linux and compatible with almost all servers e.g. Apache, IIS, and Tomcat etc. It is easy to learn and very efficient on the server side. PHP will be used for interaction with database where the requests and the responses are stored.

2.3.5. **MySQL**
“MySQL is database management system which is developed, distributed and supported by Oracle Corporation.”[^12] MySQL provides adding, accessing, and processing data in database and database will be used to keep commands and responses from Android application with user’s information. These MySQL actions will be accessed by PHP inside the Web application.

2.3.6. **AJAX (Asynchronous JavaScript and XML)**
Ajax is not a new language: “it is a technique for creating fast and dynamic web pages”[^13]. One page can include more than one forms and Ajax is used for updating some of these forms without reloading the whole page. Ajax uses a combination of the Internet standards:

- XMLHttpRequest object (for exchanging data with server)
- JavaScript (for displaying or interacting with the information)
- XML (the format for transferring data)

2.3.7. **XML (Extensible Markup Language)**
XML is a markup language and simple, flexible way of the encoding text and data. XML file can be processed purely as data by a program or it can be displayed like HTML file. It is also used the store data. It is unlike HTML
because tags are not predefined, they are self-defined. XML will be used for designing layouts of activities in Android application [14].

2.3.8. JSON (JavaScript Object Notation)
JSON is a lightweight, human readable, and language independent text format for data exchanging (Figure 2.7). It also provides convenience to parse and generate data.

```
{
    "nickname" : "arbaylan",
    "email" : "ob222av@student.lnu.se",
    "command" : "NXT please go forward"
}
```

Figure 2.8 JSON Message

2.3.9. Imgur API
Imgur’s API provides mechanism for uploading photos anonymously, uploading photos to your account and managing photos in your account [15]. As stated previously, one of the features of the degree project is taking photos by phone’s camera and sending to client to be presented to the user. Therefore on the client side, server’s database should keep them. But keeping photos as bytes is not optimal, so uploading photos to a website and keeping the links of the photos in database is the smarter way.

2.3.10. PubNub API
PubNub is described above in section 2.2.4. Shortly, PubNub will be used for real time communication between the website (client) and the android application (server). Website will send the commands by publish() method and receive the responses from the application by subscribe() method. The server application will be subscriber for receiving commands. After performing actions, it becomes publisher for sending responses.

2.3.11. Android
Android is the most popular mobile platform in the world. “It is a software stack that includes an operating system and supported by Google.”[16] There are three main layers: The kernel layer is based on Linux version 2.6 and all the open source libraries between kernel and applications are written in C and C++ [17]. The application layer is written in customized version of Java programming language. Android applications are not converted into class files like Java applications, instead to Dalvik Executable (.dex) format which is designed for systems with limited memory and processor speed. The .dex file is stored together with data and resource files in an Android package with an .apk suffix. The .apk file is similar to .jar file that contains all necessary files to install application on Android mobile device. Android applications have essential building blocks that called application components. There are four main
application components help to define the overall behavior of the application [18]:

- **Activities** represent a visual interface that helps to interact with user. The visual interface based on XML and the most commonly used component is Activity component in an application.
- **Services** run in background to perform long-running operations and do not provide a user interface.
- **Content Providers** let the application to share its data with other applications.
- **Broadcast receivers** respond the system broadcast announcements.

A component has to be declared in manifest file to be used. The manifest file is an XML file that contains all the information about application requirements which include entry point, permissions, component capabilities, additional libraries, and version of the Android [19].

Google provides the Android Development Tools (ADT) to develop Android applications with Eclipse IDE. ADT provides all required functionalities for creating, compiling, debugging Android applications from Eclipse IDE and an Android device emulator to test applications without a real device but emulator does not provides all capabilities which an Android mobile has.
3. Solution
Chapter 3 exposes the algorithm, architecture, and the implementation of the degree project. Source code, diagrams will be used to give more understanding.

3.1. Analysis
There are two sides of the degree project: client side and server side. A website will be designed as a Web application on the client side. It should provide users simple interface to send commands and to see responses in proper form. It should be responsible for database interactions and administration. Commands and responses should be kept in database and they should be kept in different two tables with nickname, email, time and date information to track the responses received or not and to see the communication is in real time or not by the time difference between commands and responses (Figure 3.17). Responses which will be presented to the users will be taken from responses table. User should login as admin to manage these entries (Figure 3.16).

Android application which is on the server side will be responsible for establishing communication between users and the sensors of Android device and Lego Mindstorms NXT. It should be able to receive commands from Web client. After interpretation of these commands, it should handle the actions to responsible sensors. Camera and GPS of the Android device, serve motors of the Lego Mindstorms NXT are the sensors that should be used. Android application should be able to send responses of the handled actions to the Web client.

The process should start when the user submits command(s) which will be sent by PubNub API and then stored in command’s table in the database. The second step should be on server side: Server application should receive the command and parse it. After that, actions should be performed e.g. take a photo, get location data, move Lego Mindstorms NXT forward/back or turn it to left/right, or combination of these actions. Afterwards responses should be sent to the Web client. Basically the process can be demonstrated as Figure 3.1.

![Figure 3.1 Demonstration of Architecture](image_url)
3.2. Structure and Design

In section 2.3 and section 3.1 general architecture and structure is already explained (Figure 3.1). This section describes the architecture and structure of the degree project with help of UML diagrams \[20\].

The Use-Case diagram represents the main functionalities of the applications (Figure 3.2).

Figure 3.2 Use Cases

Figure 3.3 Class Diagram
In Figure 3.3 Class diagram is shown for describing the structure of Android application by visualizing classes, methods and relationship between classes. In this class diagram only methods and relationships between classes are shown, fields are omitted for better demonstration.

The processes mentioned in section 2.1 and shown as use cases in Figure 3.2 will be demonstrated with sequence diagrams.

![Figure 3.4 Photo Sequence Diagram](image)

Figure 3.4 Photo Sequence Diagram

Figure 3.4 depicts the process of taking a photo. Diagram will be explained step by step.

1. Add command: User should add command by clicking Take Photo button in Web application.
2. Submit command: User should submit command by clicking submit button for sending command to Android application.
3. Server activity will receive the command and call SensorHandler activity to parse and handle command.
4. SensorHandler will call the CameraHandler activity and photo will be captured.
5. Photo will be uploaded to the Imgur and link of photo in Imgur server will be returned to Server activity.
6. Server activity will send response, the link of the photo, to the Web client by PubNub API.

![Figure 3.5 GPS Sequence Diagram](image)

Figure 3.5 GPS Sequence Diagram
Figure 3.5 presents the process of getting location information. First three steps are performed in the same order for all command types. Location information will be taken from GPS sensor or network inside of the SensorHandler activity. Then latitude and longitude data will be returned to the Server activity and sent to the Web Client.

![Movement Sequence Diagram](image)

**Figure 3.6 Movement Sequence Diagram**

Figure 3.6 demonstrates the process of movements of the serve motors of the NXT. SensorHandler activity will send data over Bluetooth to move Lego Mindstorms NXT. After that it will send movement information to the Web client. User can also submit a combination of these commands without any restriction of number of commands.

### 3.3. Implementation

The implementation of the project and website design will be explained in this section. The first step of the implementation should be server application. As we mentioned before, server application has two parts: One part is Android-NXT part; the other part is Android-Internet part. Android-NXT part will be the first implemented part.

#### 3.3.1. Android Application

Android application should connect to Lego Mindstorms NXT as first step. After establishing a connection, it can communicate with Lego Mindstorms NXT. Server application will be explained step by step.

- **ChooseNXT**

This class allows the user to scan for other Bluetooth devices and querying the phone’s Bluetooth adapter for paired devices. **ChooseNXT** class is derived from Android sample application [21]. This Activity should appear as a dialog window in the main activity and list all paired devices (Figure 3.7). It should also be able to discover the detected devices in the area. When the user chooses a device MAC address of the device should be sent to the parent Activity as a result of the Intent. The devices should be listed in the dialog window and they are specified to be a robot toy.
• **MainActivity**

`MainActivity` class is the main activity of the project. Connection between phone and Lego Mindstorms NXT is already implemented with `ChooseNXT` and as mentioned before `ChooseNXT` activity should appeared as dialog window. Before the connection to the Lego Mindstorms NXT, Bluetooth of the phone should be checked if it is enabled or not. The control mechanism is implemented inside this activity. User can enable Bluetooth by manipulation of the application (Figure 3.8).

![Figure 3.7 Paired Devices](image)

![Figure 3.8 Bluetooth Permission](image)

![Figure 3.9 Main View](image)
If user refuses to turn on Bluetooth, application will automatically terminate itself. After activating Bluetooth, application will offer user to choose the NXT device by calling ChooseNXT activity (Figure 3.7). Afterwards phone will connect the chosen NXT (Figure 3.9) and user becomes ready to control NXT. User also has option to activate server with calling the Server by clicking “Control NXT by Internet” and it will explained at Server.

- **NXTCommunicator**
  Connection between Lego Minidstorms NXT and Android device is established by calling **NXTCommunicator** activity after user chooses the device from dialog window inside main activity. Firstly the device MAC address is received from main activity, then Radio Frequency Communication (RFCOMM) channel is established and connection becomes ready [22]. Afterwards, commands can be sent to Lego Mindstorms NXT.

  **NXTCommunicator** establishes a connection and provides functionality to move Lego Minidstorms NXT. These functionalities are derived from Jacek Federovsky’s “NXT Remote Control” project [23].

- **Server**
  **Server** class is the fundamental activity of the application. The communication between client and server application is implemented in this activity. PubNub channels are established here to receive commands and send responses to client with help of threads and handlers [14]. There are some requirements to make active this activity and all functionalities:

  - Bluetooth should be enabled.
  - Phone should have Internet connection [24] (Figure 3.10).
  - GPS or Network should be enabled for accessing location information [14] [24] (Figure 3.11, 3.12).

  All these requirements should be checked at main activity before this activity called. If requirements are met, **Server** activity will be launched. Additionally, when a new activity is launched Bluetooth connection is cut automatically by system, so Bluetooth should be checked again at start of this activity.
Server activity establishes the connection between Android application and Web application. This activity parses commands received from client and executes corresponding functions: taking photo, getting location data etc. After execution, application publishes the response. Actual respond time changes depend on the execution of functions. For example, after photo is taken will be uploaded to Imgur’s server and then the link received will be sent to the client. Duration of this process will be different each time depends on the internet speed. This activity also sends “connection lost” message when the connection is lost between Android device and Lego Mindstorms NXT. Server activity uses SensorHandler at background to execute functions.

- **SensorHandler**
  
  SensorHandler activity is parsing commands received from client in a Json object and handling these commands. This activity is called from Server activity, when a command received from client. After that it parses command and perform the action with private methods e.g. takePhoto(), getLocation(), moveForward(), moveBack(), turnRight() and turnLeft(), then it returns response to its parent activity. Response is sent over PubNub channel with publish() method inside Server. It also uses CameraHandler class for taking photo.

- **CameraHandler**

  The application has to be able to take photo from the phone’s camera according the goals of the thesis. There are two ways of taking photo with phone in Android: one of them is using the camera application of the phone and the other one is creating a customized camera [24]. This class provides a customized camera which allows the application to capture photo without interaction with user because this should be done remotely from client application by user commands. Additionally adjustment of preview properties before capturing photo and management of surface view are implemented in this class.

- **AndroidManifest**
All the information about application requirements is stored in manifest file. Permissions and hardware requirements will be explained in this section.

Bluetooth hardware is main requirement of the phone and it is controlled by declaring codes below at manifest file of the android application:

```xml
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus"/>
```

And these permissions below should be declared in manifest file for access and control the Bluetooth devices inside activities:

```xml
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
```

Additionally, activities need permission for accessing Internet, camera, network state and they are declared in this manifest file.

```xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

### 3.3.2. Design and Implementation of the Web application

In this section the client application will be described which is a Web application. Html is used to design the webpage. The webpage contains two forms: one for building commands and another for displaying responses. First form contains some buttons and input textbox where user is able to add commands e.g. Take Photo, Get GPS data, Move Forward, Move Back, Turn Right, Turn Left, and Submit (Figure 3.13). User can add commands easily over this form just by clicking on buttons and preparing a command set. After adding commands and clicking submit button, client application will send the commands and user information (nickname and email) in a Json object. Additionally nickname, email and commands will be saved to the database for management (Figure 3.14).

![Interactive part of website](image)

Figure 3.13 Interactive part of website
The input data should be checked to be in proper form with JavaScript functions. It is checked after submitting form. Sending input data in Json object to Android server is implemented with PubNub and database interaction is implemented with PHP and MySQL.

The declaration of PubNub API keys should be inside <body> tag to use publish() and subscribe() methods. For both of the methods, declaring one channel (which specifies the connection name) and the message is enough. For receiving responses from the server, subscribe() method is declared in <body> section of the HTML file. Ajax is used for saving responses to the database because one of the forms inside the webpage should be reloaded. (Figure 3.15)

```php
<?php
...
mysql_connect('localhost','arbancom_ar','destur');
mysql_select_db ('arbancom_guest');
mysql_query("INSERT INTO `NXTR` (`nickname`,`email`,`respond`,`date`,`time`) VALUES ('$nickname','$email','$respond', '$date', '$time')");
?>
```
There are two channels for communication in both client and server applications. As mentioned before in section 2.2.4, in PubNub a subscriber can be a publisher at the same time. There are two channels to provide it. Web client is subscriber with “android” channel and publisher with “android_sub” channel. Android server is subscriber with “android_sub” channel and publisher with “android” channel at the same time.

The website also has a control panel to manage the database. User should have administrator privileges to be able to access it (Figure 3.16). After accessing the control panel, user is able to monitor commands and responses entries and remove them (Figure 3.17).

![Admin Login](image)

**Figure 3.16 Admin Login**

<table>
<thead>
<tr>
<th>Nick</th>
<th>Email</th>
<th>Report</th>
<th>Date</th>
<th>Time</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>hey</td>
<td><a href="mailto:hey@geonni.com">hey@geonni.com</a></td>
<td>Lat: 41.025877, Long: 28.5400861</td>
<td>2012/08/26</td>
<td>13:23</td>
<td>Delete</td>
</tr>
<tr>
<td>adminNXT <a href="mailto:mt@geonni.com">mt@geonni.com</a></td>
<td>Connection Lost</td>
<td>2012/08/26 13:24 Delete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adminNXT <a href="mailto:mt@geonni.com">mt@geonni.com</a></td>
<td>Connection Lost</td>
<td>2012/08/26 14:30 Delete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>android <a href="mailto:and@gmail.com">and@gmail.com</a></td>
<td>NXT moved forward for 1 second</td>
<td>2012/09/05 13:28 Delete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>android <a href="mailto:and@gmail.com">and@gmail.com</a></td>
<td>NXT moved forward for 2 second</td>
<td>2012/09/05 13:29 Delete</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>android <a href="mailto:and@gmail.com">and@gmail.com</a></td>
<td>NXT moved forward for 3 second</td>
<td>2012/09/05 13:31 Delete</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nick</th>
<th>Email</th>
<th>Command</th>
<th>Date</th>
<th>Time</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>ali</td>
<td><a href="mailto:and@geonni.com">and@geonni.com</a></td>
<td>Take a picture! Move forward with power100 for 1 seconds</td>
<td>2012/09/09 18:14</td>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td>and</td>
<td><a href="mailto:and@geonni.com">and@geonni.com</a></td>
<td>Move forward with power100 for 1 seconds</td>
<td>2012/09/09 18:16</td>
<td>Delete</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.17 Responses and Commands Table**

PubNub is also has a developer console that provides simple subscribe and publish interface. Developers can use it for testing publish and subscribe methods manually and monitor the messages that are sent from given channel. (Figure 3.18)
3.3.3. Result

Android application and Web application are implemented. Firstly Android application is implemented and explained step by step together with screenshots. Android emulator is never used because it does not support Bluetooth or GPS properties of a real device. So application is developed on Android smartphone. Bluetooth connection and camera problems was the most faced problems. Eclipse debug perspective is used to detect problems. Google Search was the best source to find solution of problems. Afterwards, Web application is implemented. Microsoft Expression Studio is used as development environment for design of the webpage. Internet explorer is used as test environment for web pages.

Finally the product is tested on different real devices e.g. a Google Nexus S with Android 4.1 and 4.0, HTC Desire S with Android 2.3, and Sony Ericsson Xperia with Android 2.3 and it was working properly.
4. Conclusion and Future Work
In this chapter the conclusion of the product and work is given to the readers. Final product is compared to the goals. This chapter ends with suggestions for future work.

4.1. Conclusion
An application was developed using camera and GPS sensor of the phone and movement motors of the robot together. The application is working properly and in correspondence with the goals that are mentioned in section 1.4. This thesis shows that NXT robot use cases can be extended using Android device with Internet connection.

The client application provides simple interface for users to send commands via Web application. Responses received from Android server are displayed on a website. The website can be accessed from everywhere using following address: http://nxt.geonni.com/. A real time communication is performed between Web-client and Android-server by means of the PubNub API, threads and handlers. Data transmission takes less than 0.25 seconds per message.

Developers can build different applications using this infrastructure to extend NXT robot functionality in different ways.

4.2. Future Work
The product can be extended by using many of the features and sensors of the Android mobile phone and the Lego Mindstorms NXT. For example ultrasonic sensor, color sensor, touch sensor etc. of Lego Mindstorms NXT and accelerometer, compass, barometer, microphone etc. of smartphones. Developers can find new solutions and use cases. Developers can use communication infrastructure of the project. New projects can be developed with the approach of using Android device as a communication bridge, additional computational power and sensor data source for NXT robot. For example a simple robot which detects the fire and calls fire station to give coordinates, and then calculates how big it is, takes photos and sends again to station. Another example can be that a robot in baby room which detects the movement of baby and triggers the phone to listen to the sound then due to the level of sound calls the parents.
References


