|  |
| --- |
|  |
| **Monitoring the status of the garbage containers** |
| Test |
|  |

|  |
| --- |
|  |

**Authors:** Xavier Cerqueda Puig

Bernat Garcia Torrentsgeneros

Pierre Biojoux

Quentin Studeny

Junyoung Bang

Joonas Luukkanen

**Supervisor:** Torben Gregersen

**Date:** 14/11/2016

# 

# Table of contents

[Table of contents 2](#_Toc469317906)

[2. Introduction 4](#_Toc469317907)

[3. Test specifications 4](#_Toc469317908)

[4. Test setup 5](#_Toc469317909)

[5. Unit tests 5](#_Toc469317910)

[5.1 Sensor 5](#_Toc469317911)

[5.1.1 Description 5](#_Toc469317912)

[5.1.2 Pre-requisites 10](#_Toc469317913)

[5.1.3 Test procedure 10](#_Toc469317914)

[5.2 3G Shield 11](#_Toc469317915)

[5.2.1 Description 11](#_Toc469317916)

[5.2.2 Pre-requisites 14](#_Toc469317917)

[5.2.3 Test procedure 14](#_Toc469317918)

[6. Integration tests 14](#_Toc469317919)

[6.1 Definition 14](#_Toc469317920)

[6.2 Description 15](#_Toc469317921)

[6.2 Pre-requisites 17](#_Toc469317922)

[6.3 Test procedure 17](#_Toc469317923)

[7. Acceptance tests 18](#_Toc469317924)

[7.1 Use case 1: authentication test 18](#_Toc469317925)

[7.1.1 Description 18](#_Toc469317926)

[7.1.2 Pre-requisites 18](#_Toc469317927)

[7.1.3 Test procedure 18](#_Toc469317928)

[7.2 Use case 2: check status of garbage bins 18](#_Toc469317929)

[7.2.1 Description 18](#_Toc469317930)

[7.2.2 Pre-requisites 18](#_Toc469317931)

[7.2.3 Test procedure 19](#_Toc469317932)

[7.3 Use case 3: check position of garbage bins 19](#_Toc469317933)

[7.3.1 Description 19](#_Toc469317934)

[7.3.2 Pre-requisites 19](#_Toc469317935)

[7.3.3 Test procedure 19](#_Toc469317936)

[7.4 Use case 4: See details of garbage bin 19](#_Toc469317937)

[7.4.1 Description 20](#_Toc469317938)

[7.4.2 Pre-requisites 20](#_Toc469317939)

[7.4.3 Test procedure 20](#_Toc469317940)

[7.5 Use case 5: Find the best garbage collection route 20](#_Toc469317941)

[7.5.1 Description 20](#_Toc469317942)

[7.5.2 Pre-requisites 20](#_Toc469317943)

[7.5.3 Test procedure 20](#_Toc469317944)

[7.6 Use case 6: Change status of garbage bin 20](#_Toc469317945)

[7.6.1 Description 20](#_Toc469317946)

[7.6.2 Pre-requisites 20](#_Toc469317947)

[7.6.3 Test procedure 21](#_Toc469317948)

# 2. Introduction

In following document, we will describe acceptance test for garbage bin collection system. Requirements to pass the tests can be found in the document ‘Requirement specification’.

# 3. Test specifications

In table 1 we list components being tested.

|  |  |  |
| --- | --- | --- |
| **Device** | **Environment** | **Notes** |
| Microcontroller | Arduino Mega2560 |  |
| Ultrasonic sensor | MB1202 |  |
| Smartphone | Android | API XX+ |
| Communication module | SIM5218E | AT Commands |

*Table 1 List of devices*

# 4. Test setup

Figure 1 shows testing setup for completed project. Some tests may be performed before final prototype is completed.

*Figure 1 Test setup*



From the figure 1 we can see that testing environment consists of two big section. For most use cases these sections can be tested separately. Full system also includes one external actor which is mobile operator.

# 5. Unit tests

## 5.1 Sensor

### 5.1.1 Description

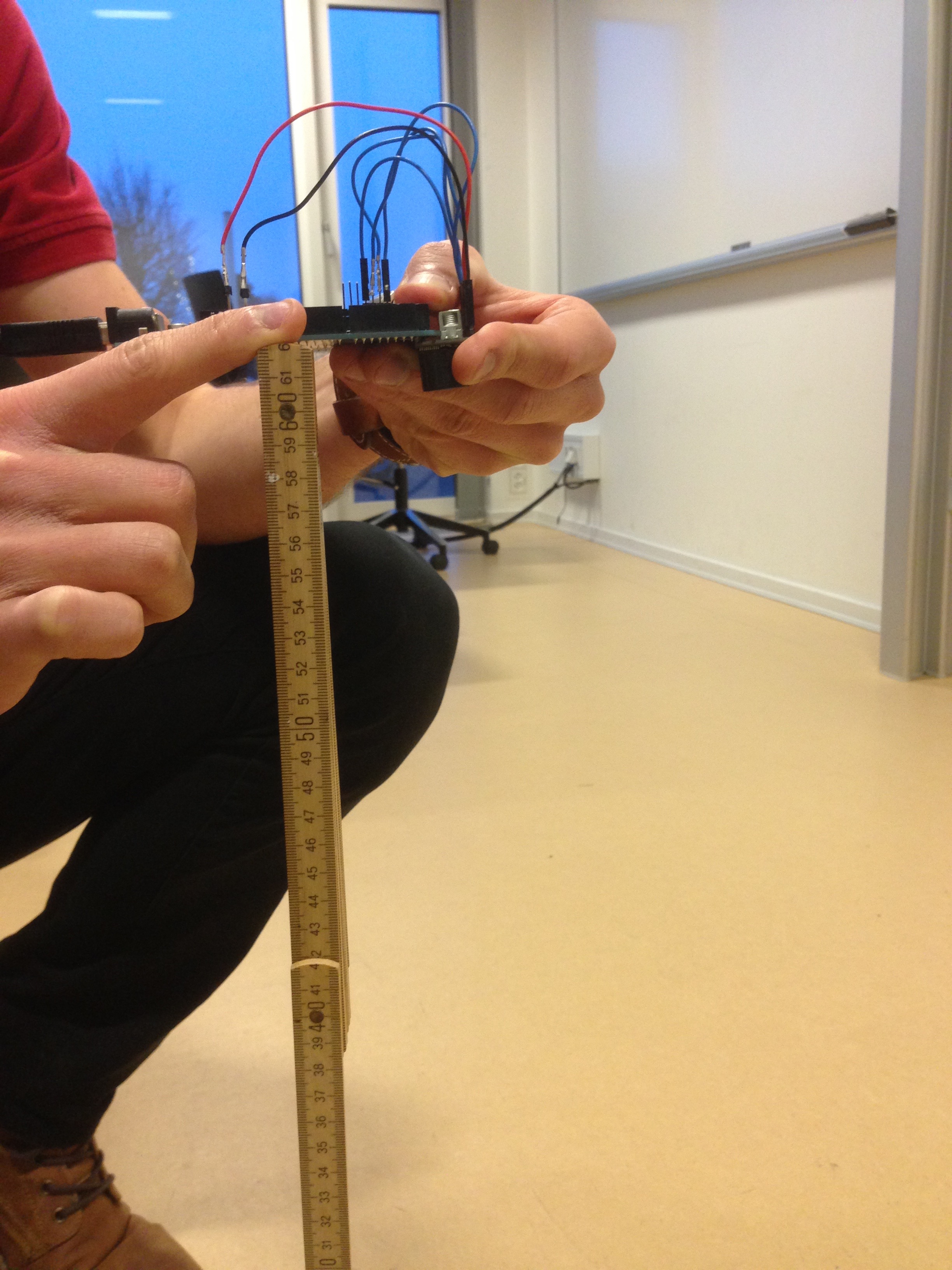
This test is to see if sensor data can be read.

For this one we had to write the code to show the sensor's results first. Thus, unit test and implementation test were done simultaneously. After linking code, sensor and microcontroller, we confirmed that the sensor was receiving and delivering input & output. Then we did some basic range test on it, to confirm its adequacy; first putting obstacles in an open environment and then in an empty trash.

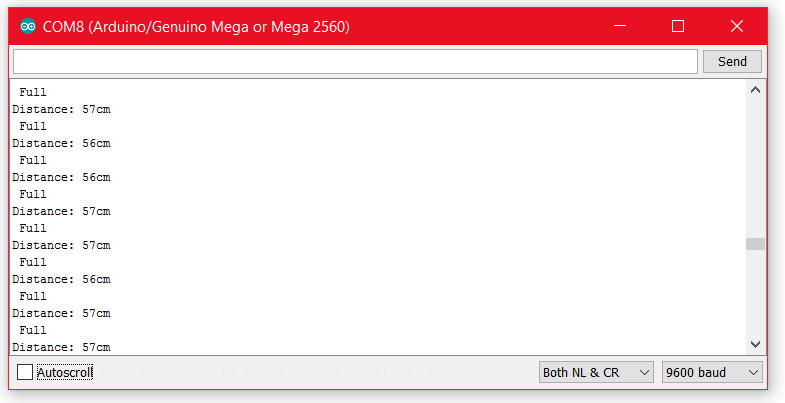
Putting obstacles on different distance, and comparing the real distance with what’s sonar sensor are sending to Arduino microcontroller, we can know how is the sonar accuracy. First, we tested with three different distances, just putting the sensor focus to ground.

And, we could compare the real distance with sonar response (display in the serial monitor). As you can see following we take different pictures while we was doing the test.

0,6 meter:

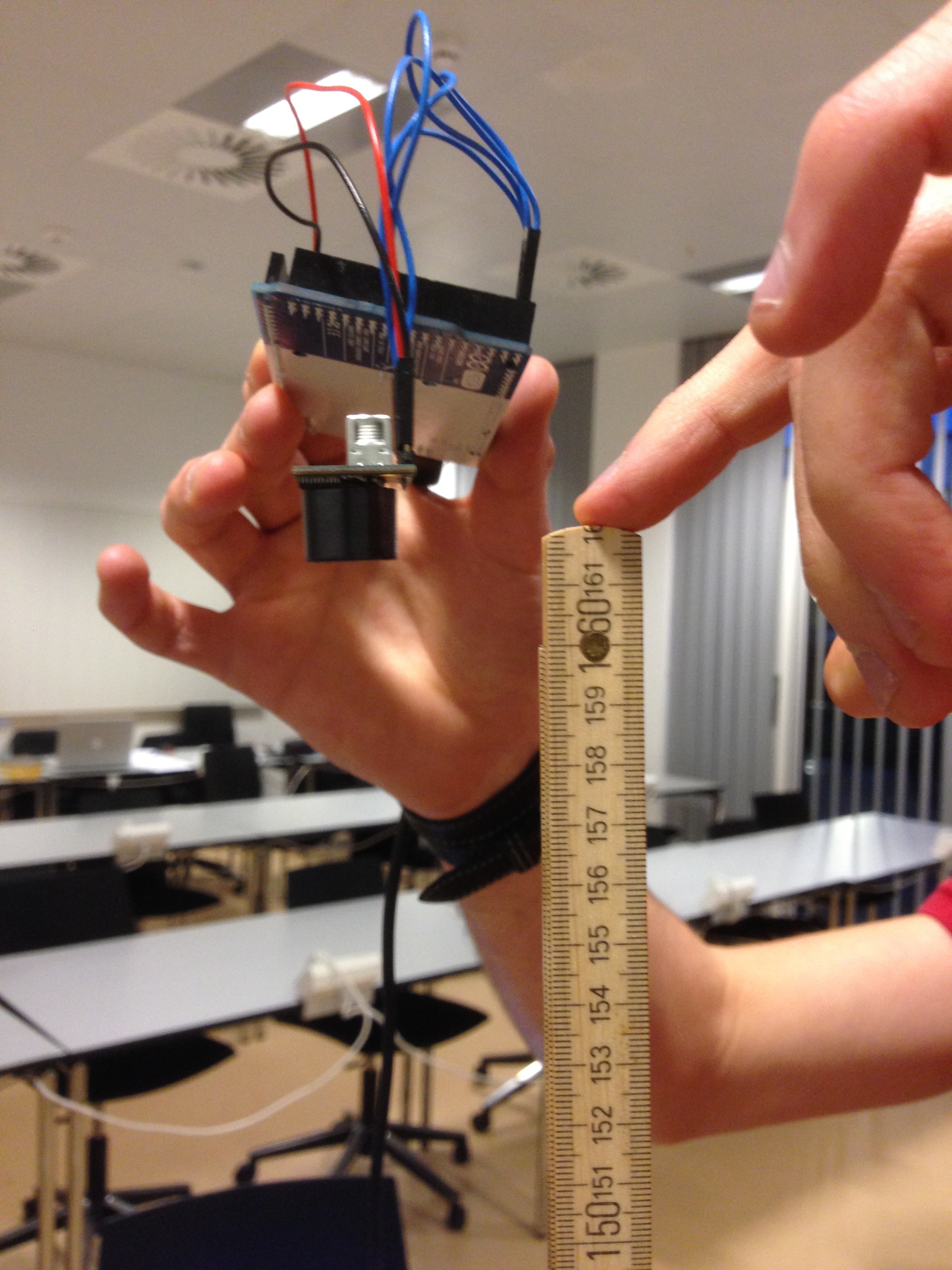


*Figure 1: 0,6m measurement sensor unit test*

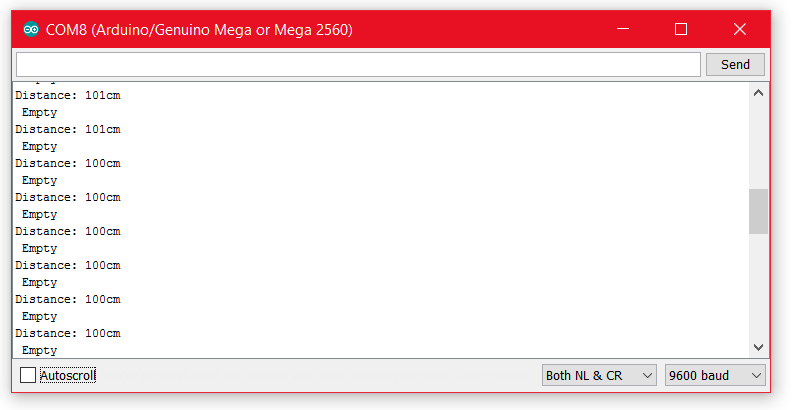


*Figure 2: sensor unit test getting data 0,6m*

1 meter:

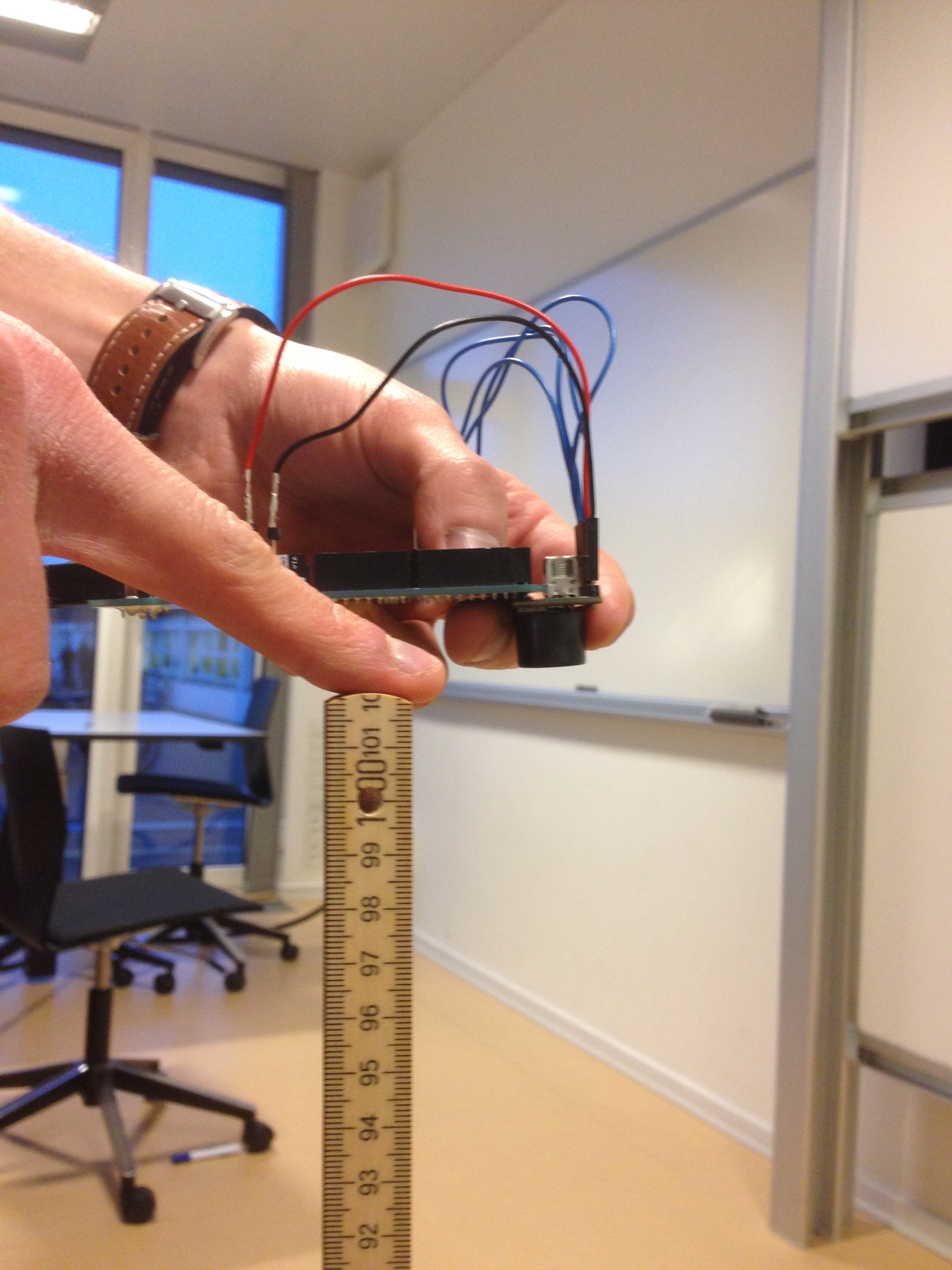


*Figure 3: 1m measurement sensor unit test*

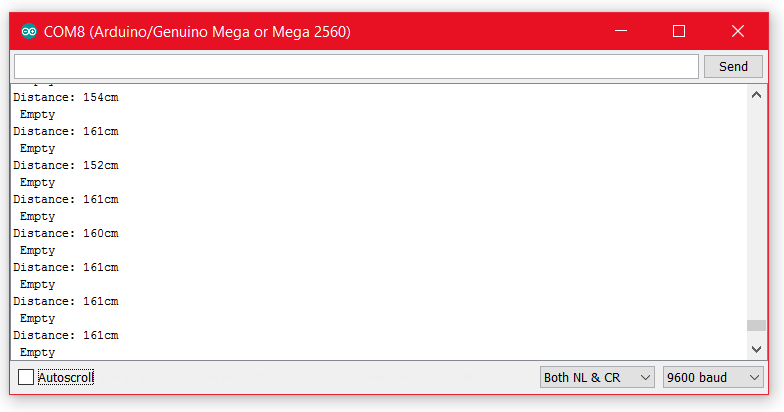


*Figure 4: sensor unit test getting data 1m*

1,5 meter:



*Figure 5: 1,6m measurement sensor unit test*



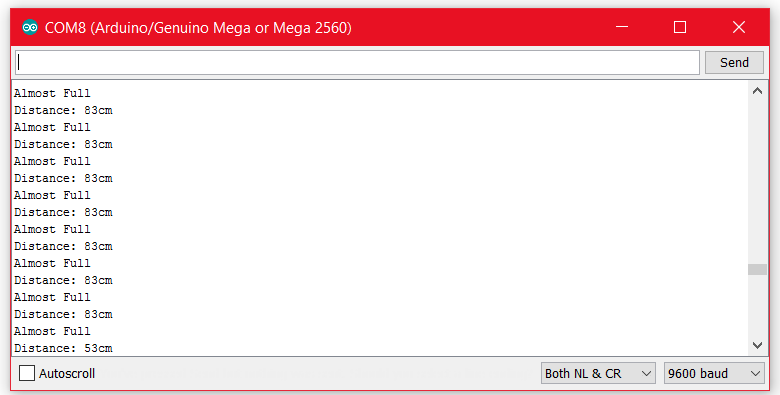
*Figure 6: sensor unit test getting data 1,6m*

After that we proceeded to test in the real garbage bin. Then we take a random garbage bin and proceed to test it, first with an empty garbage bin and then we put an obstacle to simulated garbage content.

+

*Figure 7: sensor unit test measurement in real garbage bin*

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



*Figure 8: sensor unit test getting data from real garbage bin*

Now the garbage bin system without 3G shield it is completely working well.

There is no real acceptance test for the sensor, but some of the other acceptance tests are relying on the sensor to work.

### 5.1.2 Pre-requisites

* Sensor has been connected to Arduino
* Arduino has been programmed properly

### 5.1.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Set up the sensor | Set up is done | OK |
| 2 | Get read from sensor | Data read successful | OK |
| 3 | Show sensor data | Sensor data is shown in serial console | OK |

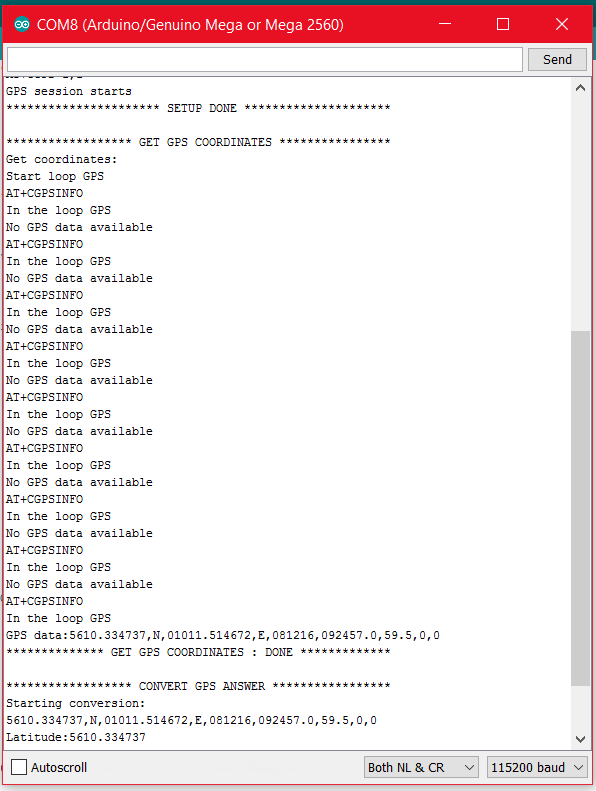
## 5.2 3G Shield

### 5.2.1 Description

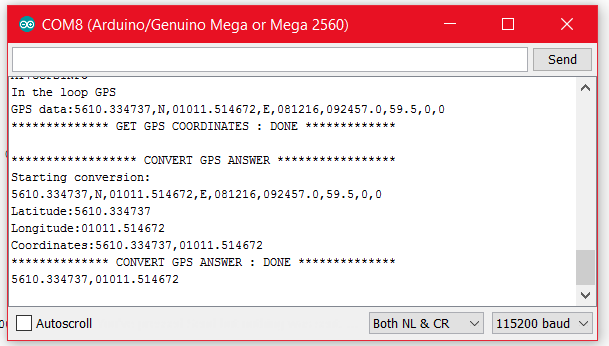
This test is used to see if the 3G shield is functioning properly and can be used in our project. There are two different tests for the 3G shield, in one hand for get the location and in the other hand to send the data to the Xively personal.

First we test the shield to get location of the supposed garbage bin. After gathering the location we have to transform the data in a more simple string because we don’t need date, altitude and time. After convert the data the Arduino microcontroller have the latitude and longitude to send it to the Xively personal cloud, but we will explain that after.

The Arduino steps while it get the location and convert the data:



*Figure 9: 3G shield unit test getting location*

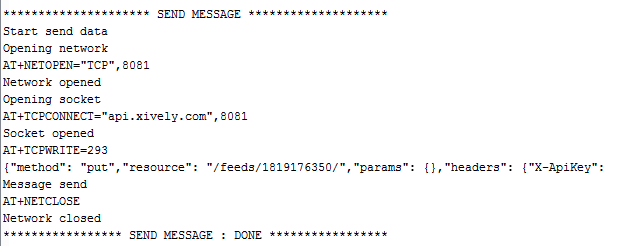
**

*Figure 10: 3G shield unit test convert data*

In the second step is test the connection between Arduino board (with the 3G shield) to Xively personal.

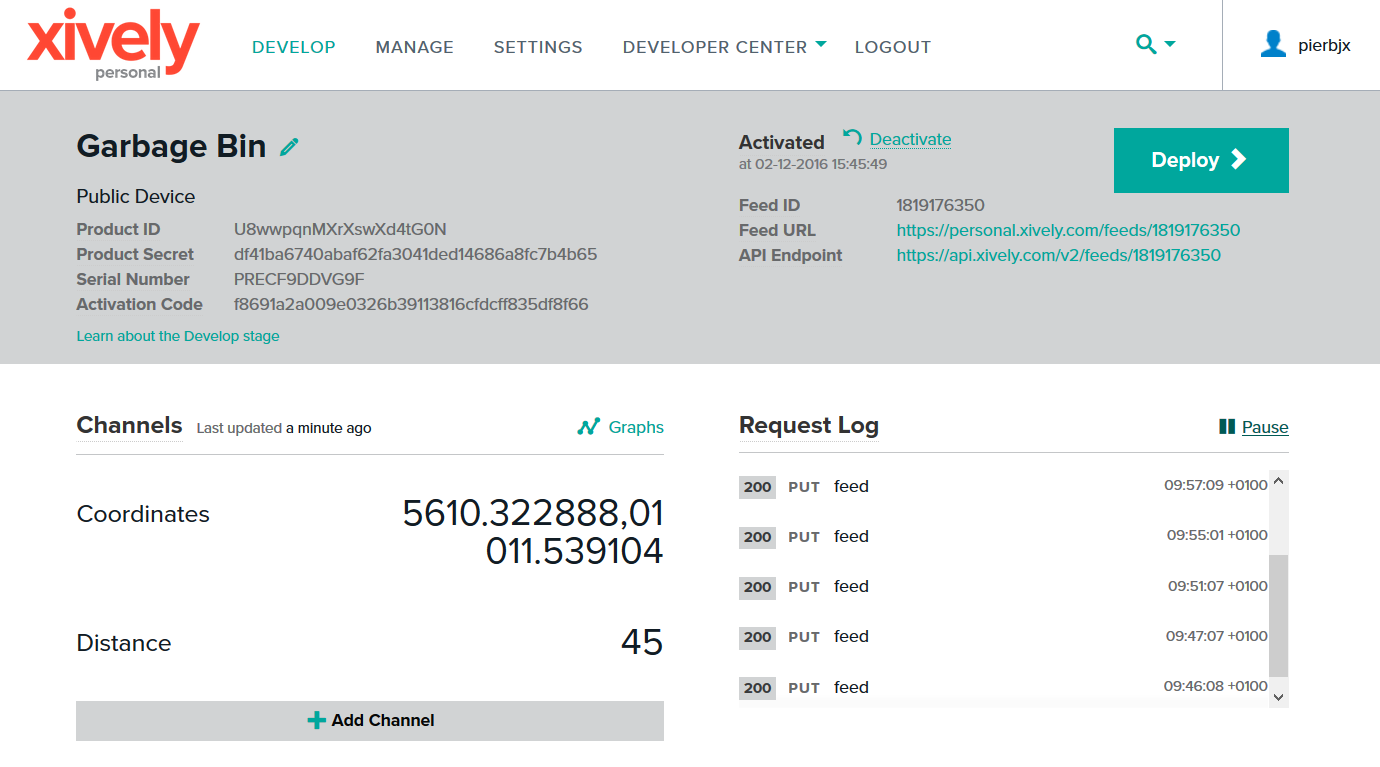
We fake two strings just for testing the connection, one for the distance and other one for the location. Distance fake is 45cm and the longitude and latitude are 5610.322888 for latitude and 01011.539104 for longitude.

First the Arduino board do the set up of the ports, after that it tries to send the two fake data to the Xively.



*Figure 11: 3G shield unit test getting data*

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



*Figure 12: 3G shield unit test Xively personal database*

Now we have all the results to pass to next step that it is to do the integration test.

### 5.2.2 Pre-requisites

* The 3G shield has been connected to Arduino
* Arduino has been programmed to properly

### 5.2.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| STEP | Procedure | Expected result | Status |
| 1 | Set up the 3G shield | Set up is done | OK |
| 2 | 3G gets the location | Location is got | OK |
| 3 | Convert the data | Data is converted | OK |
| 4 | Send the data to the Xively cloud | Data is sent | OK |

# 6. Integration tests

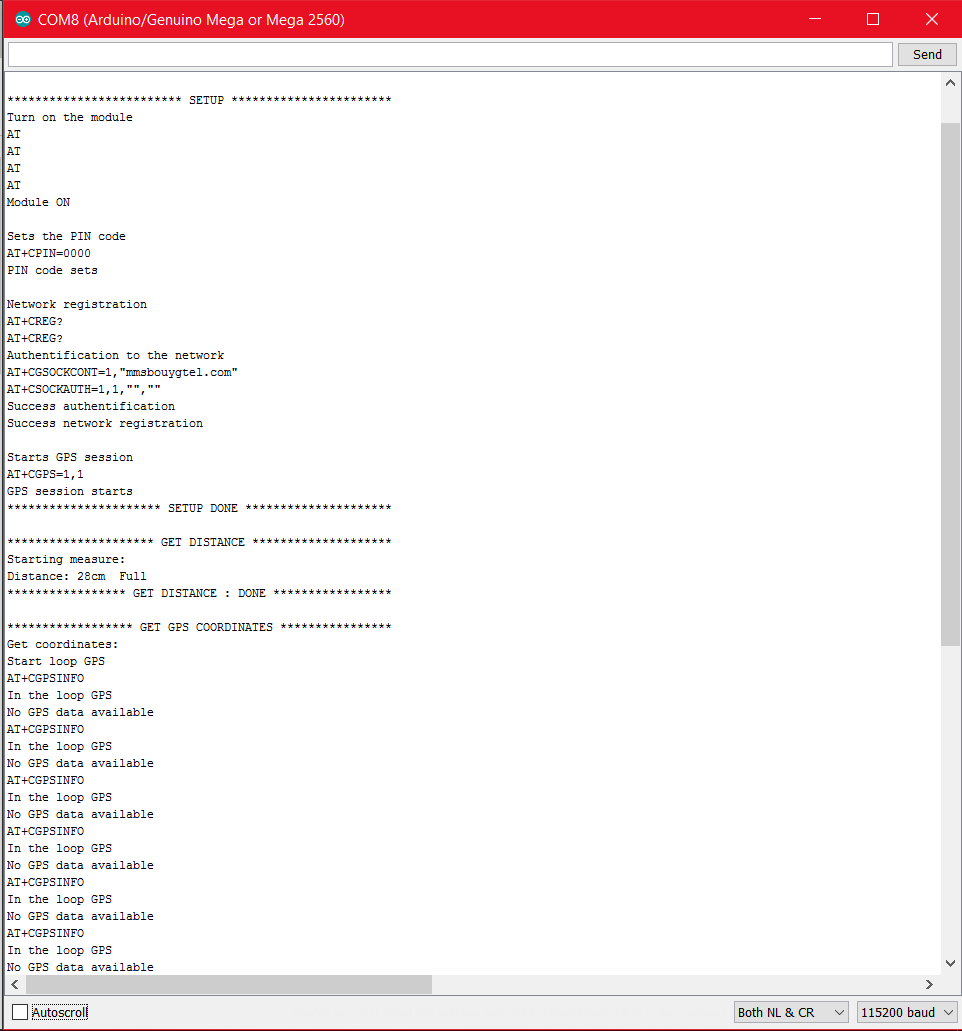
## 6.1 Definition

Integration testing is where each individual software modules is combined and tested as a group, it occurs after unit testing and before validation testing.

### 6.2 Description

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.

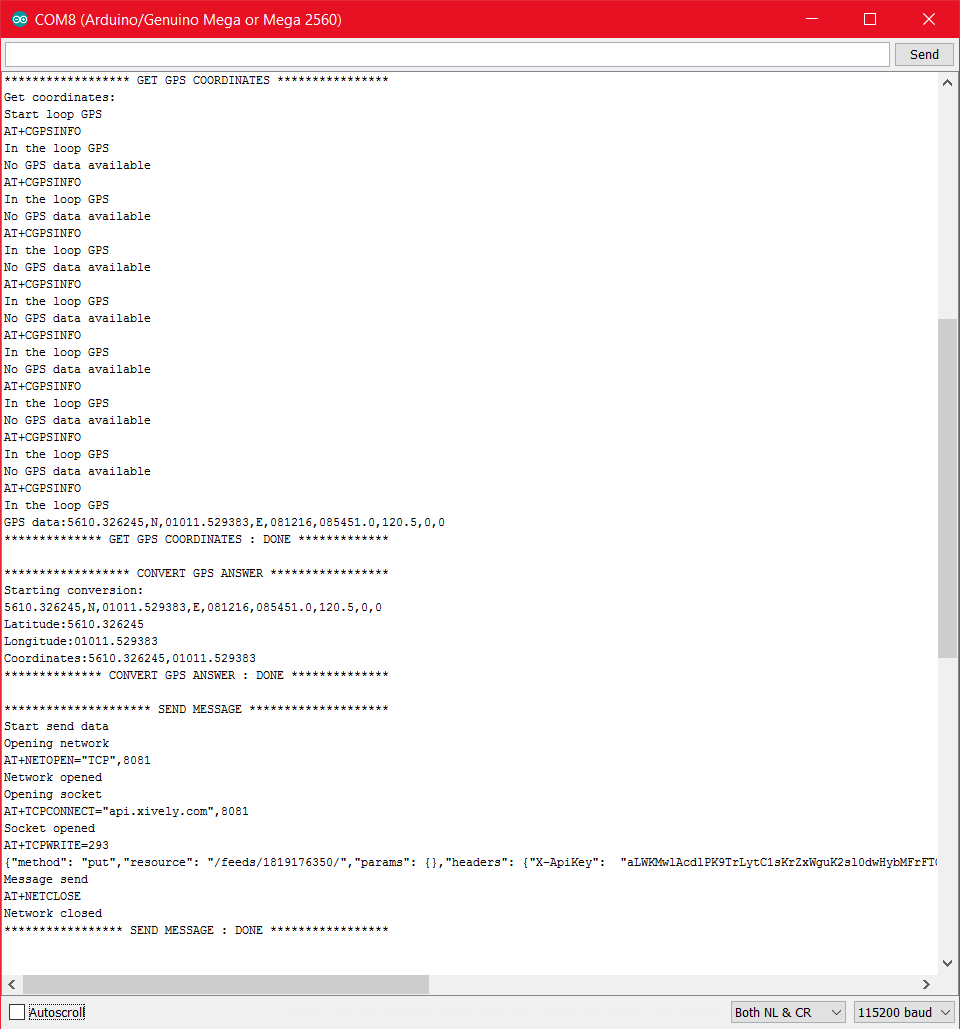
We wrote code to implement the final connection between entire system. So now the first step is turn on the entire system and set up all pins. After that the ultrasonic sensor is going to get the distance and the microcontroller stores it. Besides system is waiting for GPS answer but sometimes it can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



*Figure 13: First step of integration test*

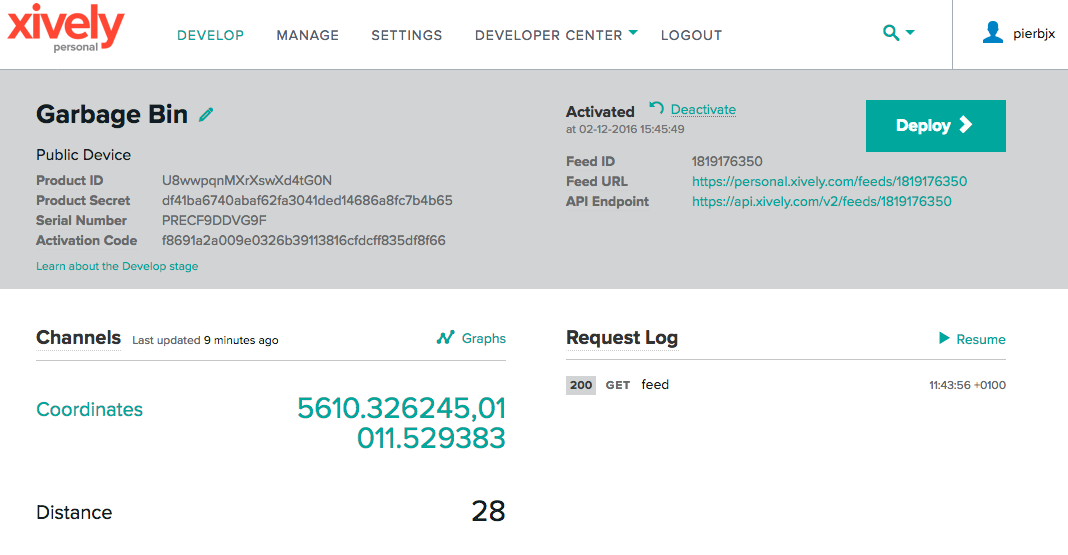
After waiting enough finally the microcontroller get the coordinate location and precede it to convert ass we already see it in the unit test part.

Once microcontroller has all the data it proceed to send it to the Xively cloud. Thenceforth the code has a delay of 10 minutes before it starts the loop again.



*Figure 14: Second step of integration test*

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



*Figure 15: Xively after the integration test*

### 6.2 Pre-requisites

* 3G shield has been connected
* Arduino has been programmed properly.

### 6.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Set up of the entire system | Set up is done | OK |
| 2 | Sensor gets de distance | Data is got | OK |
| 3 | 3G shield gets the location | Location is got | OK |
| 4 | Convert the data | Data is converted | OK |
| 5 | Send data to Xively cloud | Data is received in Xively cloud | OK |

# 7. Acceptance tests

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

## 7.1 Use case 1: authentication test

### 7.1.1 Description

This test 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 are no escape mechanism in the login part that could be used to modify our database through the login system.

### 7.1.2 Pre-requisites

* Smartphone application is installed
* Database with credentials has been created

### 7.1.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Insert wrong credentials  (username: horse, password: race) | Failure to authenticate | OK |
| 2 | Insert special characters (!?,;) | Failure to authenticate | OK |
| 3 | Insert correct credentials (admin, 1234) | Successful authentication | OK |

## 7.2 Use case 2: check status of garbage bins

### 7.2.1 Description

Test is to see connection between android phone and system.

### 7.2.2 Pre-requisites

* System inside garbage bin is online
* Sensor is reading data
* Data can be sent to android smartphone
* Smartphone application is installed

### 7.2.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Establish communication link | Data can be sent and received between interfaces | OK |
| 2 | Store data | Data is stored in Xively | OK |
| 3 | Fetch data | Data from Xively is fetched and parsed in Android | OK |
| 4 | Display status | Object is displayed in Android | OK |

## 7.3 Use case 3: check position of garbage bins

### 7.3.1 Description

Test is to see if garbage bins are shown correctly on a map inside smartphone application.

### 7.3.2 Pre-requisites

* Garbage bin objects are stored in android
* Application is installed
* Google maps API is working

### 7.3.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Open MapsActivity | Map loads | OK |
| 2 | Get GPS position from objects | LatLng object is created | OK |
| 3 | Put markers on the map | Markers are placed in garbage bin locations | OK |
| 4 | Show location of bins on the map | Garbage bins are shown on the map correctly | OK |

## 7.4 Use case 4: See details of garbage bin

### 7.4.1 Description

Find more information about selected garbage bin.

### 7.4.2 Pre-requisites

* Use case 3 must be working
* Smartphone application must be installed

### 7.4.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Select desired garbage bin from the map | Show detailed view of selected bin | OK |
| 2 | Select desired garbage bin from the list | Show detailed view of selected bin | OK |

## 7.5 Use case 5: Find the best garbage collection route

### 7.5.1 Description

Navigate from current position to desired garbage bin

### 7.5.2 Pre-requisites

* Smartphone application must be installed
* Google maps must be working
* You must have network connection on smartphone
* Current location must be determinable

### 7.5.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Select desired garbage bin from the map | Show detailed view of selected bin | OK |
| 2 | Choose to navigate to garbage bin | Route to garbage bin is shown | OK |

## 7.6 Use case 6: Change status of garbage bin

### 7.6.1 Description

Modify data inside smartphone to match status of garbage bin.

### 7.6.2 Pre-requisites

* Data about garbage bins must be stored inside smartphone
* Smartphone application must be installed
* Use case 4 must be working

### 7.6.3 Test procedure

|  |  |  |  |
| --- | --- | --- | --- |
| Step | procedure | Expected result | Status |
| 1 | Select desired garbage bin | Show detailed view of selected bin | OK |
| 2 | Choose operation to perform on garbage bin object | Data is modified correctly | NOK / database must also be affected for persistence |