Monitoring the status of the garbage containers

Project Report

Authors: Xavier Cerqueda Puig

Bernat Garcia Torrentsgeneros

Pierre Biojoux

Quentin Studeny

Junyoung Bang

Joonas Luukkanen

Supervisor: Torben Gregersen

Date: 16/12/2016

Table of Contents

1. Abstract

This report is the result of the international program AAD – Applied App Development for exchange students enrolled to the course ITIPRJ – Multidisciplinary Project during fall 2016. The goal of the project was to develop a system capable of helping trash collectors to collect garbage bins thanks to a smartphone app.

The system is confined to only one garbage bin and controlled by a microcontroller. The location and access to the internet is provided through a 3G shield. To get the range of the bin we use a sensor. Thus, we can send data to the smartphone through a personal server.

The project succeeded in implementing a functional Android app, capable of listing bins, seeing bins on a map, consulting details, updating the status and displaying the route to go to the bin selected.

In conclusion, a functional prototype has been developed based on the requirement specification.

2. Introduction

This project was developed during the semester of fall 2016, for the ITIPRJ course at Aarhus University. This project consists of an Internet of Things in the garbage bin, which transmits their full or empty status through 3G. The purpose of this project is to optimize garbage collection through this data transmission.

To accomplish this project, we set up following goals:

For the software part

Do a GPS tracking of garbage containers

Do TCP connections to store information about each garbage bin (on a private server)

Create an Android application to visualize the data

Get information about state of garbage containers on the app (JSON)

For the electronic part

Use an Arduino microcontroller to transmit data through internet– the garbage bin will essentially be an IoT

3G/GPRS Shield for Arduino

Monitor the current state of the bin through a sensor

You can find more information on the **Project description.pdf** document **[Reference 4]**.

Figure 1 below describes our plans for the system.

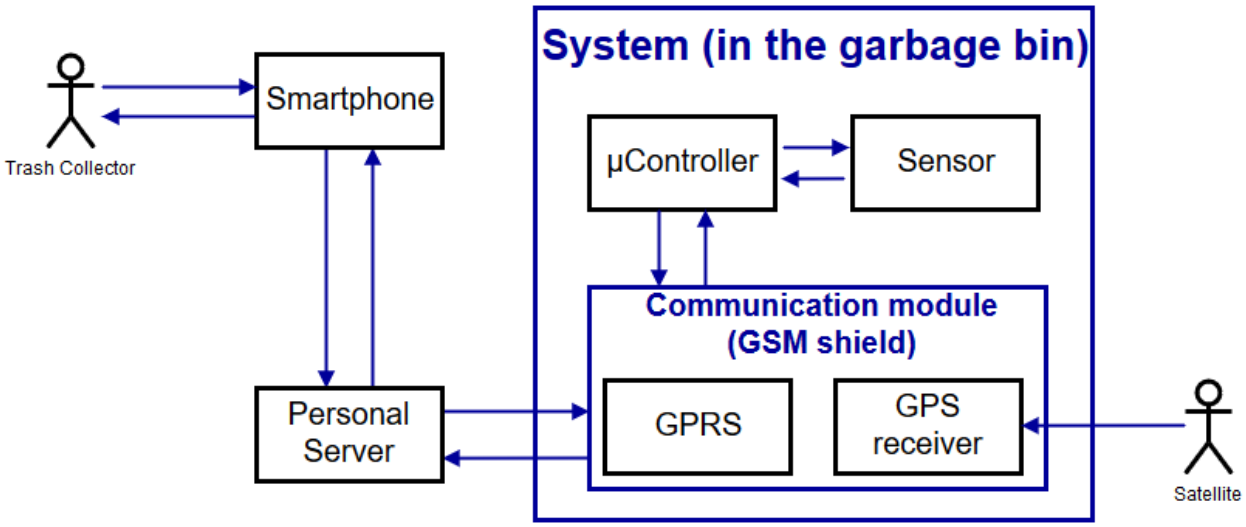

Figure 1: System overview

We divided into documentation part, application part and main controller part to make this project efficiently.

Documentation part

This is all documents that we had to write for the project and show to our supervisor: Torben Gregersen. This part was important to understand what we needed and where we were going. As we got more and more into it, the documentation had to be more detailed.

Application part

This is the part of development application using Android Studio. There are many functions in the app such as login, list of bins, details of a chosen bin, garbage bin position on the map and the route from current position to garbage.

Main controller

An ultrasonic sensor measures the distance between the sensor and the garbage. 3G/GPRS shield gets location and sends the information to the personal server (Xively) and it's stored there. Using 3G Communication, the smartphone can get information from the server.

Here, you can find our division of labour:

Tasks	Quentin Studeny	Pierre Biojoux	Xavier Cerqueda Puig	Bernat Garcia Torrentsgeneros	Joonas Luukkanen	Junyoung Bang
Sensor			X	X		
Android Application		X			X	X
Communication 3G/GPRS		X	X		X	
Use Cases		X	X			
IBD		X	X			
BDD				X		X
Sequence Diagram		X	X	X		
Project description	X	X	X	X	X	X
Pre-Analysis	X	X	X	X	X	X
Requirement specification	X	X	X	X	X	X
Architecture & Design	X	X	X	X	X	X
Tests	X	X		X	X	
Project Report		X	X	X	X	X

Table 1: Division of labour

3. Requirement

In this section, we describe the system and defined use cases. Furthermore, we describe our non-functional requirement.

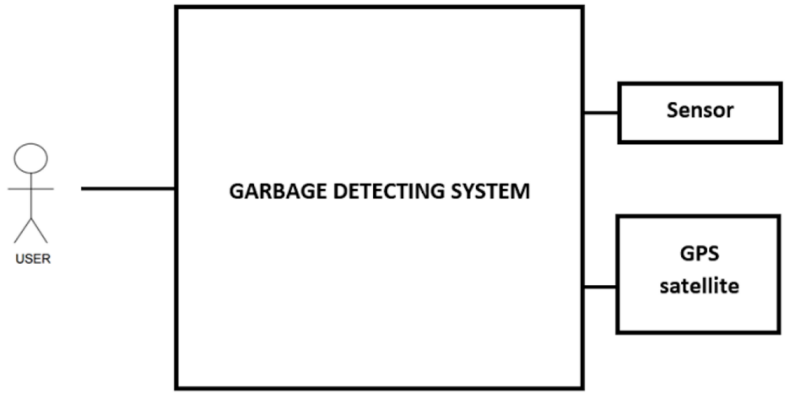

Figure 2: Requirement view

User: Main users are trash collectors. They will be able to check the status and location of the garbage containers. They are also able to prepare their collection route depending on which containers need to be emptied.

Sensor: Electronic device which collects data about the status of the garbage bins and send it to the microcontroller to be analysed.

GPS Satellite: Device responsible to send the location signal to the system (GPS receiver) in order to be analysed.

3.1 Use cases diagram

This is use case diagram **[Reference 1]**. Through this diagram, you can figure out how it works.

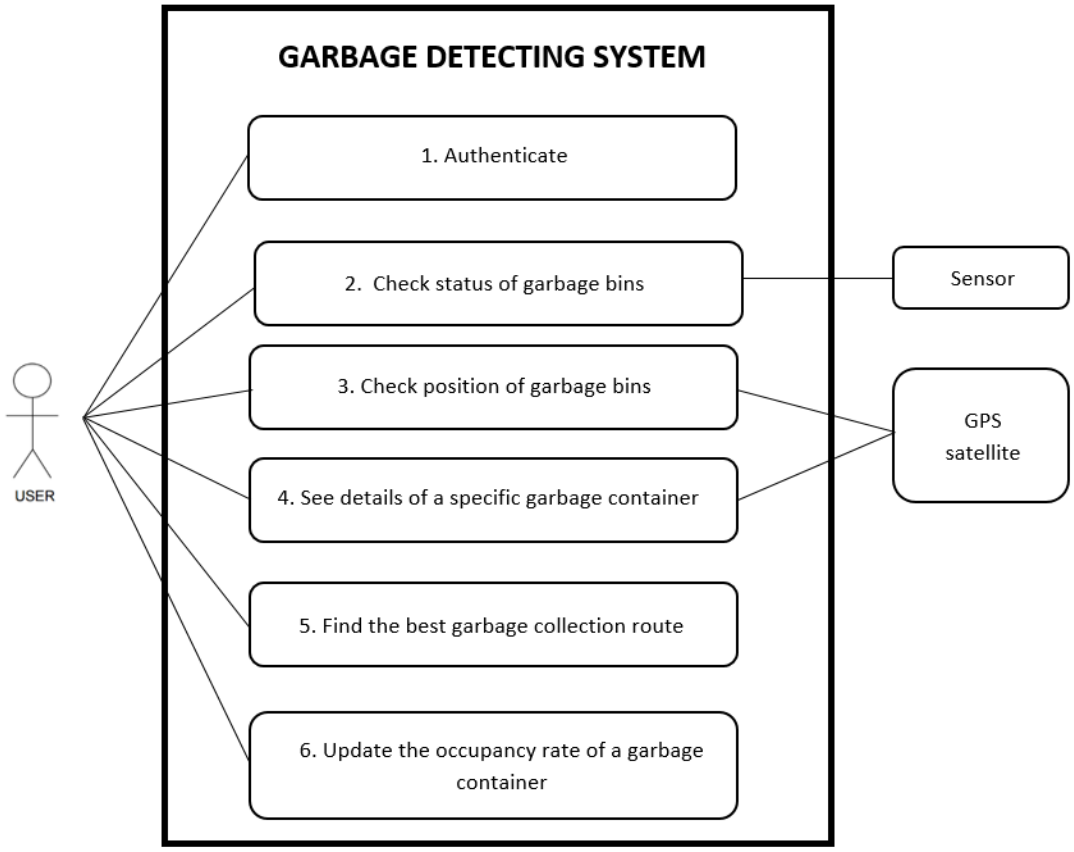

Figure 3: Garbage detecting system Use Case

Use case 1: The goal is to access to the application through log in. If the login or the password is incorrect, then the app shows the message “Invalid password or login”.

Use case 2: This is for checking status of garbage bins. It lets us to know how full are garbage containers. If the app can’t update data, the app shows the message “We can’t update data, check your internet connection.”

Use case 3: The goal is to see on the map the location of trash containers. If App can’t find your position, the app shows the message “We can’t find your position, check your internet connection.”

Use case 4: This is about details of a specific garbage container. You can see a description of a specific garbage container with many details.

Use case 5: This is about finding route. User can find the best garbage collection route on the map.

Use case 6: This is about updating the occupancy rate of a garbage container. After container have been emptied, the details on garbage container will be updated.

3.2 Non-functional requirement

On one hand, our system will be placed in a bin, and thus should be able to endure outdoor use and external elements like humidity, collisions, and average but not extreme temperatures (-40 to 85°C).

The ultrasonic sensor optimal functional temperature is from 20 to 50℃because of sound velocity, but we don't need to worry about sound speed variations since the bin isn't very large.

On the other hand, our project requires a sensor to know how much distance there is between the garbage bin and the top of the garbage container.

We have to know if the bin is full or not with our ultrasonic sensor. So, these sensors must have at least a range distance from 20cm to 2m. Because of the bin's size, the sensors maximum should be about 2m.

For more details see the **Requirement Specification.pdf** document **[Reference 6]**.

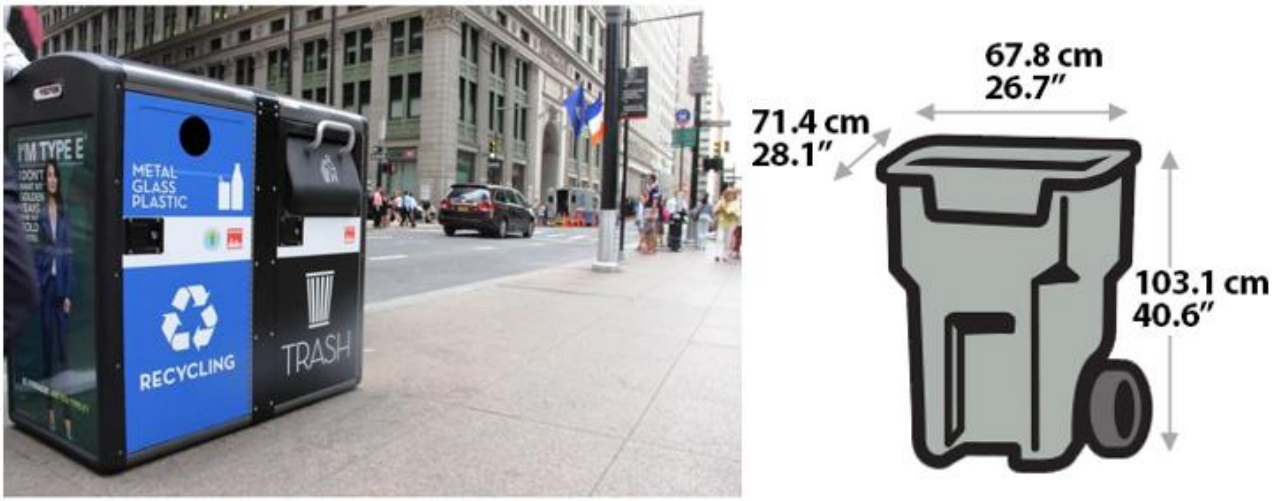

Figure 4: examples of garbage bin

Our project also contains external interface requirements where we explain our smartphone app's user interface. This application will be responsible to detect when there are empty garbage bins and show them on map. You can see that on **Requirements specification.pdf** document **[Reference 6]**.

4. Limits

We made only one prototype. it means that we just have one garbage container. Thus, the server Xively stores temporarily only one garbage container's data. Accordingly, we tested our system with this limitation in our minds.

The GPS signal connection doesn't always work as we want because of issues for GPS reception problem.

The distance data measured by the ultrasonic sensor may change depending on trash position. Because the data of distance depends on the angle of the ultra-wave reflection. To improve accuracy, we think we should put at least two ultrasonic sensors in the garbage container. But as it's a prototype, we only have one.

We tested our ultrasonic sensor between 25cm and 160cm and it worked fine. Nevertheless, in the datasheet specifications it is said that it can work until 765cm. If we consider the size of garbage container, it satisfies our conditions.

5. Realisation

5.1 Method

For this project, we used a lot of different analytical methods to implement the prototype and keep going on with the documentation. You can find the description and proposal of each method below.

5.1.1 SysML

SysML **[Reference 2]** is a general-purpose graphical modelling language that support the analysis, specification, design, verification, and validation of complex systems. First, we described domain model, which is based on the **Requirement specification.pdf** document **[Reference 6]**. After that, we defined external environment using a Block Definition Diagram(BDD). Finally, we described more detail through Internal Block Diagram(IBD), which is about internal signals between the garbage container system and others devices.

5.1.2 N+1 View

N+1 view **[Reference 3]** is a tool to implement software architecture part. It consists of logical view, process view, data view, security view, development view and physical view. You can see a detailed description of the different views in the **Architecture & Design.pdf [Reference 7]** document.

5.1.3 Development Tools

Development tool	Explanation
Arduino IDE	We used it to test sensor and network protocol.
Google drive	We stored all documents about this project on the google drive supporting versioning.
Microsoft Office	We used Microsoft Office Word to write documents.
Android Studio	We used it for android application development.
Java language	We choose java language to develop the application because of android studio.
Draw.io	We used this online tool to make diagrams for this report.
Github	We received help from open source distributed on the GitHub

Table 2: Development tools

5.2 Analyse

We describe the pre-analysis carried out in order to implement the project. This is about processes to compare different devices available in the market and let you know why we choose each one to make this project. You can check details on the **Pre-analysis.pdf** document **[Reference 5]**.

5.2.1 Microcontroller

We have chosen the Arduino Mega2560 because it would be more readily available since it's a popular platform. Moreover, the University can provide us some and the cost will not be an issue for our team. And it also meets our project requirements.

5.2.2 Ultrasonic Sensor

The purpose of the sensor in our project is to know whether the garbage container needs to be emptied or not. We compared with 4 different ultrasonic sensors. All the sensors fill requirements for our application. University's embedded stock has model MB1202 12CXL_MaxSonar-EZ0 available for rental. So, we decided to use it as it's adequate for the prototype.

5.2.3 Communication Module

In our project, communication plays a critical role given that our main goal is to accomplish communication between garbage container and user's phone. We discussed about the different options we had, such as, Wi-Fi, Bluetooth, LoRA, GPRS and 3G. Considering user's working environment, 3G is the best choice to communicate with phone devices.

5.2.3 Server

To store location and the range of bins that our µcontroller measures, we decided to use a personal server. Thus, the smartphone could get data easily. Between different servers, we chose Xively personal given that it's easy to use. To store data, it uses the name of the channel ("Location" and "Distance") and a API key to identify where to store values, and to parse data, we use JSON.

5.3 Architecture

This part contains all the information about the internal and external connections that conforms the system.

5.3.1 Domain model

Before we describe Architecture, we consider a Domain model. This view will make you understand more easily about our project.

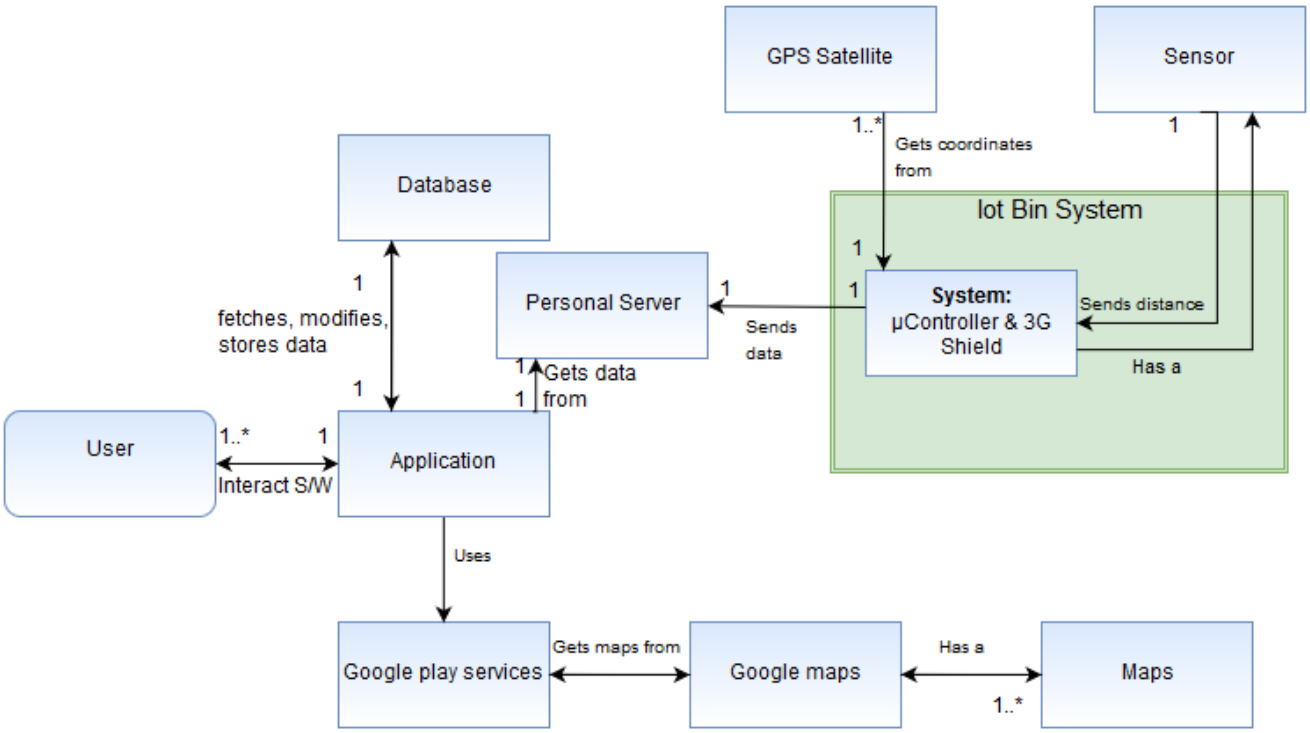

Figure 5: Domain model

5.3.2 Hardware

This part contains all information about the internal and external connections that conforms the system. We will use two different types of SysML diagrams to explain the functionality: BDD (Block Diagram Description) and IBD (Internal Block Diagram). For more explanations about the connections and hardware design you can see the document **Architecture & Design.pdf [Reference 7].**

5.3.2.1 Monitoring Garbage container BDD

This figure shows the main system blocks and their connections. Monitoring the status of the garbage container block consists of Smartphone and Garbage container block. And you can know each block's ports, input and output through BDD. It is to get a better understanding of all connections between the different two big blocks.

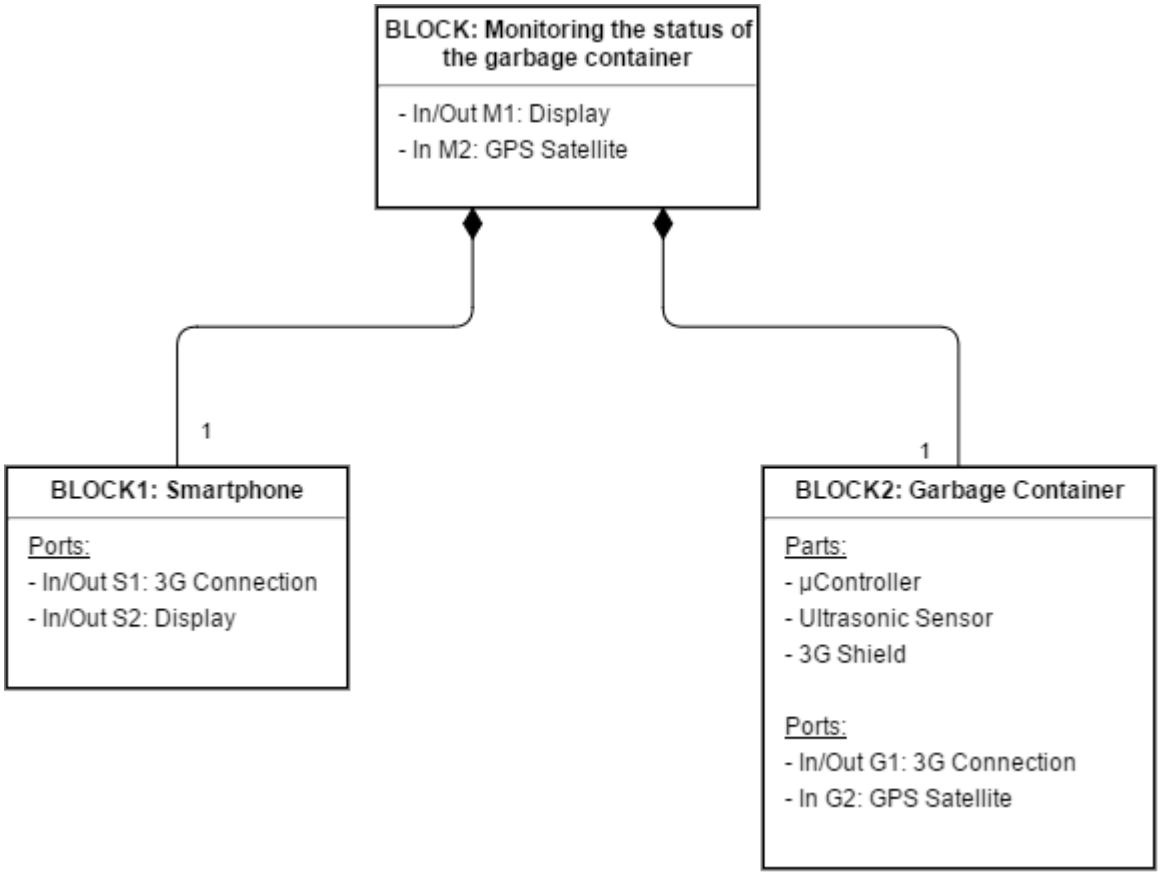

Figure 6: Monitoring Garbage container BDD

5.3.2.2 Monitoring Garbage container IBD

This is about more details on each internal blocks. So we describe Display, 3G connection and GPS satellite. Through IBD, you can figure out how to connect each block.

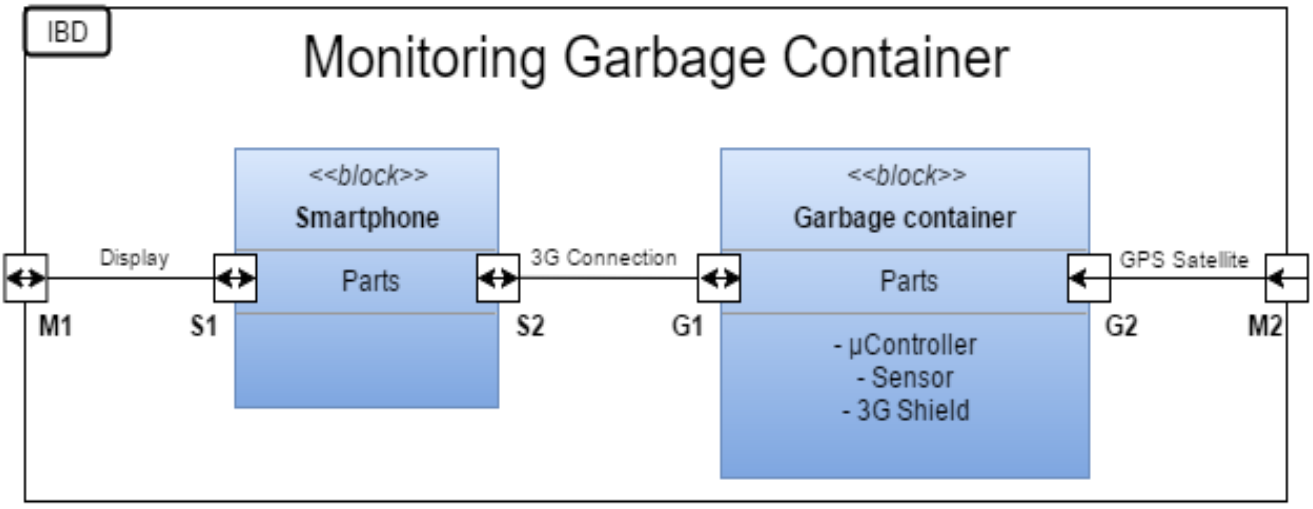

Figure 7: Monitoring Garbage container IBD

5.3.2.3 Garbage Container BDD

This is the Garbage container BDD. Garbage Container block consists of 3G shield, micro controller and ultrasonic sensor block. And you can know each block's ports, input and output through BDD.

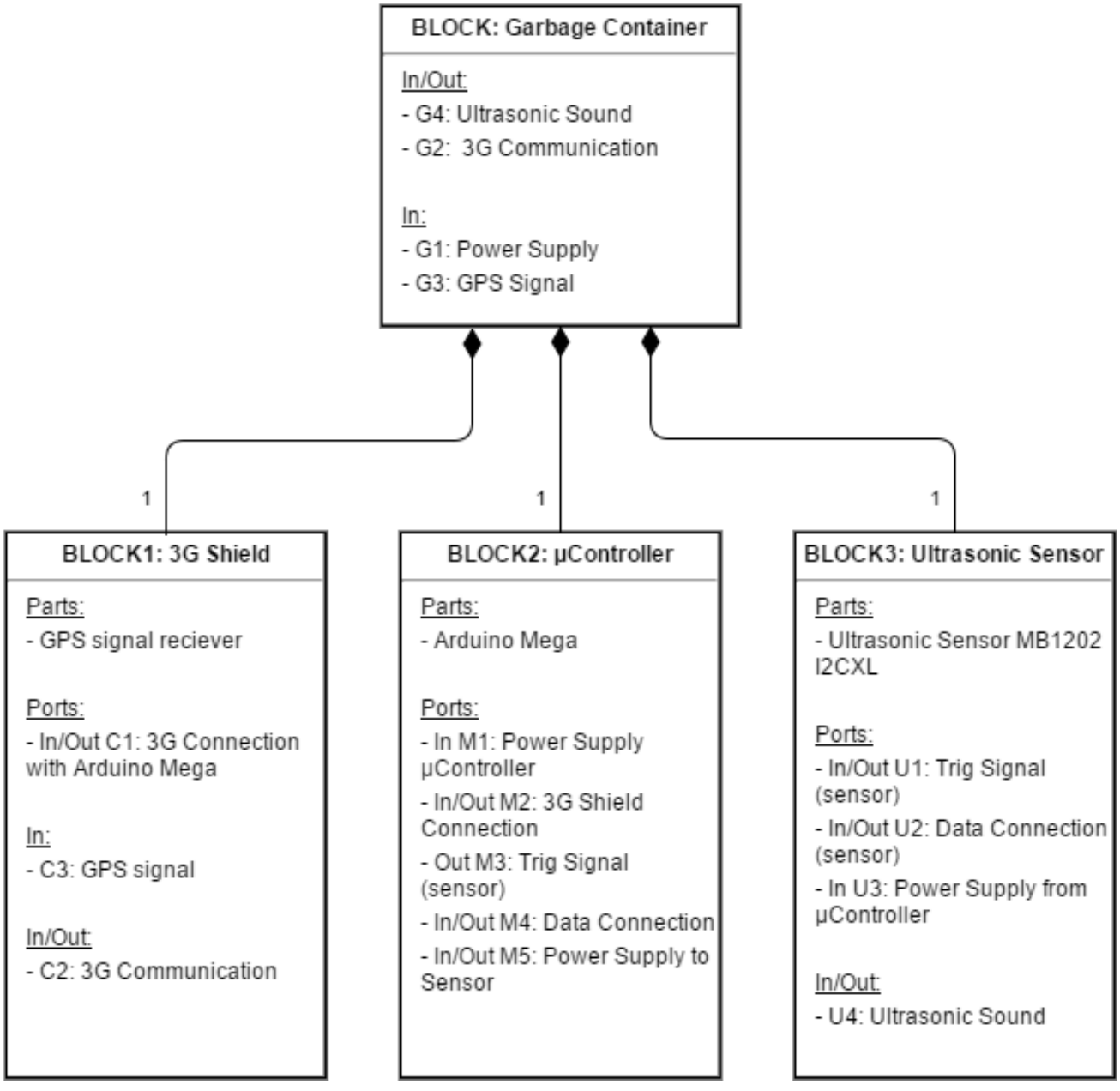

Figure 8: Garbage container BDD

5.3.2.4 Garbage Container IBD

This is about more details on each internal blocks. So we describe 3G communication, GPS signal, ultrasonic sound, physical assembly, power supply sensor, trig signal sensor, data communications sensor.

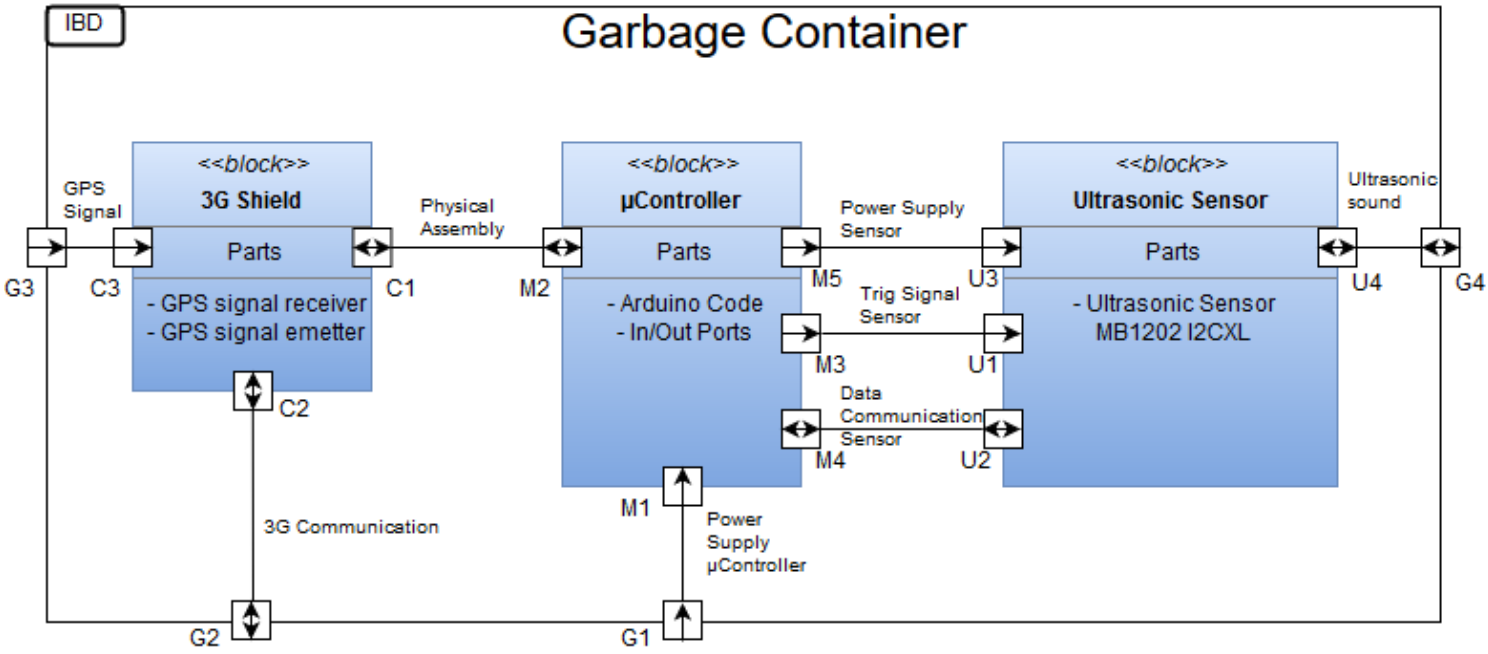

Figure 9: Garbage container IBD

5.3.3 Software

We opted for the agile-style development, since it fits our requirements. We will use the N+1 model view as a structural basis for this following part of the document. Architecture is described with more details in the document **Architecture & design.pdf [Reference 7]**.

Overall software view of the system is described in the following figure.

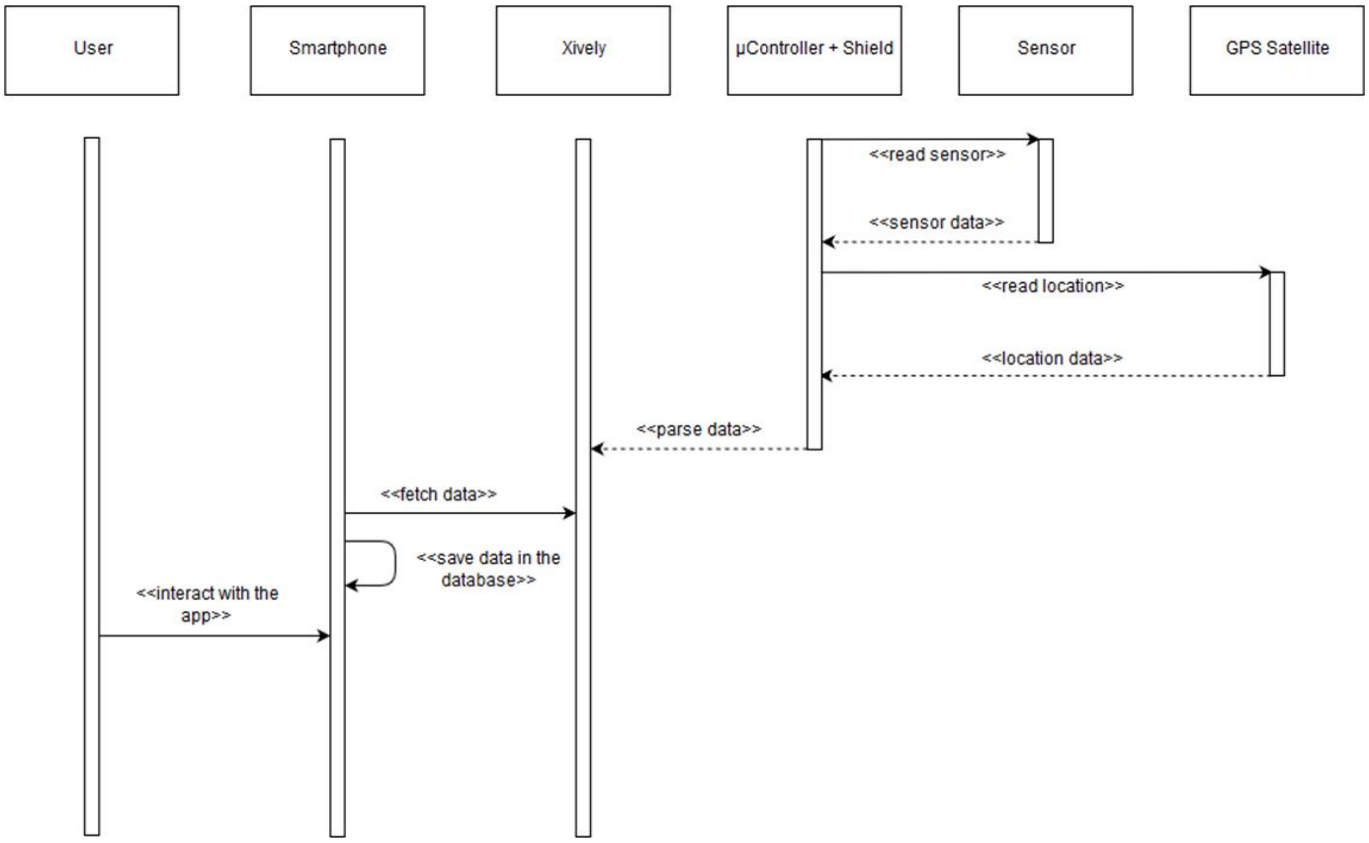

Figure 10: System Sequence Diagram

This sequence diagram describes general dataflow of the system and does not describe them in more details. In general, we can say data needs to flow from system inside garbage bin towards smartphone app which displays data in understandable format to user.

Below, we show you a complete analysis of a use case 3:

"Check position of the trash containers"

This use case describes how the smartphone displays the user, the location of all garbage containers on the map using different markers.

We explain different classes which are involved and then we see the sequence diagram.

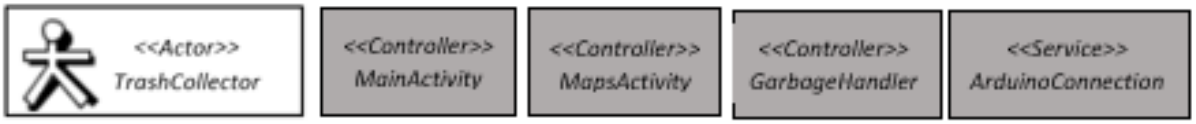

Figure 11: Check position - Involved classes

TrashCollector: The trash collector is an actor. It's him who interacts with the app and see what is displaying on the screen. Therefore, it is not a class.

MainActivity: The controller-class main activity is the menu where you have two features: list of garbage bins and map where you can see the location of garbage bins.

MapsActivity: The controller-class check position is the connection between the user and the smartphone. That allows the user to see on the map the location of all the garbage containers.

GarbageHandler: The controller-class garbage handler is to control database of garbage bins inside the Android.

ArduinoConnection: The service-class Arduino connection allows the trespass of data between database and Xively server.

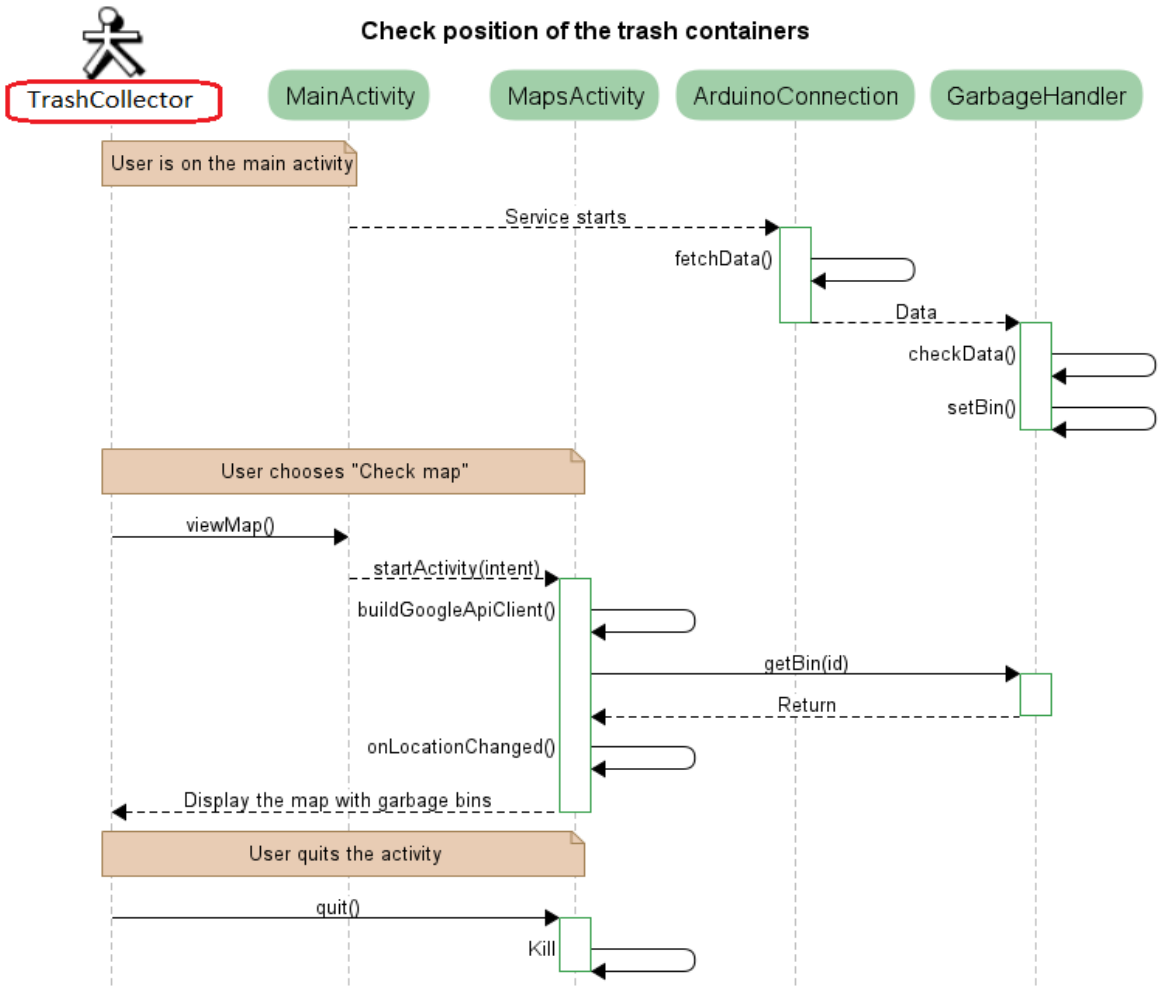

Figure 12: Check position Sequence Diagram

Now, we see the class diagram with all methods implemented in each class. In bold, it's methods that the sequence diagram uses.

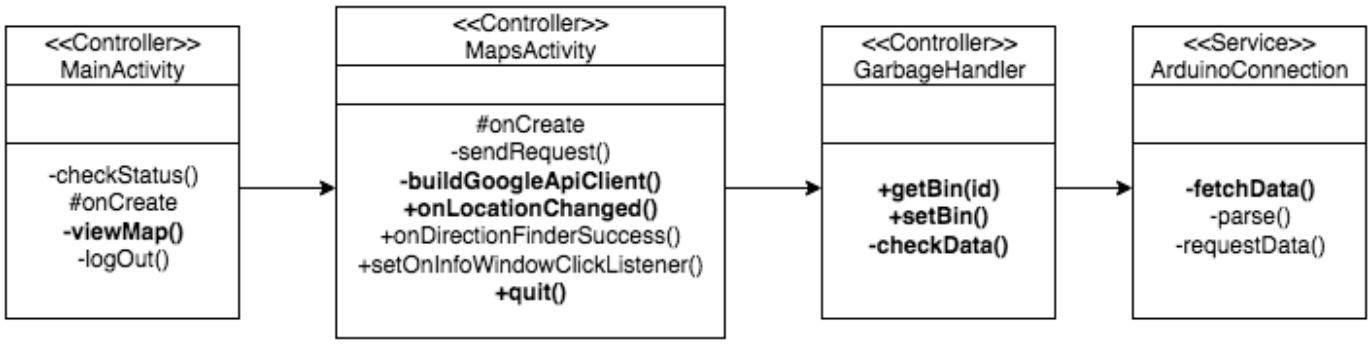

Figure 13: Check position Class Diagram

The process shew above for use case 3 "Check position of the trash containers" is repeated for each use case. This leads to the entire class diagram for phone shown below.
For detailed flow of application and system we must inspect the class diagram for phone.

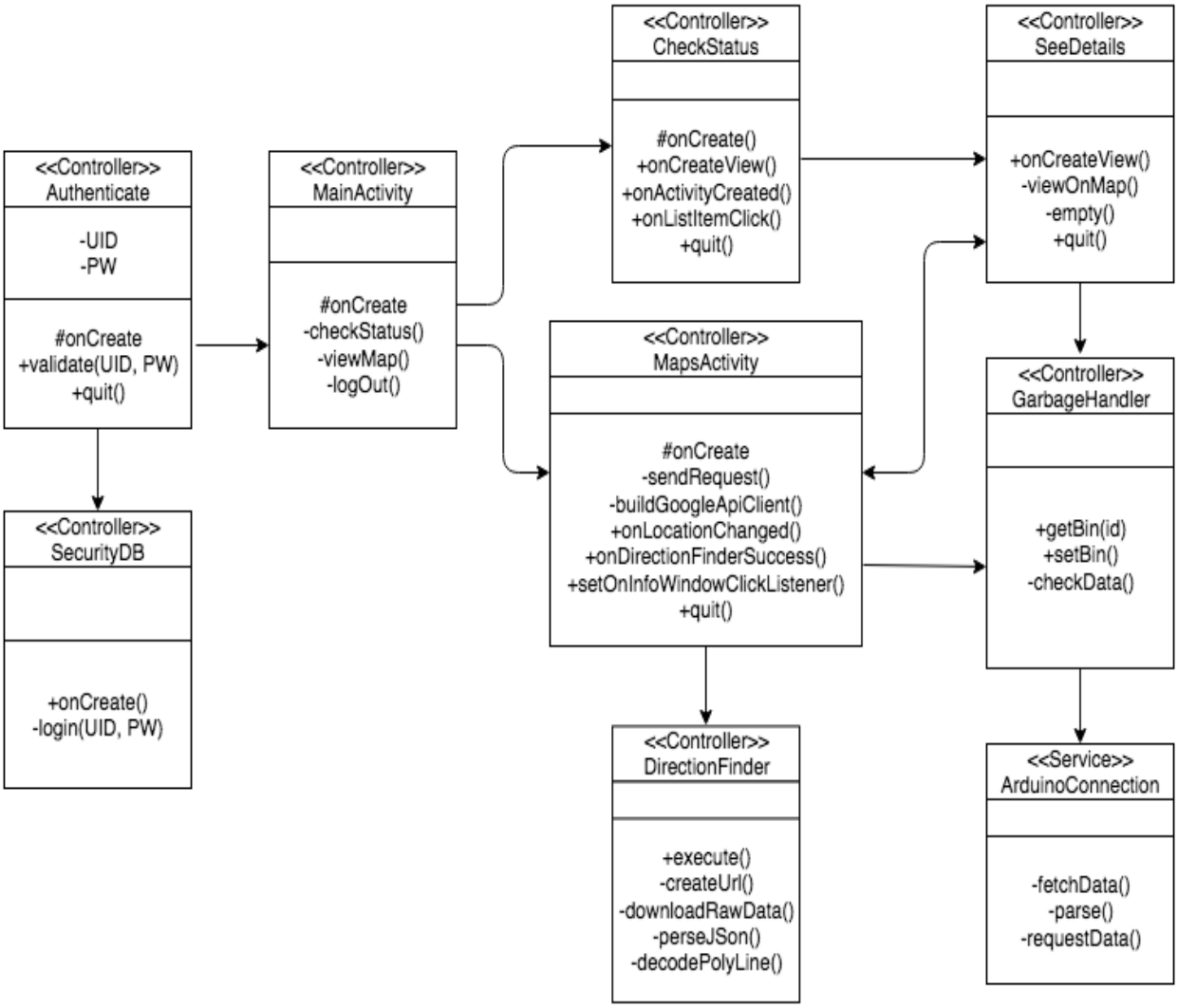

Figure 14: Entire Class Diagram for Phone

With this diagram, we can see structure. It gives us better view of the methods and data flow used to create complete android application. Implementation of said parts is discussed later in the implementation view.

For Arduino code, we have following sequence diagram.

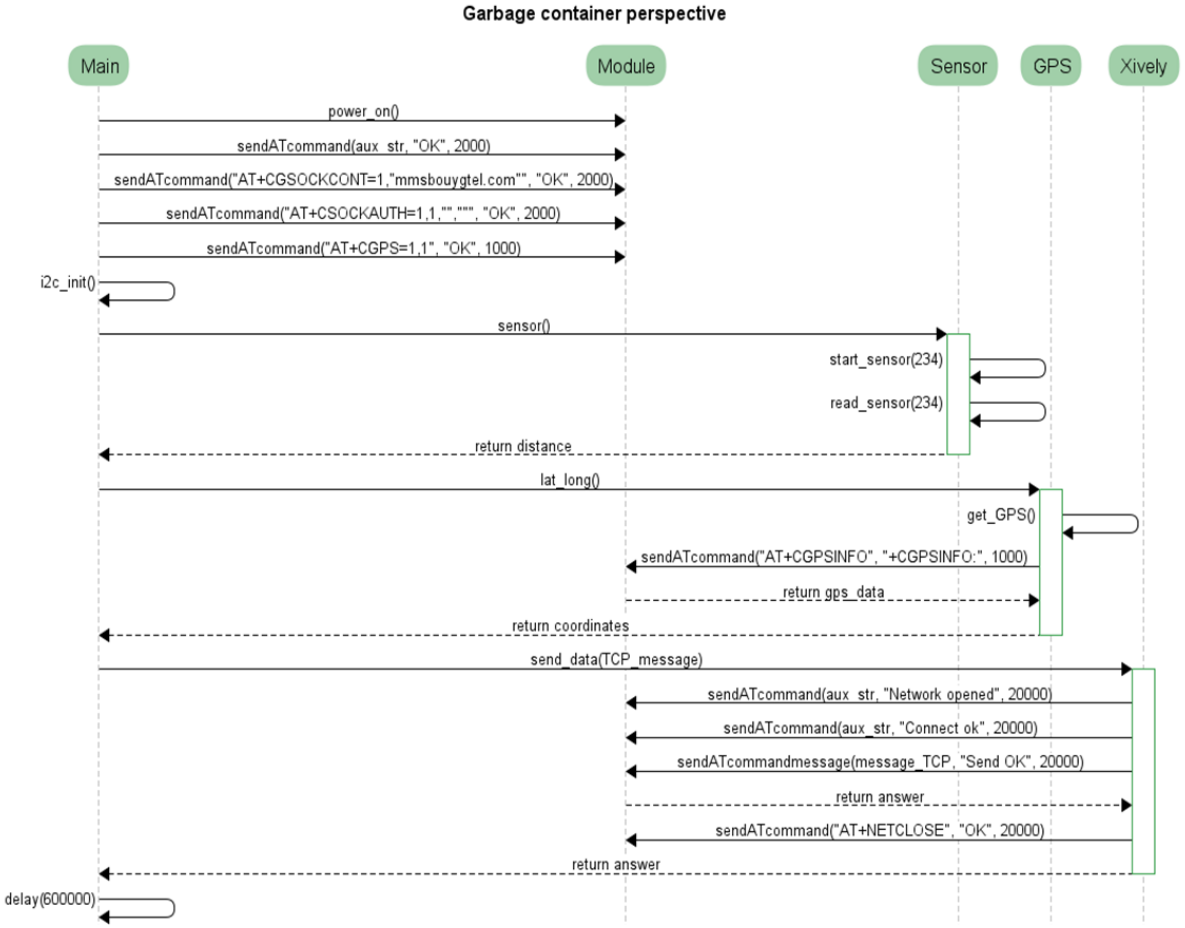

Figure 15: Garbage Container Perspective Sequence Diagram

Which displays sequence and methods used to achieve wanted results inside Arduino. Program needs to gather and send data in a loop to work properly. Loop can be created via interrupts or using Arduinos build in nature to repeat main loop.

5.4 Design

5.4.1 Hardware

Ultrasonic sensor is the part of hardware in our project. This figure show us about connection with ultrasonic sensor and Arduino.

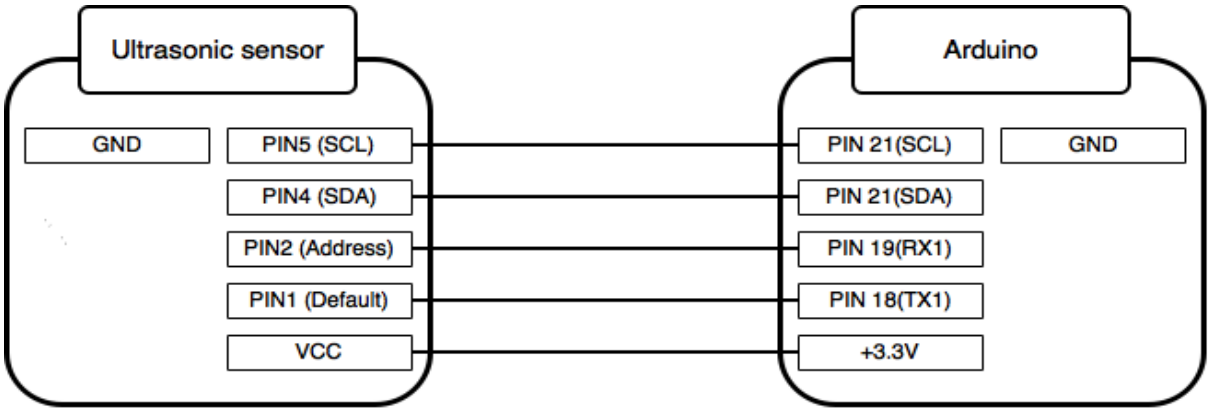

Figure 16: Ultrasonic sensor connection

And this formula is the way how to calculate the distance with sensor and object. General ultrasonic sensor will take the same way to calculate distance.

$$\text{Distance(cm)} = \frac{\left(343\left(\frac{m}{s}\right)\cdot \text{high level time}(\mu s)\right)}{2}\cdot\frac{1}{10^6}\left(\frac{s}{\mu s}\right)\cdot\left(\frac{10^2 \text{cm}}{1\text{m}}\right)$$

5.4.2 Software

We used the N+1 model view as a structural basis for this following part of the document with process, data and deployment view.
For more details, software is described in the document **Architecture & design.pdf [Reference 7]**.

5.4.2.1 Process view

Following section describes interaction between software components of the system. In this section, we described how components interact with each other and which communication channels they need. We split project into two larger software: Mobile app, Arduino.

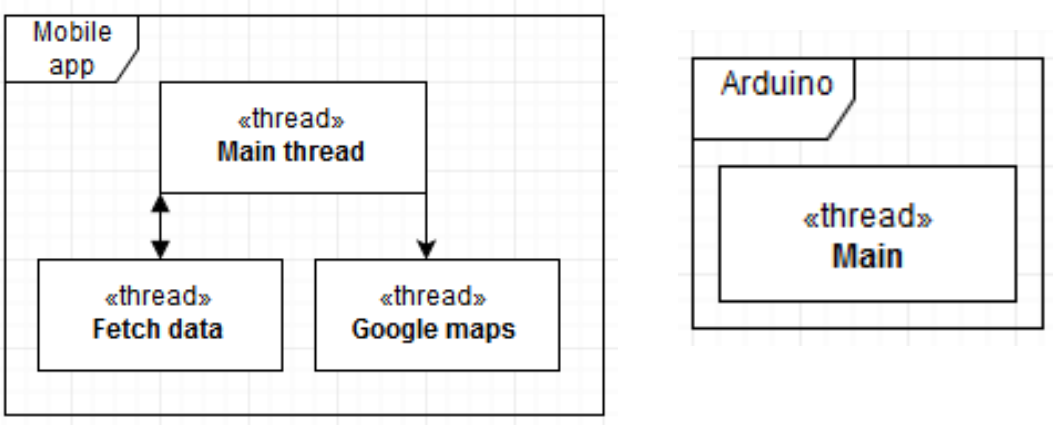

Figure 17: Process view

5.4.2.2 Data view

In our project, we can find two separate databases in the smartphone. To store it, we use a SQLiteDatabase. First, we have the SecurityDB which stores user data to access to the app. Then, we have the GarbageHandler which stores data about trashes specifics.

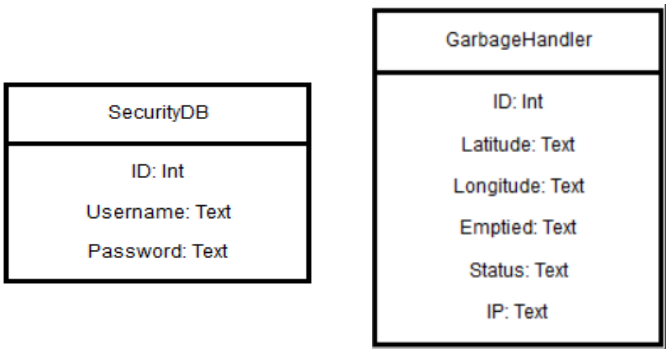

Figure 18: Data view

To send data to the smartphone, we use a personal server (Xively). The microcontroller sends values to Xively through the 3G shield and the smartphone get data using JSON and Volley. It is an HTTP library that makes networking for Android apps easier and faster. Like this, the phone should be able to update Database.

5.4.2.3 Deployment view

The deployment view is relatively straightforward, as the interactions between our components are, as we can see in this diagram, without extensive interrelation.

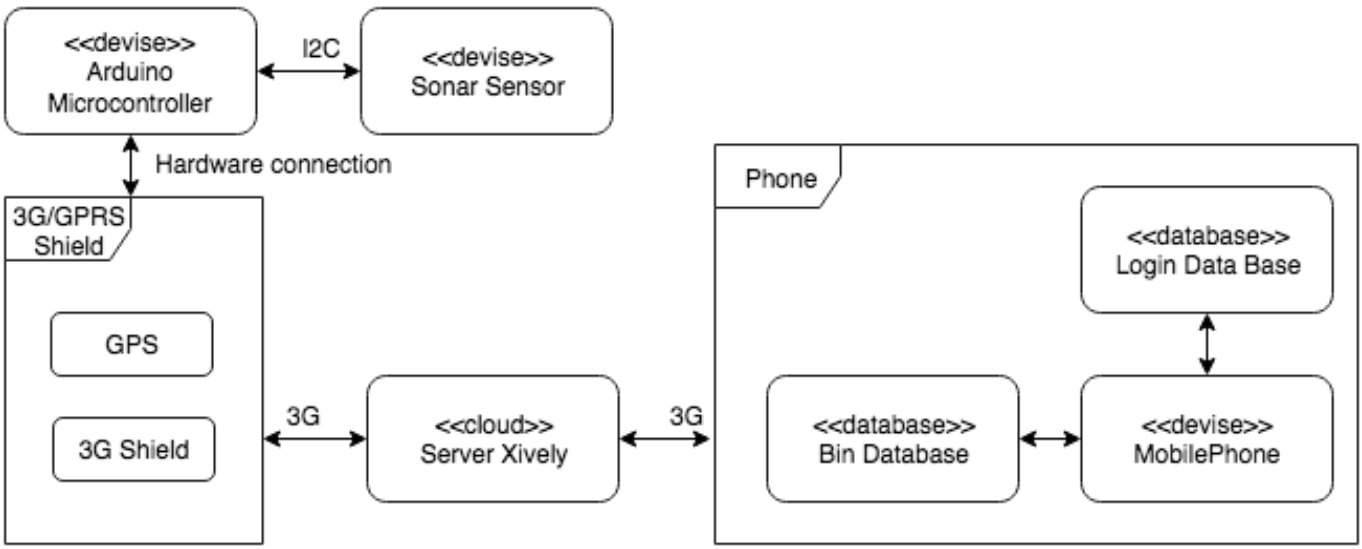

Figure 19: Deployment Diagram

3G Protocol: It allows communication between the microcontroller, the server and the smartphone. We send values “Distance” and “Location” to the server thanks to a TCP

connection. It's a TCP_Message which contains this information.
To get the GPS data, we call the command AT+CGPSINFO which gives us a string with many details of our current location (latitude, longitude, date, altitude for example).

I2C Protocol: To trespass data between the sensor and the microcontroller we use the I2C protocol. This method only uses few wires to communicate. The sensor sends the data when it's requested by writing a one in the address pin.

5.4.2.4 Security view

We describe measures to protect data integrity, availability and confidentiality. Threats and countermeasures will be handled case by case instead linking them to use cases. Linking threats to use cases may leave critical vulnerabilities undiscovered via unintended routes of use.

5.5 Implementation

For talking about the implementation, we divided into two parts: hardware and software implementation.

5.5.1 Hardware

Hardware implementation was done using Arduino Mega2560, Cooking hacks SIM 5218 and MB1202 sensor. Choices are discussed further in the document **Pre-Analysis.pdf [Reference 5]**.

Hardware implementation for our project is simple. 3G shield can only be attached to Mega 2560 in one way and documentation on it is found in shields documentation.

Sensor uses I2C protocol to communicate with Arduino. Sensors pin connection is described in the Arduino programs source code.

5.5.2 Software

Software implementation is done in two parts due project consisting two different platforms. We are firstly going to inspect development on Arduino and later on Android.

5.5.2.1 Arduino

Implementation for Arduino is done with Arduino C and using Arduino IDE. Arduino IDE uses GCC as its compiler.

Arduino functionalities are as follows.

- Sensor reader and data handling
- Getting GPS position
- Data transfer to Xively cloud

Pre-requisites for Arduino software development are as follow.

- Sensor must be connected and working
- 3G sensor must be connected and working

In **Architecture & design.pdf** document **[Reference 7]** pages 36-38 we show code snippets which demonstrate previously introduced functionalities.

5.5.2.2 Android

Android implementation is done using Java and Android studio as IDE. Compiling is done by Android studio.

Android functionalities are as follows.

- Authenticate
- Fetch data from Xively and parse it
- Create GarbageBin object and store it in database
- Display location and status of GarbageBin to user
- Route to GarbageBin

Pre-requisites

- Java SE Development kit is installed
- Android Studio has been configured
- Dependency libraries have been added (Volley)

Android program for our project is extremely modular due to nature of Android application activity lifecycles. In the source code, you can see app split into multiple activities, database handlers and one service.

Inside Android program we have created GarbageBin object which is explained in **Architecture & design.pdf** page 35 **[Reference 7]**.

In the Authenticate activity we handle login. User inputs credentials which are then sanitized and checked against authentication database using SecurityDB database handler. Successful login leads user to MainActivity.

While creating MainActivity we also start service ArduinoConnection which gets data from Xively and stores GarbageBin object into database using GarbageHandler. From MainActivity we can enter list of garbage bins or MapsActivity.

List of GarbageBin takes use of GarbageAdapter which is tool to display GarbageBin objects in list form. From the list, we can choose GarbageBin object and get more detailed view of selected GarbageBin.

From detailed view, we can empty the bin or view in on the map. Launching MapsActivity from DetailsActivity or MainActivity puts markers of GarbageBins in to the map and allows navigating to them using google services. From the MapsActivity we can also go to detail view by clicking the markers.

6. Test

In this part, we describe acceptance test for garbage container collection system. Requirements to pass the tests can be found in the **Requirement specification.pdf** document **[Reference 6]**.

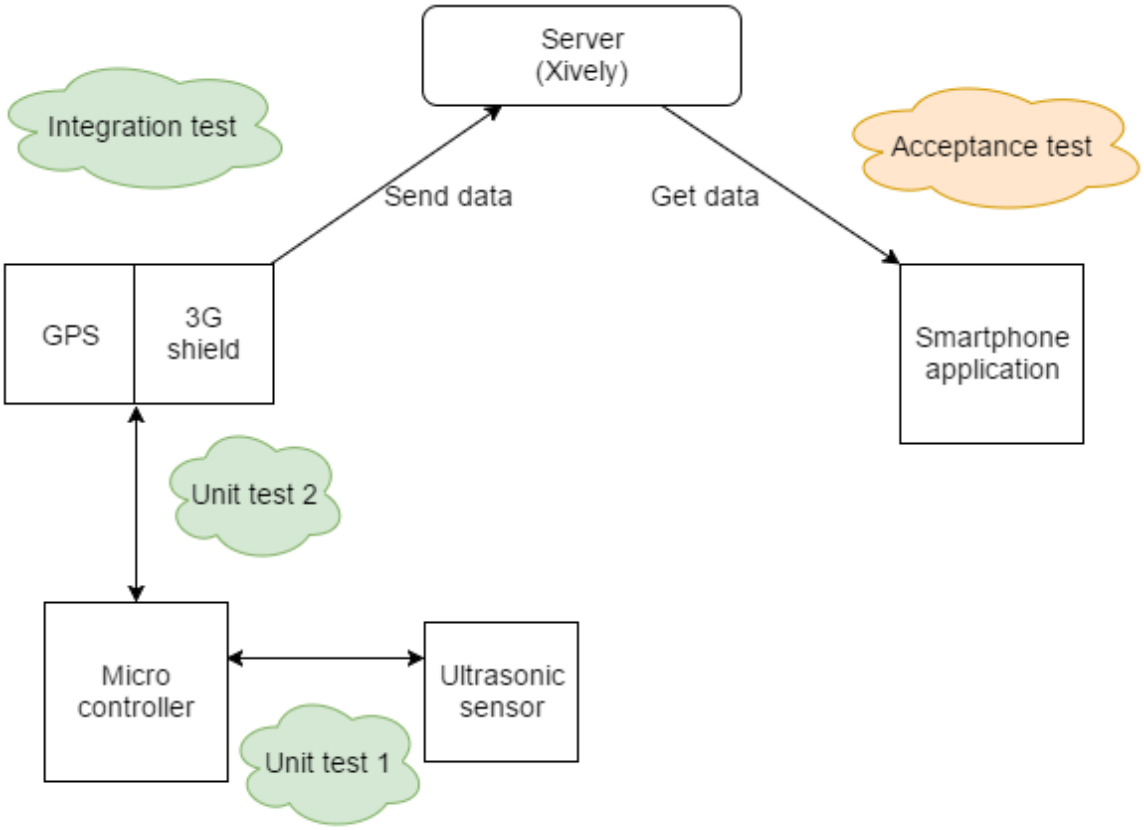

Figure 20: Test Diagram

This table below is about our components list tested.

Device	Environment	Notes
Microcontroller	Arduino Mega2560	
Ultrasonic sensor	MB1202	
Smartphone	Android	API 24+
Communication module	SIM5218E	AT Commands

Table 3: Components tested

6.1 Unit Test

6.1.1 Sensor

This test is to see if sensor data can be read. Putting obstacles on different distances and comparing the real distance with what sonar sensor sends to Arduino microcontroller. We can know how is the sonar accuracy. After that, we try to test it in a real garbage bin.

Different tests we did, were successfully and now we can say that the ultrasonic sensor can be implemented on our system. Below, there is a picture of how we did the real test in the garbage bin. For more details, you can take a look in the **Test.pdf** document **[Reference 8]**.

Figure 21: Sensor unit test measurement in real garbage bin

We measured the approximate distance and we could see that it was around 80cm. The sensor response was this:

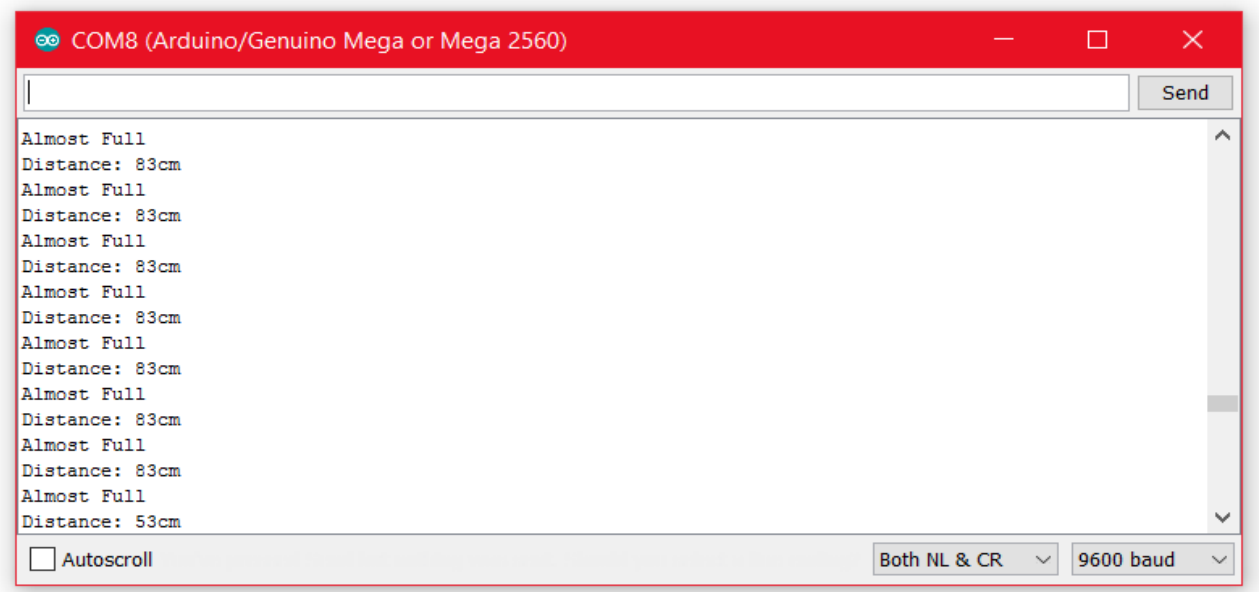

Figure 22: sensor unit test getting data from real garbage bin

6.1.2 3G Shield

This test is used to see if the 3G shield is functioning and can be used in our project.

The first part of the 3G shield test code is checking the GPS signal. You can check the latitude value and longitude value through this figure 23.

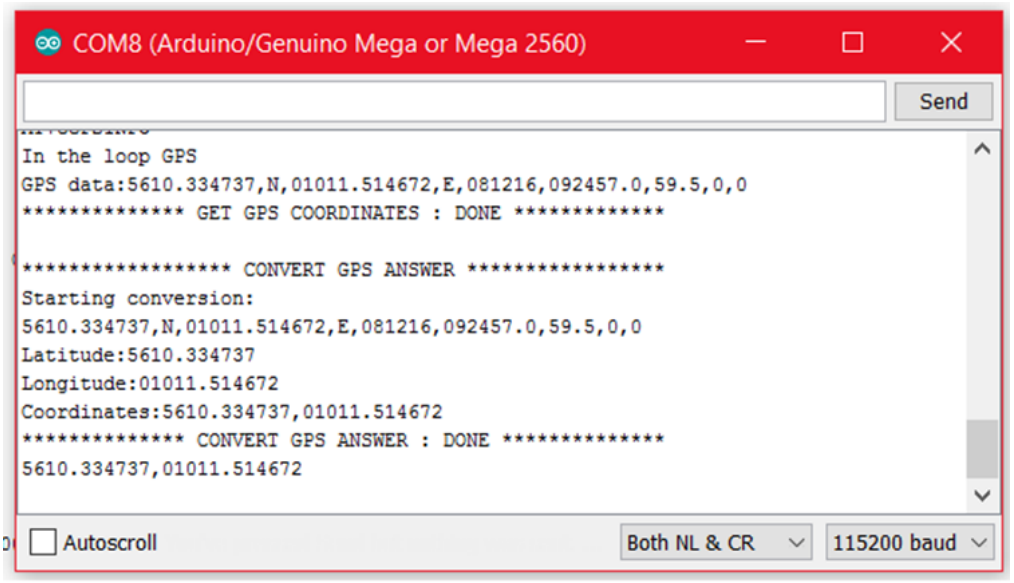

Figure 23: 3G shield unit test getting location

After that, the microcontroller has to create a connection with Xively to send the message which contains our values. As you can see in figure 24, the network and the socket are opened and closed successfully. Therefore, the message has been sent.
For this part, we create values. Distance = 45 and Location = 5610.322888, 01011.539104.

```
********************* SEND MESSAGE ********************
Start send data
Opening network
AT+NETOPEN="TCP",8081
Network opened
Opening socket
AT+TCPCONNECT="api.xively.com",8081
Socket opened
AT+TCPWRITE=293
{"method": "put","resource": "/feeds/1819176350/","params": {},"headers": {"X-ApiKey":
Message send
AT+NETCLOSE
Network closed
***************** SEND MESSAGE : DONE *****************
```

Figure 24: 3G shield unit test getting data

As we can see, between the data we faked and the data stored in Xively personal database it is the same. Therefore, the connection between Arduino board and Xively personal cloud is working correctly.

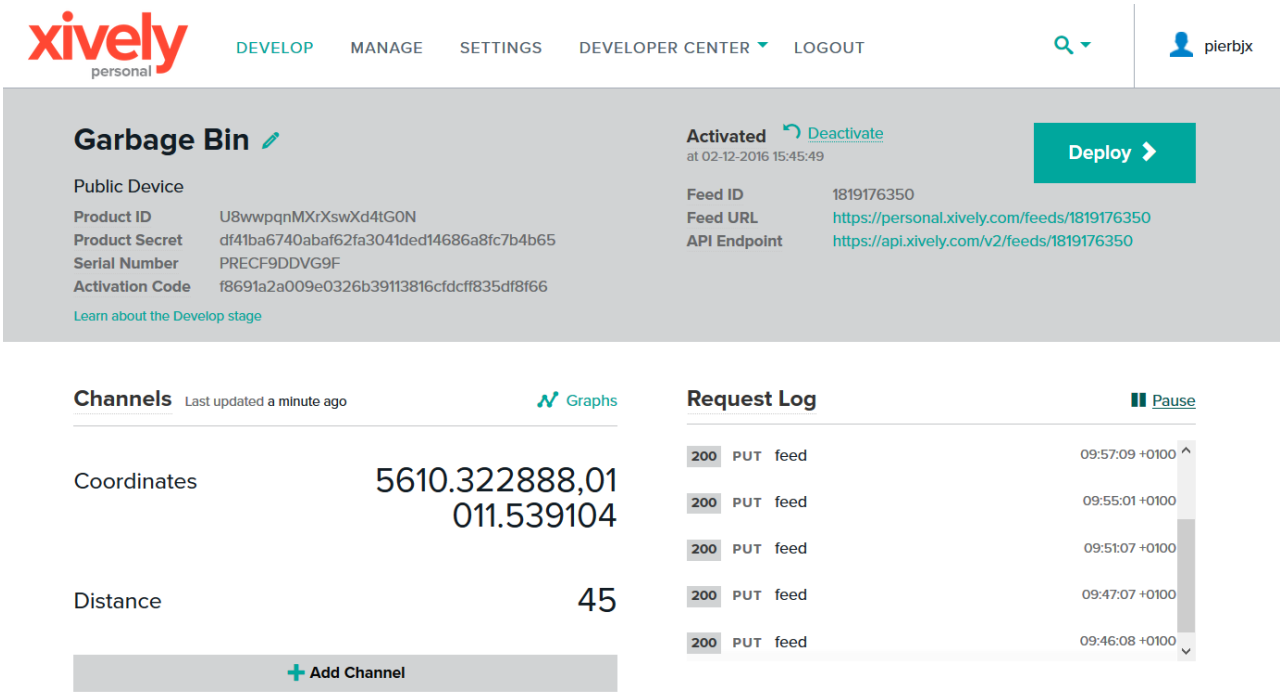

Figure 25: 3G shield unit test Xively personal database

For more details, consult the **Test.pdf** document **[Reference 8]**.

6.2 Integration Test

This test is done to see if the real data from the ultrasonic sensor can be sent from the Arduino to the Xively personal cloud.

We wrote code to implement the final connection between entire system. So now the first step is turn on the entire system and set up all pins. After that the ultrasonic sensor is going to get the distance and the microcontroller stores it. Besides system is waiting for GPS answer, but sometimes it can be long because of the weather or place we are. In the next picture, we can see all this process.

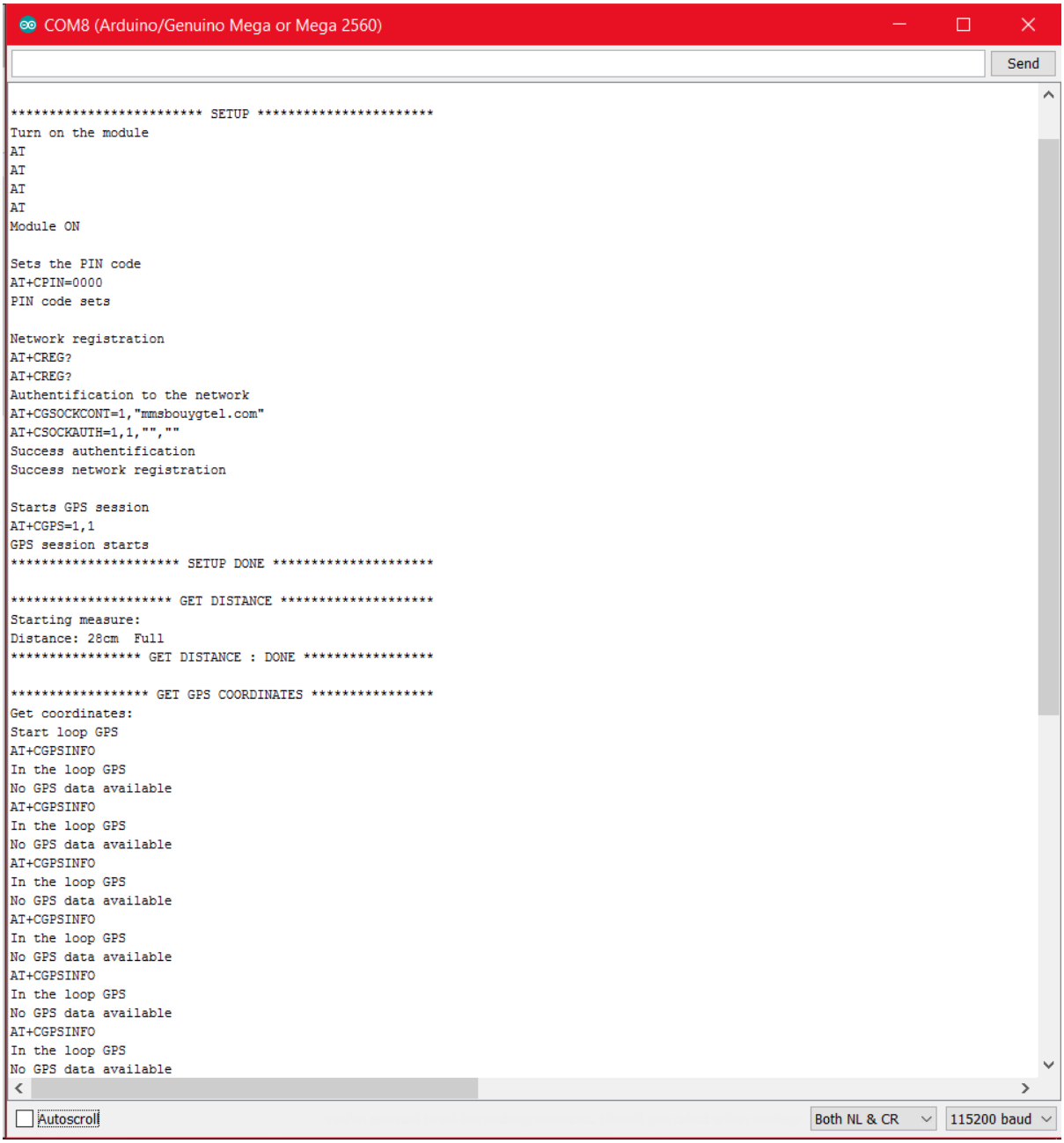

Figure 26: First step of integration test

Finally, after a short waiting time, the microcontroller gets the coordinates.
Once microcontroller has all the data it proceeds to send it to the Xively cloud. Thenceforth the code has a delay of 10 minutes before it starts the loop again.

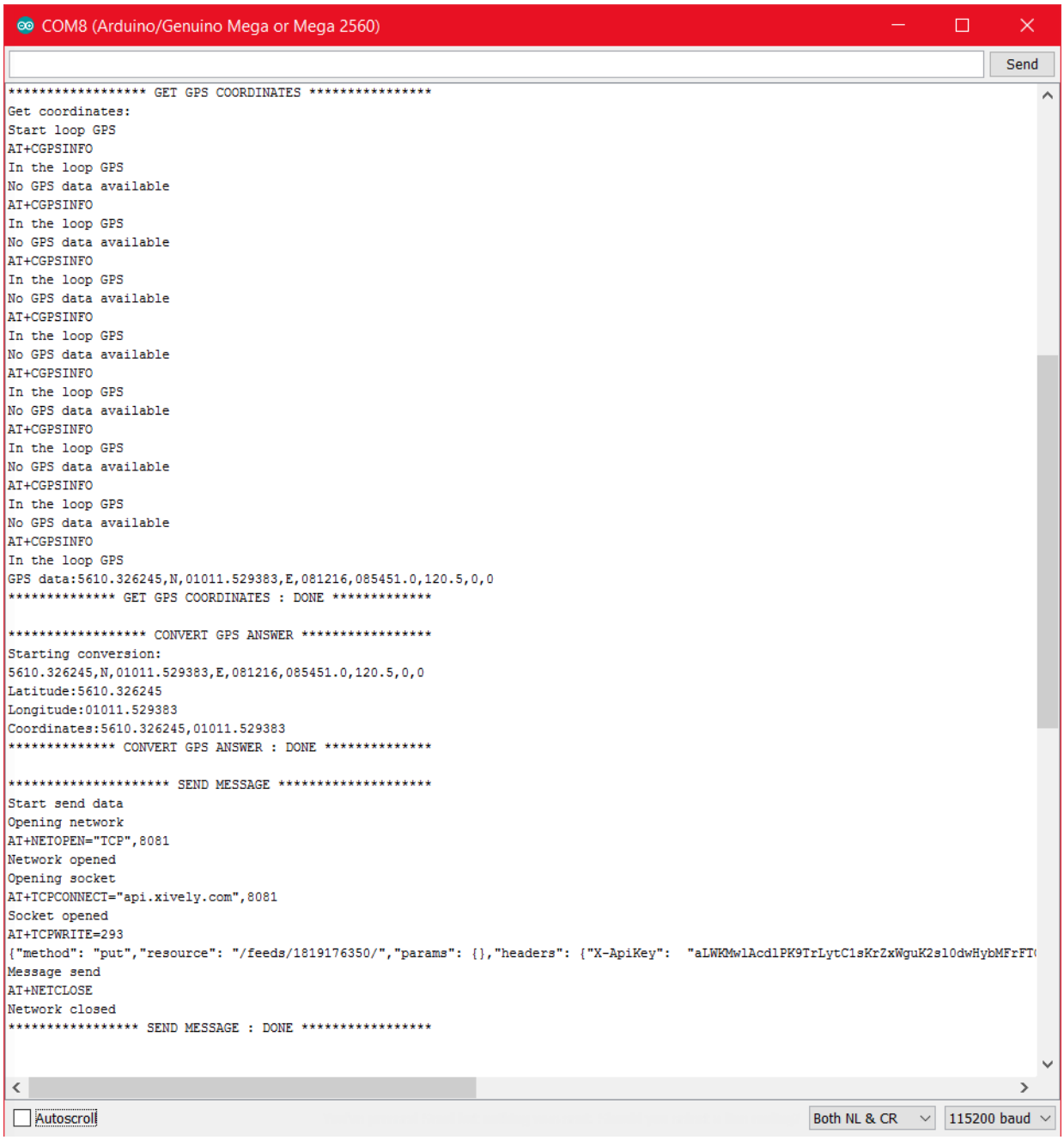

Figure 27: Second step of integration test

Here, microcontroller got a distance of 28 cm, latitude of 5610.326245 and longitude of 01011.529383. We check on the Xively and we can see that data is stored there, so the system works perfectly.

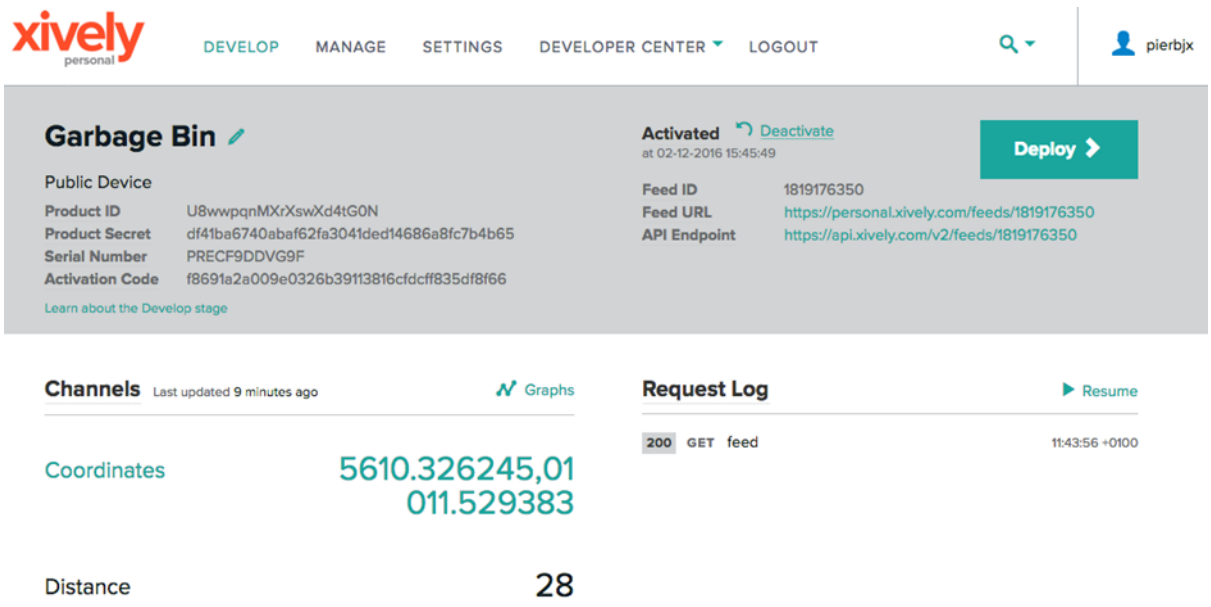

Figure 28: Xively after the integration test

6.3 Acceptance test

In this part, all the acceptance tests are done through checking the status of our miscellaneous use cases tests described previously. These use cases tests consist mainly of using the application and its interface.

Starting for the authentication test, use case 1, is used to see that authentication works and is secure. It is done through verifying if our authentication system is working properly, i.e. if login works as intended and if there is no escape mechanism in the login part that could be used to modify our database through the login system.

After that, the acceptance test proceeds to check status of garbage bin test, use case 2, which is to see connection between android phone and system. And now proceed to check position of garbage bin, use case 3, which is to see if garbage bins are shown correctly on a map inside smartphone application.

Next step for the acceptance test is to check if we can find more information in see details, use case 4. The next use case: find the best garbage collection route, we tested if the app could navigate from current position to desired garbage bin.

Finally, acceptance test proceeds to test if we could modify data inside smartphone to match status of garbage bin, for the use case 6: change status of garbage bin.

All acceptance test got a correct result, for more details see the **Test.pdf** document **[Reference 8]**.

7. Results

At the initial stage of the Garbage bin project we have set the goals in Project Description report that we have planned to achieve. In this section, the accomplished results will be overviewed.

The app was designed to show to the user (garbage collector): the list of garbage bins, how full they are, update status, see them on a map and find the faster route to the selected bin. The entire app works well. Nevertheless, only one garbage container is correctly stored in the Xively server since we only have one prototype system. We create others bins in the database of the smartphone to have a better vision of what the result should be.
Therefore, the app is ready to be a good prototype and our results are convincing.

All the structure of the garbage bin, that includes microcontroller, 3G shield and ultrasonic sensor, has been split into different parts to gain a specific result.

In the case of the sensor, our results were what we expected. Sensor gets the data with little accuracy (± 2cm), but in our case it doesn't really matter, given that we just use a scale of three steps in a range of 2 meters.

In the case of the GPS, we used AT command (AT+CGPSINFO) to get the location and we convert it into a string to only have the latitude and the longitude. Microcontroller gets data; however, there could be a waiting time.

In the case of the connection, the microcontroller sends data to a personal server Xively through the 3G shield after getting data. Then, the smartphone fetches data in the server.

Thus, our results are completely satisfactory and we can say that our expectations have been fulfilled.

8. Discussion of Results

8.1 System

For the system, our goal was to get all data required to display information on the smartphone. The C code was developed gradually while we were testing the correct operation of each one of the parts, such as the sensor data gathering and the server data sending through 3G connection.

This part of our project was a discovery for all of us given that most of us never code Arduino before. Therefore, we learned by ourselves and we encountered difficulties.

First of all, in the case of the connection, it was planned to make a 3G connection directly from microcontroller to smartphone using a TCP server but we got some problems and we changed the way to connect the microcontroller with the smartphone.
After some researches on the net, we had the idea to use a cloud server which is easier but needs more code. We opted for the Xively personal server. Indeed, to fetch data from the server to the smartphone it is the same protocol that we used for ITSMAP's lab exercise.
Then, we had to find the way to send data to the server but thanks to some tutorials and forums we find a way to send it.

Moreover, to get the GPS data we tried different modes. Indeed, it works with the standalone mode but the problem is that there is a waiting time to obtain location. Therefore, we tried others modes as A-GPS or S-GPS but we didn't succeed given that we had to use a GPS server and we didn't how to do this.

However, in the case of the sensor we didn't encounter difficulties. The sensor works immediately and we get the range easily.

8.2 Application

For application, our goal was to be able to fetch data outside the system and created object to manage garbage bins in the system. Our prototype for data was limited to one container for garbage but application was designed to handle multiple objects.

The Java code for the APP was developed by 3 members of the group, that used it as a final project for ITSMAP subject. That was a great help in order to save time and we take profit of the situation.

This part of our project was also part of different course with different goals. Combining two different goals led to some non-optimized parts in the application. For the learning experience this was good but all parts don't necessarily use best practices.

Applications functionality works as needed to fetch, parse and display objects. Application is also error resistant for most user input. Most user input errors are caught and don't crash the application.

User interface and support for different machines is lacking and was not focus of this project. Possibility to improve it was with course about user experience. Our team opted to keep focus on software and functionality.

With use of databases and objects we managed to create scalable and functional system to monitor IoT devices. With object oriented coding it is possible to implement it to other type of data easily.

In conclusion, we managed to make IoT device which transmit live data to android application. We also used cloud services which makes IoT experience of our project even larger.

9. Future Work

As described in the limits (page 10), we only made one prototype. Therefore, it needs more garbage container to apply it in the real life. After adding more garbage containers, we should check if it works well in our system.

We had a small problem with GPS signal reception because it doesn't work each time. So, we could use A-GPS or S-GPS mode, that are additional operating modes of the Cooking Hacks. We also need to raise precision of distance data, choosing another sensor with less distance measurement.

As a possible improvement for our project, we thought we could have add more than one sensor in order to have a better accurate measure of the distance.

10. Conclusion

The realization of this project for the ITPRJ subject, allowed us to put into practice all the knowledge acquired during these courses (ETCCP and ITIPRJ), and not only the subject of ITIPRJ, but also of all the subjects we have been studying during our careers as engineers. Combining the concepts learned in different subjects has been the key to forward all stages of the project, such as hardware design, writing official documents for the project, programming microcontroller and developing an APP to control it.

We realize that the magnitude of this project is greater than we expected at first, and small obstacles that we found in our way prove it. However, we believe that this activity is very interesting and useful, that gives the opportunity to work in teams, and test our limits as future engineers.

As for the course, we can say that the main objectives have been achieved and we are really proud of it.

Regarding the content of the project, after all the work done, we think it's finished successfully with the verifying correct operation of the APP combined with the server and the microcontroller. We can check on the smartphone the status and the position of the garbage container that we used for the prototype and update it when something changes. We should say that we used two different software programs, one for the app development, that is Android Studio, and the other for the microcontroller programming, that is C for Arduino.

We split up the project in small tasks and each of us was responsible to develop his part as we explained previously.

We can say that we achieved the objectives we set at the beginning, despite the different contingencies that have arisen during the course and have had to overcome one by one. As world issues, we can highlight on one hand, getting the 3G connection working, as it was the first time for all of us to work with this type of communication. But after 3 weeks searching on the net, reading tutorials and asking some people, we succeed. On the other hand, we had the APP and server combination, as at first, we couldn't succeed in updating the data, but now it's not a problem anymore.

The best part of our project was that we could test each part separately without affecting the other parts and it was easier to put all the parts working together.

We would like to remark that the learning acquired in the Project part of ETCCCP subject and the meetings we had each week with the project supervisor, have been a huge help to develop this project.

Finally, we can draw a very important conclusion: all the problems that we have arisen, helped us to better understand how works the entire system and specially the software part, as there is not a lot of hardware in our prototype.

11. References

[Reference 1]: **Use case:** http://searchsoftwarequality.techtarget.com/definition/use-case

[Reference 2]: **SysML & UML:** https://www.visual-paradigm.com/features/uml-and-sysml-tools/

[Reference 3]: **N+1:** https://en.wikipedia.org/wiki/4%2B1_architectural_view_model

[Reference 4]: **Project description.pdf**

[Reference 5]: **Pre-analysis.pdf**

[Reference 6]: **Requirement specification.pdf**

[Reference 7]: **Architecture & Design.pdf**

[Reference 8]: **Test.pdf**

12. List of Figures