Bibliography

Objectif Nux. App inventor : Premire application - orientation du smartphone, april 2012. URL `http://objnux.1s.fr/index.php?post/2012/04/29/App-Inventor-%3A-Premier-programme`. Accessed: 20.04.17.

Workshop 7 :

Embedded Systems Engineering

IoT Applications Prototyping

Team B:	SmartCart
Date:	22nd April 2017
Lecturer:	Dionysios Satikidis, MSc

Team members:

Project leader:	1644604	Timo Acquistapace
Developer:	1644612	Wojciech Lesnianski
Data-Analyst:	1644609	Markus Just
Documentation-Manager:	1644625	Simon Schneider

Deadline	24^{th} April, 2017

Contents

List of Abbreviations

Abstract

This present document describes the product SmartCart of the team IoT-Designers. The aim of the application is to offer a smarter way of shopping to the users. The application offers the possibility of easily marking items as added to the cart and the navigation through a list via gestures, in order to support and relieve the user during shopping. Therefore, the recognition of gestures is done via the built-in acceleration sensor in combination with the magnetic field sensor.

Instead of handling the shopping list with the help of pen and paper, this app should make this easier with gesture control. The main focus is on two use cases offered to the user. This allows the user to switch between the items in the shopping list with one hand and simple but unambiguous gestures, as well as to check off the already found items.

To recognize the chosen gestures, the acceleration sensor and the magnetic field sensor are used. The retrieved acceleration values are used to determine the movement that is made. The magnetic field sensor sensor serves to recognize the orientation of the smartphone and to be able to subtract out its influence out of the acceleration values.

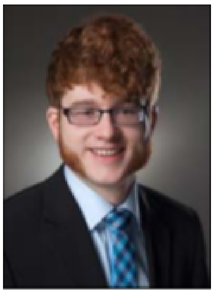
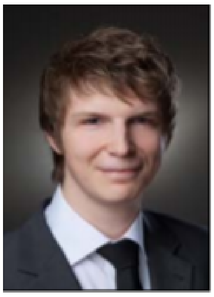
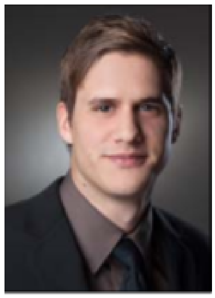
In order to derive gestures from the values obtained from the sensors, data were first acquired which reflect the respective gestures. Various measurements were carried out in order to assign the different sensor values. In order for the gestures to be carried out independently of the orientation of the mobile phone, various mathematical calculations are additionally required.

The end of this document describes the results of the app and how to use it.

Chapter 1

Smart Cart - A smarter Way of Shopping

In the following sections, the idea and implementation of SmartCart – an Android application that will simplify your shopping experience – will be exposed. The application was created in the course of the workshop "IoT Applications Prototyping" by the team IoT-Designers:

Timo Acquistapace	Wojciech Lesnianski	Markus Just	Simon Schneider
			
Project Leader	Developer	Data Analyst	Documentation Manager

1.1 Introduction

Since the idea of SmartCart evolved during the workshop, this section firstly introduces the initial product idea of SmartCart and the change of scope that project went through. Later on, the architecture and the system's context as well as the used state machine and its evolution are described. The following sections focus on the recognition of gestures. For this purpose, the mathematical basics that are necessary to recognize gestures are derived and the gestures that are used to control the system are introduced.

1.1.1 Initial Idea of SmartCart

The first concept of SmartCart was to offer its user the possibility to add items to a shopping list and to get this shopping list ordered automatically as the user enters a supermarket. The entered grocery is determined with the help of the Microsoft Here API. Based on the knowledge of the accessed shop and the ordering of its departments, the items that were previously added to the shopping list, should be ordered.

1.1.2 Change of Scope and final Idea of SmartCart

Even though the initial idea of SmartCart would have been a very helpful application to the user, it is strongly based on the collaboration with the operators of the supported supermarkets. This is especially true for the data acquisition regarding the offered products and the available departments of a supermarket. Therefore, the initial scope of the application was changed towards an application that is less dependent on master data.

The revised concept of SmartCart focusses more on the interaction of the user and the application. It omits the features of recognising a shop that is entered and ordering the list of shopping items according to the recognised shop. Instead, the application should offer the possibility of easily marking an item as added to the cart and of navigating through the list via gestures. The recognition of gestures is done via the built-in sensors for acceleration and the magnetic field sensor.

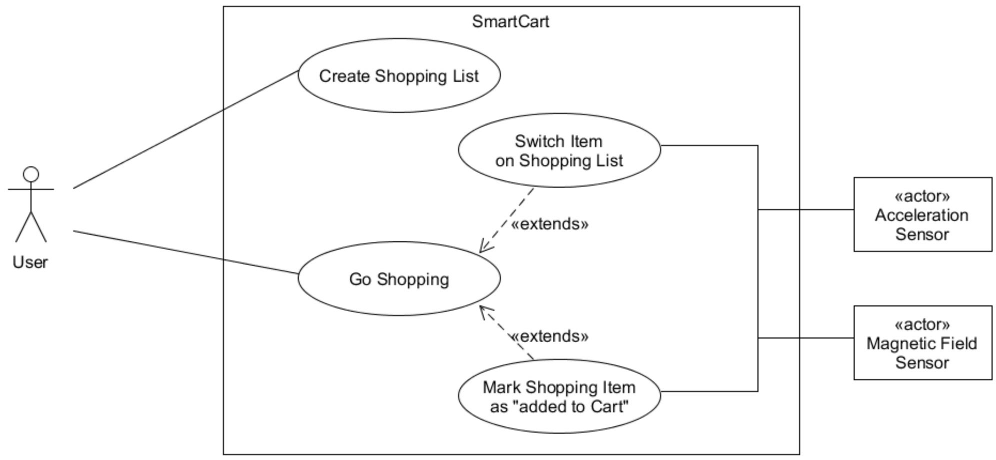


Figure 1.2.1: Overview of the System's Use Cases

Even the initial idea was discarded during the workshop for the mentioned reasons, the focus that is now put on the user interaction might nevertheless support the initial idea.

1.2 System Analysis

The upfront steps that were taken to create a shared understanding and to examine possible solutions in building the application are shortly presented in the current section.

1.2.1 Use Cases

SmartCart in its final scope addresses two main use cases: the use case of creating a shopping list and adding items to it as well as the process of going shopping itself (see 1.2.1). The use case "Go shopping" implements the main user interaction that consists of switching the next item to buy and of marking an item of the shopping list as 'added to cart'. This interaction takes place via gestures made by the user with its smartphone that are detected by the SmartCart application.

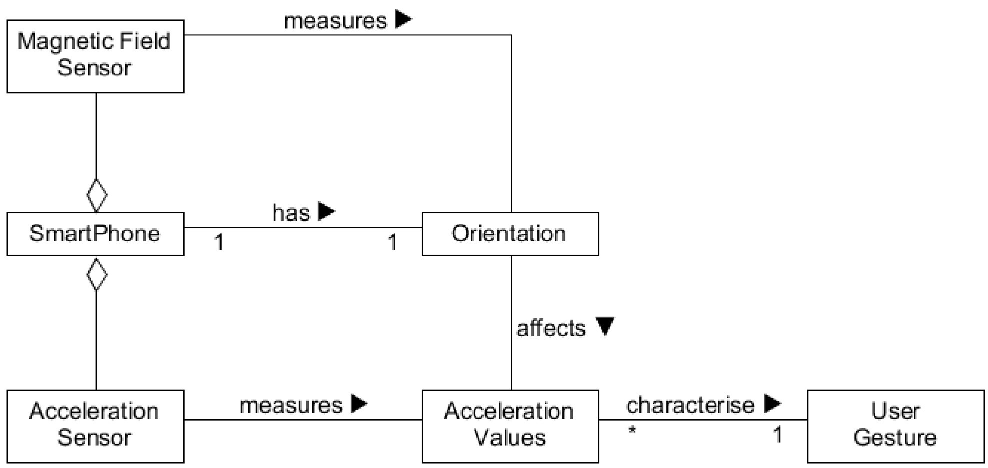


Figure 1.2.2: Data Model of the User Gestures to recognize

1.2.2 Relations between the captured Gestures and the used Sensors

Figure 1.3.1 shows a model of how the smartphone, the sensors and the gestures that should be recognized are related to each other. The acceleration sensor provides information about the smartphone's speed-up along its coordinate axes. In the most cases, these axes are not aligned with the standard x-y-z axes because of the smartphone's orientation. Since the orientation affects the measured acceleration values, it has to be taken into account when recognizing the user's gestures.

1.2.3 Application Context

The context of the application can be retrieved from figure 1.2.3. The inputs are the values of the accelerometer a_x, a_y and a_z as well as the angles $azimuth$, $pitch$ and $roll$ that determine the smartphones orientation. The current state and any other information of the application are visualized on the smartphone's display.

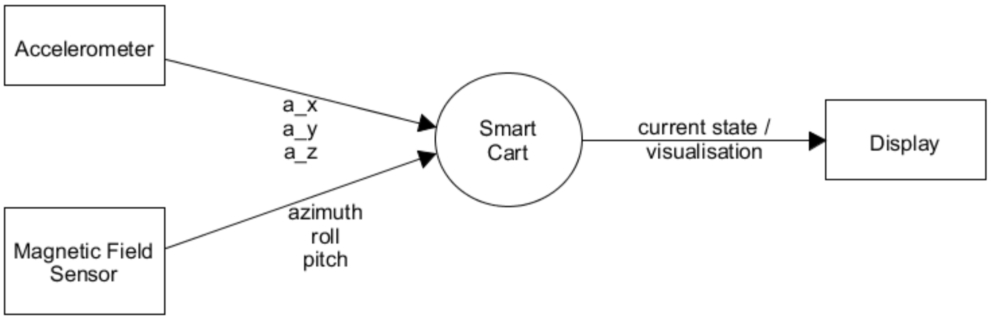


Figure 1.2.3: Context of the Application to develop

1.2.4 Finite State Machine

The app SmartCart is developed as a finite state machine. A finite state machine is defined by the fact that it can only have finite states. In order to describe such a machine, the so-called state diagrams are used. The state diagrams resulting from the analysis and evolved during the development cycle are presented in the following subsections.

First State Machine

This section describes the first design of the state machine. As can be seen in figure 1.2.4, this consists of only two states. The reason for only two states is that at this early development cylce neither the gestures nor the action triggered by the gestures could be accurately defined. Therefore, a data analysis was first carried out to ensure which gestures are possible at all and which make sense for the application.

Evolution of the State Machine

After the completion of the data analysis, the gestures and actions could be defined (see section **??**) **TODO: Kapitle vom Markus referenzieren**. Subsequently, the state machine could also be evolved and adapted. After starting the application, it is in the "Start" state. Afterwards, it is possible to start the purchase. If purchasing is started with "Start Shopping", the application is in the "Recognise Gesture " state. In this state, the application

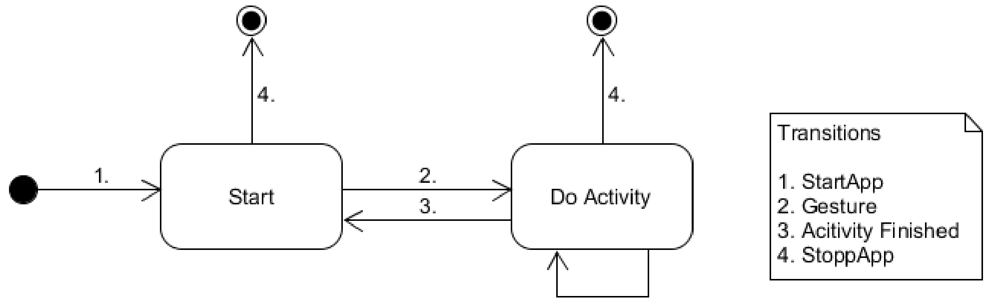


Figure 1.2.4: First Finite State Machine

waits for a gesture. For this purpose, two gestures were defined, each of lead to a certain action. If a clockwise gesture of the mobile phone is detected, this leads to the state “Check Item”. In this state, the selected item is checked off. Subsequently, the state switches back to the “Recognise Gesture” state. If a counter-clockwise gesture is detected, this results in the “Switch Item” state. In this state, the current item is swapped to the end of the shopping list. After the action has been completed, the system returns to the previous state. In general, the status “ Recognise Gesture ” will remain as long as no valid gesture is detected or the purchase is terminated. In addition, it is possible to terminate the app in all states. The complete state machine is shown in figure 1.2.5.

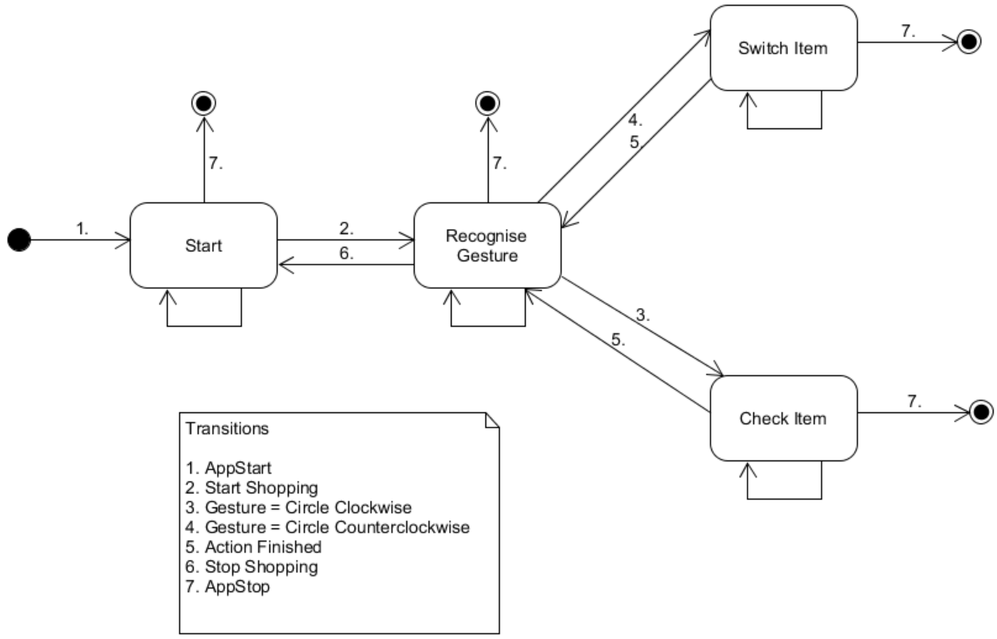


Figure 1.2.5: Evolutioned Finite State Machine

1.3 Mathematical Basics of Gesture Recognition

The recognition of gestures is based on measured acceleration values. These values depend, as it was already depicted in section 1.2.2, on the orientation of the smartphone. The mathematical relation of the measured acceleration values and the smartphone's orientation will be derived in this chapter. At first, the impact the smartphone being rotated along one of its axes is investigated in isolation. Afterwards, the results are combined and the final equation for each of the acceleration values is set up.

1.3.1 Terminology of the possible Rotations

There exist three different possible rotations that are measured by the magnetic field sensor:

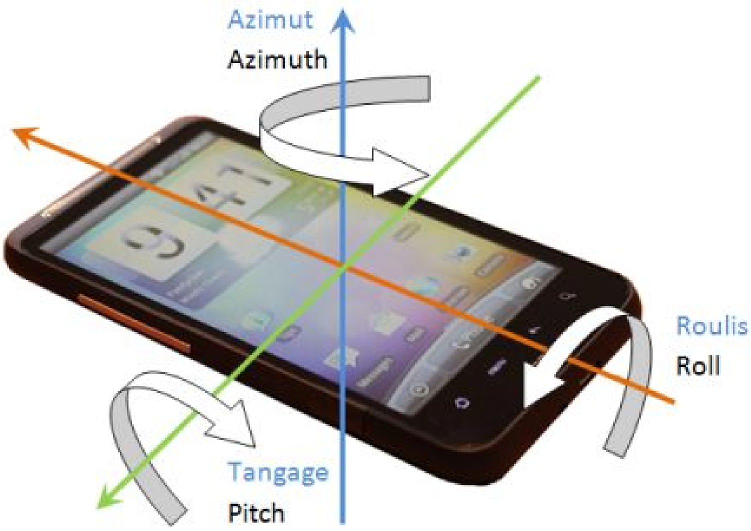


Figure 1.3.1: Possibilities of rotating a Smartphone, see Objectif Nux [2012]

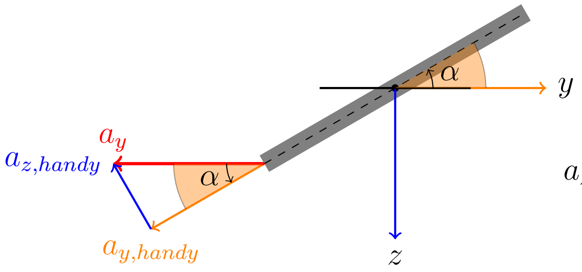
- **Pitch**
  The angle of a rotation around the x-axis is called pitch. In the following equations, α will be used to describe the value of pitch that is retrieved from the magnetic field sensor.

- **Roll**
  The angle of a rotation around the y-axis is called roll. In the following equations, β will be used to describe the value of roll that is retrieved from the magnetic field sensor.

- **Azimuth**
  The angle of a rotation around the z-axis is called azimuth. In the following equations, γ will be used to describe the value of azimuth that is retrieved from the magnetic field sensor.

1.3.2 Acceleration depending on Pitch

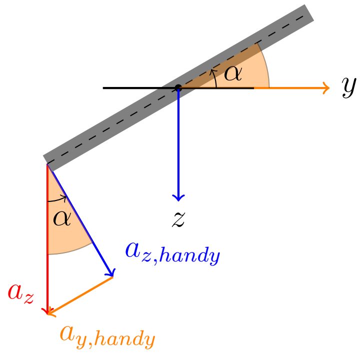
The current section investigates the effect of a rotation around the smartphone's x-axis on the measured acceleration values. Figures 1.3.2 and 1.3.3 show how accelerations in the direction of x and of z respectively might be decomposed into the different acceleration vectors that are parallel to the smartphone's axes. The magnitudes of these vectors are measured by the smartphone's acceleration sensor.



$$a_{z,handy} = \tan(\alpha) \cdot a_{y,handy} \tag{1.1}$$

$$a_y = \frac{a_{y,handy}}{\cos(\alpha)} \tag{1.2}$$

Figure 1.3.2: a_y depending on Pitch



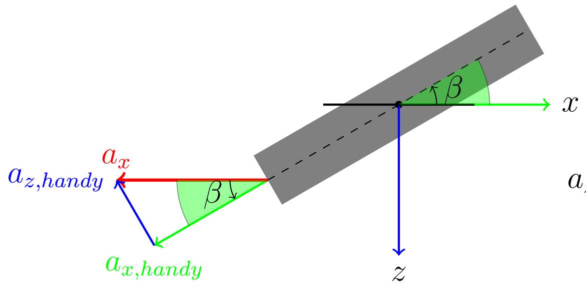
$$a_{y,handy} = \tan(\alpha) \cdot a_{z,handy} \tag{1.3}$$

$$a_z = \frac{a_{z,handy}}{\cos(\alpha)} \tag{1.4}$$

Figure 1.3.3: a_z depending on Pitch

1.3.3 Acceleration depending on Roll

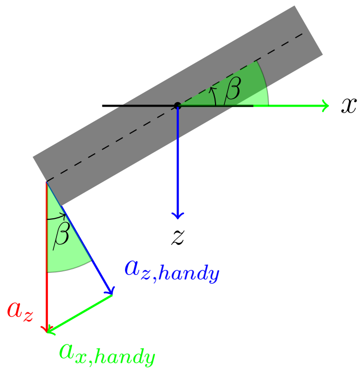
A rotation around a certain axis affects the measured acceleration values of the axes that are parallel to axis of rotation. Therefore, if the smartphone is rotated around its y-axis, the accelerations in the direction of x and z have to be investigated. The composition of the acceleration vectors can be retrieved from the figures 1.3.4 and 1.3.5.



$$a_{z,handy} = \tan(\beta) \cdot a_{x,handy} \tag{1.5}$$

$$a_x = \frac{a_{x,handy}}{\cos(\beta)} \tag{1.6}$$

Figure 1.3.4: a_x depending on Roll



$$a_{x,handy} = \tan(\beta) \cdot a_{z,handy} \tag{1.7}$$

$$a_z = \frac{a_{z,handy}}{\cos(\beta)} \tag{1.8}$$

Figure 1.3.5: a_z depending on Roll

1.3.4 Acceleration depending on Azimuth

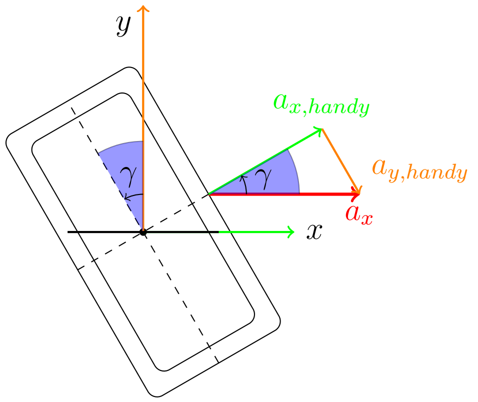
The last rotation that is examined is the one around the z-axis. This rotation affects the measured acceleration values in the direction of x and y. The vector-decomposition is depicted in the figures 1.3.6 and 1.3.7.

1.3.5 Final Equations for the Acceleration Values

After the acceleration values have been examined depending on each rotation value in isolation, the results are combined together in this section.

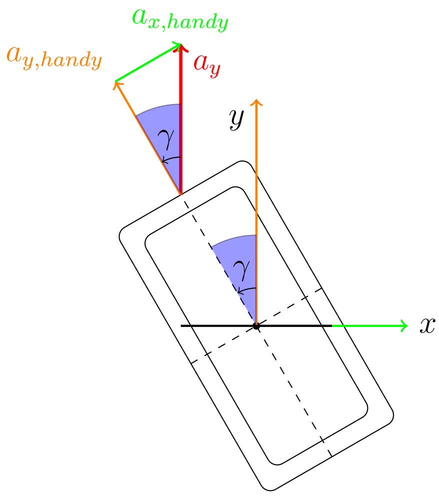
Final Equation: X-Acceleration

As it has been shown in the equations (1.6) and (1.10), the acceleration in the direction of x depends on the values of roll and azimuth. Combining these two equations leads to the following equation:



$$a_{y,handy} = \tan(\gamma) \cdot a_{x,handy} \tag{1.9}$$
$$a_x = \frac{a_{x,handy}}{\cos(\gamma)} \tag{1.10}$$

Figure 1.3.6: a_x depending on Azimuth



$$a_{x,handy} = \tan(\gamma) \cdot a_{y,handy} \tag{1.11}$$
$$a_y = \frac{a_{y,handy}}{\cos(\gamma)} \tag{1.12}$$

Figure 1.3.7: a_y depending on Azimuth

$$a_x = \frac{a_{x,handy}}{\cos(\beta) \cdot \cos(\gamma)} \tag{1.13}$$

The measured value of $a_{x,handy}$ is affected in two cases:

- if the handy is accelerated in the direction of z and the value of roll is not equal to 0
- if the handy is accelerated in the direction of y and the value of azimuth is not equal to 0

These affectations are defined by the equations (1.7) and (1.11). Taking these equations into account leads to the following, final equation:

$$a_x = \frac{a_{x,handy} \cdot (1 - \tan(\beta) \cdot a_{z,handy} - \tan(\gamma) \cdot a_{y,handy})}{\cos(\beta) \cdot \cos(\gamma)} \tag{1.14}$$

Final Equation: Y-Acceleration

The equations (1.2) and (1.12) show the dependency of an acceleration in the direction of y on the values of pitch and azimuth. If these equations are put together, the result is given by:

$$a_y = \frac{a_{y,handy}}{\cos(\alpha) \cdot \cos(\gamma)} \tag{1.15}$$

The measured value of $a_{y,handy}$ is affected in two cases:

- if the handy is accelerated in the direction of z and the value of pitch is not equal to 0
- if the handy is accelerated in the direction of x and the value of azimuth is not equal to 0

Therefore, the equations (1.3) and (1.9) have to be included in the calculations. This leads to the following, final equation:

$$a_y = \frac{a_{y,handy} \cdot (1 - \tan(\alpha) \cdot a_{z,handy} - \tan(\gamma) \cdot a_{x,handy})}{\cos(\alpha) \cdot \cos(\gamma)} \tag{1.16}$$

Final Equation: Z-Acceleration

The basic equation for the acceleration in the direction of z is retrieved by assembling the equations (1.4) and (1.8) that define the accelerations dependency on the values of roll and pitch respectively.

$$a_z = \frac{a_{z,handy}}{\cos(\alpha) \cdot \cos(\beta)} \tag{1.17}$$

The measured value of $a_{z,handy}$ is affected in two cases:

- if the handy is accelerated in the direction of y and the value of pitch is not equal to 0

- if the handy is accelerated in the direction of x and the value of roll is not equal to 0

If these cases (see equations (1.1) and (1.5)) are considered, the final equation is given by:

$$a_z = \frac{a_{z,handy} \cdot (1 - \tan(\alpha) \cdot a_{y,handy} - \tan(\beta) \cdot a_{x,handy})}{\cos(\alpha) \cdot \cos(\beta)} \tag{1.18}$$

1.4 Recognition of Gestures

The system recognizes two kind of gestures (see also 1.4.1):

- A circle that starts from the bottom and runs clockwise
- A circle that starts from the bottom and runs counter-clockwise

The first gesture is used to mark an item as "added to cart" while the second one serves to switch through the items on the shopping list.

(a) Add Item to Cart

(b) Switch Item

Figure 1.4.1: Gestures to Recognize

List of Tables

List of Figures

Bibliography

Objectif Nux. App inventor : Premire application - orientation du smartphone, april 2012. URL `http://objnux.1s.fr/index.php?post/2012/04/29/App-Inventor-%3A-Premier-programme`. Accessed: 20.04.17.