Implementing Business Processes using Internet of Things and Cloud Infrastructure

Master thesis

Jens Paulsen

Kongens Lyngby 2017

DTU Compute
Department of Applied Mathematics and Computer Science

Master of Science in Engineering
Computer Science and Engineering

Implementing Business Processes using Internet of Things and Cloud Infrastructure
Master thesis

Jens Paulsen
s040533@student.dtu.dk

Technical University of Denmark
Department of Applied Mathematics and Computer Science

DTU Compute
Richard Petersens Plads
Building 324
2800 Kgs. Lyngby
DENMARK

Tel. +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Abstract

This thesis explores the characteristics of Internet of Things (IoT) devices and principles in a proof of concept (poc) setup. This is explored due to the fact that the maturity and interest in developing systems using these principles is increasing rapidly at the moment. The poc setup will consist of an internet connected scale and a barcode scanner sending measurements to a cloud platform. We find that augmenting these devices with intermediary micro-controllers allows us to flexibly extend the limited network capabilities of the devices to allow cooperation with a cloud platform (Platform-as-a-Service) using the technologies and standards required.

The proof of concept explores the characteristics of IoT development, architecture, security and performance from a practical point of view using Amazon Web Services (AWS) technologies. We implement the poc system using these technologies, and find that in most ways the AWS platform provides us the aforementioned characteristics to a degree which lets developers build, verify and scale a PaaS system easily.

In the design and implementation we take into account resiliency against loss of internet connection and power in the different components of the system to minimize the need for technical assistance by the end-user or store personnel. Here we find that the most common scenarios are very well covered using capabilities already provided by operating systems, protocols and frameworks, but that more advanced failure modes require advanced logic to let the system diagnose the issue. Also, we find that if we cannot mitigate issues, the next best thing is to aid end-users and support personnel with good status and diagnostics capabilities.

Further we explore how the moving of a business process - in this scenario we will model a manual process using no digital hardware - from the analogue to the digital domain influences this work process and how it is performed after being digitized. We find several ways to improve the process by implementing fraud prevention using the scale as a verification device. Furthermore, we identify ways to enhance the shopping experience using smart phones during the shopping and a scale for checkout verification. We chose not to implement this in the poc setup, as it diverges too far from the hardware provided for the project and as it does not add sufficiently to the IoT-characteristic exploration that is already established in the present scenario.

Preface

This report is written as the Masters Thesis in my studies to become a Master of Science in Computer Science and Engineering at DTU. This report marks the final project of the studies, where I will show that I am able to handle the problem solving and analysis techniques thought at DTU, while at the same time applying the broad, yet specialized, knowledge about information technology that I have gained during these studies. Also, I will implement, at least part of the system, and test and evaluate the characteristics in relation to the problem statement and requirements identified at the start.

During my masters studies I have chosen the specialization of "Software Engineering", which has shaped the collection of courses I have taken towards courses specializing in software specification and modeling, although I have also had courses involving web services, distributed systems and security, to name a few. Hence, this report will probably have a strong emphasis towards analysis and modeling of the system, as this is of personal interest to me. Luckily my supervisor Stig Høgh has been good at keeping me focused on the practicalities of building the system and making it ready for a demonstration of the parts of the system that I have actually build, as the importance of the practical implementation is not to be underestimated in this project.

I hope that readers of this thesis will enjoy reading about the intricacies of electronic scales, Raspberry Pi's and cloud services, as much as I have enjoyed working with them.

Acknowledgments

I would like to thank my supervisor Stig Høgh for the support and time he has put into helping me finish this project. He has certainly spend more time on counseling me and giving me constructive feedback than I could have ever asked for.

Also, I would like to thank my beautiful wife for her never ending support and cheering, and for giving med all the time and breathing room that she could afford me during my thesis writing process. Without her support, this project would definitely not have been possible.

Contents

CHAPTER 1 Introduction

It is the aim of this thesis to explore the characteristics of Internet of Things (IoT) principles in a proof of concept (poc) setup. The poc setup will consist of an internet connected scale sending measurements to a cloud platform. The proof of concept will explore the characteristics of IoT development, architecture, security and performance from a practical point of view using specific implementation technologies which will be chosen during the analysis phase of the project. The design and implementation will also take into account resiliency against loss of internet connection and power in the different components of the system to minimize the need for technical assistance by the end-user.

Further we will explore how the moving a business process - in this scenario we will construct a manual process using no digital hardware - from the analogue to the digital domain influences this work process and how it is performed after being digitized. For example, it may be worth changing the steps in the work process to make it more efficient in a digital system compared to the way the work process is performed without the use of a computer.

CHAPTER 2 Problem Statement

In the field of Internet of Things devices, the computer industry and their customers in the business- and consumer sector face several challenges including but not limited to; security, resiliency, performance.

In this report we will use existing consumer and industry hardware to build a setup for exploring the characteristics and challenges faced in making these devices more flexible, secure, and updateable. The list of hardware chosen in the project is listed here:

- A Mettler Toledo Geo 23 scale with tcp/ip and serial communication interface.
- A handheld scanner (Datalogic TD1120-BK-90)
- Two Raspberry Pi computers (a model 1B and a model 3B)
- A LED strip containing eight RGB diodes for connecting to a Raspberry Pi (add-on module Blinkt!)
- A cellular modem for connecting these devices to the internet (Huawei AF22)

The devices listed above are used for exploring how we might design and build a system consisting of semi-independent devices which are used together to fulfill a task that a user might use these devices for. For this we choose a scenario of a self-service grocery shopping system, inspired by the ones used in the supermarket chain Føtex. We use this scenario to explore the problems stated above, and how to solve them. It is important to remember that we might as well have chosen another scenario used in other industries, and we will try to model every independent device in the system to be generic for usage in other systems, except of course for the business logic for handling the shopping itself.

CHAPTER 3 The Scenario

Normally when building an IT system you would begin by for formalizing "a vision of goals" for the system to end up fulfilling. These goals are quite abstract but lead, together with identification of the personas(and their use cases) who are going to use the system, to an identification of the formal requirements of the system to be build. These requirements then become specific features to be implemented and tested. We will not be pursuing the requirements engineering discipline quite as rigorously in this project

We have chosen to relax the requirement elicitation and verification process somewhat relative to the state-of-the-art requirements controls taught at DTU. We have done so due to several factors. First of all "we" are only one person, therefore the amount of coordination between team members is nonexistent, which lowers the requirement for formal control mechanisms somewhat. Second reason being that managing requirements rigorously adds overhead to the process of building the system, and being only one team member we would rather prioritize time for actually building the system than spending time managing the project for one person. Thirdly the focus of this project is to a high degree the practical usage of the hardware components of the system and their use in the system as a whole, and not the emphasis on requirement's management as such. Of course requirement management and system modeling are still an essential part of system building and we think that we have found a good balance of prioritizing the building of a practical demo system, with the requirements of system modeling and requirements engineering in relation to the scope and emphasis of the project. Now that this clarification and "apology" is out of the way, we can begin building the system...

For implementing the proof of concept (poc) we need a scenario describing a business process which can be implemented using the technologies chosen for this project. The key technologies are the tcp/ip (or alternatively serial-connection) connected scale and the cloud as the central data hub and data bus. By locking the implementation of the poc to a clearly specified business process we avoid changing the requirements of the implementation as we implement the different aspects of the functionality specified during the analysis of the problem description. Formulated in another way; by designing a scenario we lock the analysis to a predefined problem, and hopefully are forced to consider different aspects of the scenario which would perhaps not be considered if the problem statement were grabbed out of the air during the implementation process. Though we still reserve the right to iterate through the report and the problem statement several times during the project, at least we will be forced to consider why we change the scenario and what impact is has on the rest of the report.

3.1 The weighing functionality

For implementing the poc, we decided to use a "paying for groceries" scenario for implementation of the system. Hence, this will be the running example upon which all the documentation is modeled. It is important to note that the core concepts of the scenario can be generalized to other situations in which a weighting process takes place, data are collected and displayed from the cloud, and the scale is used as a measurement or verification method in this process. This might be weighing of goods as they are packed from a storage facility to be sent to a customer, where the weighting functions as a verification that the correct goods have been packed, before they are shipped to the customer. Another scenario might be the using the scale for verifying the number of counted votes in a ballot. Here the manually counted votes would be sorted in piles and counted with the extra step og weighing the votes to verify that the number of counted votes correspond to the actual weight of the voting paper. This has actually been proved to be a fairly accurate method[1]. The scenario will also function as a use case for a customer persona who is going to use the system. The reason we can simplify the specification process here, is that the poc is build around the hardware components (scale, handscanner etc.), which are only used by the customer. Also, all customers must (probably, if we decide so) follow the same process when buying groceries in the self-service solution, except maybe differentiated by the payment method they use. So basically we only have one use case for the customers who use the IoT-devices, and the other personas, such as the supermarket clerk, the system administrator, the system developer etc., are not taken into account in the scenario or the demo-system (the poc) that we are going to build. The scenario will be build around a business process description, which will also act as use case for the actor/persona, which is the customer.

3.2 The analog scenario

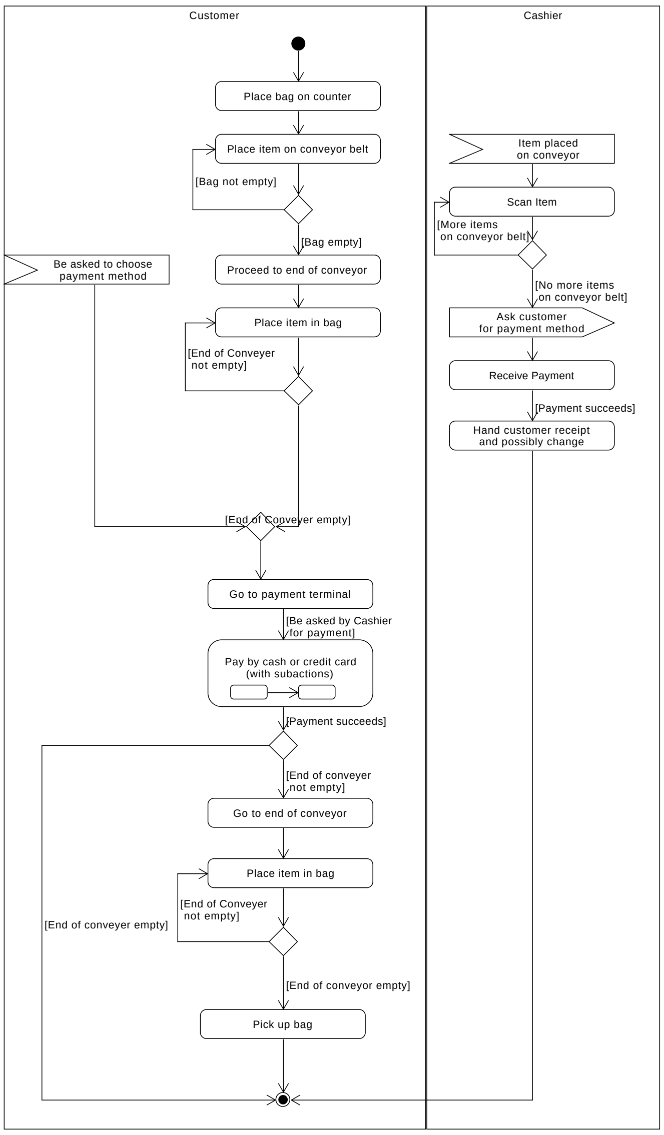
In the classical (or analog, manual etc.) process of paying for groceries, the user arrives at the cash register and places her groceries on the conveyor band, allowing the supermarket employee to scan the groceries and subsequently receive payment from her. While the first customer is packing her bags, the (checkout lane) escalator band can process another customers groceries, by sending the groceries to a second compartment in the excavator band. Thus, the band alternates between the two compartments, keeping the supermarket employee busy scanning groceries, and accepting payments while the flow of customers through the system is maximized. In Figure 3.1 you see two swim lanes, one representing the customer, and the other representing the employee at the escalator band. The customer keeps putting items on the belt until the bag is empty, then proceeds to the end of the conveyor to pack her items. At any time during the packing, the employee may choose to ask the customer for payment method. The customer approaches the payment terminal, pays, and (hopefully) the payment succeeds. If the customer has not finished packing she will do so after paying. The employee keeps scanning groceries until no more are left, asks the customer for payment, and hands her the receipt (and possibly change), before proceeding to the next customer. It must be noted that we could have chosen to make the activity diagram more complex by for example expanding on the payment sub-process, which is just shown as an abstract element here. Or we could have expanded on the receive payment step for the cashier and added more sub-steps depending on the payment method chosen by the customer. Ultimately modeling business processes is always a balance of keeping the diagram simple, while still encompassing the important steps and activities necessary for capturing the important aspects of the system.

3.3 Description of the digital real world scenario

In this section of the report we will describe the scenarios on which the poc is implemented. For this report the scenario will describe a business process wherein an end customer must scan and pay for groceries bought in a supermarket (inspired by the computerized self service queue implemented in Føtex supermarkets.) The scenario will be described using both UML compliant diagrams and text descriptions. The text below describes the steps shown in Figure 3.2 .

A customer enters the payment area with the goods she wishes to purchase. She chooses a payment counter which is not already in use, or closed for maintenance (maybe based on a color coded lamp attached to the top of the payment counter. She places her groceries bag on the side of the counter, named the "in" counter. She presses the “start” button on the display of the counter. The counter asks her if she has brought her own groceries bag or not. After answering the question the customer either places her own bag on the holder on the the side named the "out" side of the counter, or places a plastic bag from the counter instead. The terminal asks the customer to take one item from the "in" side store basket, scan it, and place it in the "out" bag. This process continues until all groceries have been scanned. The customer is not allowed to scan several goods at once, but must take it up, scan it, and place the good on in the bag on the other side of the scanner. When all goods have been scanned, the customer presses the pay button on the screen, and chooses payment method. After payment has completed the customer can pick up the "out" bag and leave the store. If during the scanning process an item where no barcode is present is found, the customer can find the item on a search list. This may be relevant when for example a vegetable has been picket up that does not contain a barcode. Vegetables which are priced based on their weight are normally weighted in the store and given a custom barcode presenting their weight and price. If the scanner cannot read a barcode during scanning then the customer can manually enter the code if they can read it. If they cannot read it, the customer may always call the employee who is assigned to the self-service checkout.

When the customer has scanned her groceries, and proceeds to press the "proceed to payment" button, the sef-service terminal may trigger a "random" check of that requires the employee who is overseeing the self-service terminals to manually approve the correctness of the scanned groceries vs. what the customer has actually placed in her bag. The reason that this is listed as "random" in quotation marks is that it may be semi-random based on pattern matching or machine learning which based on a pattern or a confidence heuristic may trigger a call for an employee checkup. We



Customer
Cashier
Place bag on counter
Place item on conveyor belt
[Bag not empty]
[Bag empty]
Be asked to choose payment method
Proceed to end of conveyor
Place item in bag
[End of Conveyer not empty]
[End of Conveyer empty]
Go to payment terminal
[Be asked by Cashier for payment]
Pay by cash or credit card (with subactions)
[Payment succeeds]
[End of conveyer not empty]
Go to end of conveyor
Place item in bag
[End of Conveyer not empty]
[End of conveyer empty]
[End of conveyor empty]
Pick up bag
Item placed on conveyor
Scan Item
[More items on conveyor belt]
[No more items on conveyor belt]
Ask customer for payment method
Receive Payment
[Payment succeeds]
Hand customer receipt and possibly change

Figure 3.1: Activity Diagram for the manual process.

do not know if this pattern matching or machine learning is actually implemented in current systems, and we will therefore defer this topic to the succeeding chapters of analysis and implementation.

In the the real world, several other factors have to be taken into consideration, such as if the payment does not succeed, the customer must cancel the process or possibly call an employee for help. We have chosen not to model these in the process, but we do realize that in a real system in production, these exceptions would have to be taken care of as well.

3.4 The scenario implemented here

In the implementation of this poc we will focus on a smaller subset of the larger scenario described above. This will be elaborated upon in the upcoming chapters. Due to the already given hardware for the project, the implementation of the scenario will comprise a touchscreen for user interaction, a tcp/ip or serial-cable connected scale for measuring the weight of the groceries, and a barcode scanner for scanning the groceries. All of the elements in the setup described above are connected to a common infrastructure in the cloud, (e.g. based on a Google or Amazon etc. cloud infrastructure) for internet of things. This cloud infrastructure will contain all of the business logic for the software functionality required by the scenario.

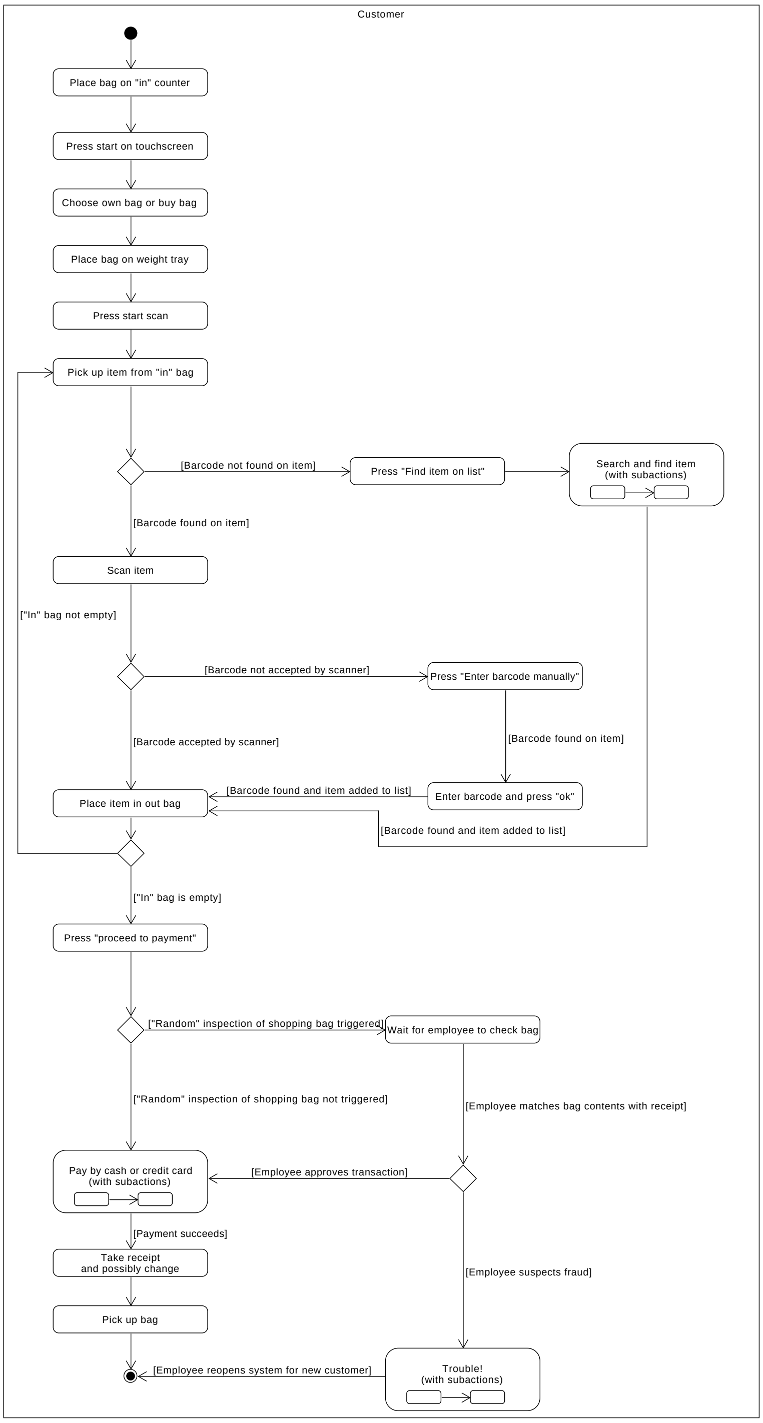

Customer
Place bag on "in" counter
Press start on touchscreen
Choose own bag or buy bag
Place bag on weight tray
Press start scan
Pick up item from "in" bag
[Barcode not found on item]
Press "Find item on list"
Search and find item (with subactions)
[Barcode found on item]
Scan item
["In" bag not empty]
[Barcode not accepted by scanner]
Press "Enter barcode manually"
[Barcode found on item]
[Barcode accepted by scanner]
[Barcode found and item added to list]
Enter barcode and press "ok"
Place item in out bag
[Barcode found and item added to list]
["In" bag is empty]
Press "proceed to payment"
["Random" inspection of shopping bag triggered]
Wait for employee to check bag
["Random" inspection of shopping bag not triggered]
[Employee matches bag contents with receipt]
Pay by cash or credit card (with subactions)
[Employee approves transaction]
[Payment succeeds]
Take receipt and possibly change
[Employee suspects fraud]
Pick up bag
[Employee reopens system for new customer]
Trouble! (with subactions)

Figure 3.2: The digitized process.

CHAPTER 4

Analysis

In the analysis chapter we try to convert the business process diagrams into requirements for the system to be build. These requirements should be as technology independents as possible, as to not lock the developers of the system into specific technologies. The technologies will be chosen later, based on an overall assessment of the requirements, the available developers, maintainability of code etc. In reality most of the hardware devices for the system were given beforehand, but the software and the specific functionalities of the software have not been given yet, so the analysis process is still warranted. We might also identify hardware requirements which the given hardware does not possess. If this should happen we will note it, and try to find a workaround. New requirements are added iteratively while the report is being written and the system is being build in parallel. Features of the implemented system are linked to the requirements as to be verified that they are fulfilled. Some requirements will be deemed outside the scope of the poc, or be prioritized low and therefore not implemented in the final poc, given limits in development time.

4.1 Functional Requirements

This section describes the functional requirements, or requirement attributes (RA) identified for the poc system used in the scenario described in the previous section. The functional requirements are elicited based on the flow in the business process model.

RA1 The customer must be able to specifically start her own session on the system, as to not confuse the customer if she approaches a terminal where the preceding customer left the system in the middle of a transaction.

RA2 The customer must be able to choose if she wishes to buy a bag for the groceries, or if she has brought her own.

RA3 The system must be able to verify that the customer actually places the groceries which she scans on the scale after scanning them.

RA4 The customer must be able to continuously and easily scan groceries, until she indicates to be done

RA7 The customer must always have a status of scanned groceries and summarized cost on screen while the scanning is commencing

RA5 The customer must be able to choose between several payment methods.

RA6 The system must be easily extendable with new payment methods.

RA8 The system must be able to ask the customer to put back a grocery item on the scale, it the customer removes the item while the transaction has not finished.

RA9 The system must be able to determine if the customer replaces one scanned item with another item during the scanning of all groceries.

RA10

4.2 Non-functional Requirements

In this section I will describe the non-functional requirements (or quality attributes, QA) [16a] identified for the proof of concept setup. Non-functional requirements encompass "quality attributes", which are not specific functionalities og the desired system, as much as they are qualities inherent in the whole system as such. For example a non-functional requirement that the system is "secure", is not the same as the functional requirement that the system uses https for web communication. While there are many possible non-functional requirements to add to a system, for this specific system only the ones deemed relevant are mentioned below. Often many quality attributes are important in a system, but they may very well conflict with each other given limited development resources. For example a quality requirement of maintainability may be made more difficult by other requirements(security, speed) if they add more complexity to the system. Therefore we must often content with the fact that all quality attributes are somewhat important but that some quality attributes are prioritized higher or prioritized in favor of other quality attributes in the specific context of the given system under development. Even though a specific non-functional requirement is discussed below, it may not necessarily need to be specifically implemented in the qualities of the system, if it is already sufficiently covered by an existing system or third party framework which the poc integrates into. For example security is important to the whole system, but if the cloud infrastructure already implements sufficient security for our system, we may only need to assert that the requirement has already been fulfilled by the cloud provider.

4.2.1 Response Time

[QA1] When the scale sends measurements to the cloud, these measurements must be sent to business logic og the system, but also specifically to the touch display on the console. We have chosen to prioritize optimization for low latency for sending these measurements to the touch screen as for example a latency of several seconds could possibly decrease the user experience, but worst case make the system unusable if the users do not recognize that the scale and screen are out of sync. The system uses soft-realtime requirements [16b], as data send between components of the system never make the system unusable as such, but rather make the user experience worse until the latency makes waiting time intolerable to the user.

4.2.2 Security Requirements

[QA2] In the security requirements for the project, all the devices connecting to each other via the cloud infrastructure, are communicating over the internet. Here we must ensure that all devices can authenticate, and ensure integrity and confidentiality [17b] of the data sent by these communication channels.

4.2.3 Privacy Requirements

[QA3] While in the security section above, the requirements of the system to ensure security in the communication between system elements, the users using the system must also place a level of trust in the owners of the system; that the system owners do not use the information that the users directly or indirectly give to them, are given to third parties or used in ways which the users do not approve of. In a shopping scenario this might be data mining the specific shopping habits of individual users to send the targeted advertisements or selling the information gathered to third parties. These privacy requirements may be fulfilled by having the supermarket adhering to already established compliance laws, such as EU privacy laws. Also, if using one or more cloud provider(s) for the system or parts of the system, these providers must also adhere to national and EU laws.

4.2.4 Resiliency and Robustness Requirements

This project will have emphasis specifically on resiliency requirements and robustness requirements. Resilience is the ability of the system to cope and recover from general errors (e.g. hardware, software and communication errors) while being in production, and robustness is the ability to handle unexpected erroneous input and/or program errors internally in the system, as to not having to recover at all. The specific points which have been identified as relevant to focus on in the system are elaborated in the following subsections.

Data connection loss

[QA5] Upon loss of data connection between the components of the system, depending on the communication channel which looses the connectivity, we have to consider the severity and countermeasures available to us to mitigate the loss, and/or reestablish connection automatically when the channel is available again.

Power loss

[QA6] During a power loss the component of the system is completely unavailable. Here the user must perhaps be notified of the failing component, and the component should be able to automatically reestablish connection to the cloud when power is reestablished.

Software error

[QA7] If a software error should occur in any component of the setup, depending on the specific service in the software in which the error occurs, the component may be able to correct the error. If it is a service of the business specific logic, a watchdog service may recover the failing software component, whereas if it is the kernel that malfunctions, the possibilities for correction may be beyond the scope of this project.

Hardware error

[QA4] Similarly to the software error, a hardware error may be recoverable under certain circumstances, but this will have to be supported by the specific hardware chosen in the setup, possibly aided by support of the operating system running on the hardware.

4.3 Business process modeling

When analyzing the business process for which the poc system is to be used, we must also consider possible ways om optimize the flow of the business process. The optimization of the business may both be relevant because the technologies used to implement the system let us improve further on the established "old fashioned" way of paying for groceries, but it may also be relevant to do because limitations in the technology make it easier to change the business process to accommodate the technology, than the other way around.

Scanning groceries after they have been placed on the scale

We have considered modeling the system so that the customer can place the item on the scale, and then scan the item while it is placed on the scale, to save time and let the scale settle while scanning. While this may save time and be a very practical solution, we have chosen not to do so, as for a groceries shopping system the scanner might be firmly mounted and not movable, as to make it easier for the customer to move the item in front of the scanner with both hands instead of having to maneuver the scanner with an attached power/data cord above the item. Also the customer would have to remember to always place items with the barcode on top. In other use cases, such as packaging similar boxed in a shipping facility it may be very relevant to scan the item on the scale, as the items for shipping may be very similar (all are boxes), while at the same time having highly trained operators working under time pressure. In a shopping scenario the customer may be in a hurry, but at the same time the system must be easy to use for all customers, both those with and without routine in using the system.

Weighing groceries which are priced based on weight during checkout

Most supermarkets sell some fruit and vegetables as priced by their weight. Therefor the customer must weight them on a scale in the groceries section and print a custom label which describes the type of vegetable, and the weight. This procedure could be moved to the self-service terminal, where the customer would weight the vegetables anyway, when going through the system. It would save the customer the hassle of weighting the vegetables at the groceries section but give several disadvantages tot the customer and store as well. Firstly the store might be interested in the customer actually weighting the vegetables and printing the custom label, because this allows the self service system to check that weight on the label is correct at checkout. The customer could still cheat by setting the first scale to a cheaper vegetable. But this could possibly be checked in the random inspections by the staff at the self servce system. Also the customer may actually have an interest in weighing the vegetables in the vegetables section, as it is difficult to add or remove vegetables when you are at the self service system. If the customer only needs a certain amount of the vegetable, she needs to be able to check when she is at the groceries, as she is then able adjust the amount of vegetables in the bag.

Considering things not placeable on the scale

The system must also be able to handle objects to large to place on the scale, such as beer crates, televisions sets etc. If the scanner is mounted and unmoveable, the customer must be able to enter the barcode manually as a fallback method, but also if the object is too large or too heavy to place on the scale, the system must be able to exempt the item from weighting.

Mobile phone as scanner during shopping - Scale as end verification

If could also be investigated whether the customers could use their own smartphones as barcode scanners, and scan the groceries while they put items into their baskets. This concept is already being tested by the Danish supermarket giant Coop, and is known as "Bip og Betal"[17a] [KÜH00]. When the customers leave the store they pay for the groceries packed and scanned in their bag. This is then secured by random inspections by employees when customers leave the store. Here a scale could possibly be a means of verification that the sum of the weight og the scanned items in the bag match the values calculated by the system. While this business process helps remove customers from the checkout lines and may allow the supermarket to have fewer employees at the checkouts, we have chosen not to pursue this idea further. But we do recognize that this may actually bring value to both the customer(ease of shopping) and the supermarket(allows to reduce personnel and space reserved for checkout).

Disruption by RFID profileration

At last the profileration of RFID tags would allow automatic scanning of the groceries, with the scale as a verification of the summed weight of the items. This would be an easy solution for the customers, if at the same time their customer loyalty card or their phone could be asked for payment. This would change the business process just like the Bip og Betal solution mentioned above. There are still problems with high prices for RFID scanners, the price of the tags themselves and difficulties in scanning tags under different conditions [MdM17], but the technology is being used in retail [Cat14], and maybe it will one day be more practical for stores to implement this business process. We think the technology has potential, but for this project this concept is to exotic that we will investigate it further.

4.4 Requirements Delimitation

We choose to delimit this project to work with the technologies relating to modeling and implementing IoT and distributed systems using state-of-the-art technologies approaches. Therefore we can initially ignore some of the above stated requirements. Of the requirements elicited above, the bullets regarding payment [RA5] and [RA6] can be ignored, as payment will not be build upon further in the project. Also, privacy requirements regarding dataprocessing internally in the system will be ignored [QA3], as they are prioritized very low in this poc system, but also often handled politically (courts fine companies which break privacy laws), besides of technologically (user controlled encryption, and one-way hashing of sensitive data.).

CHAPTER 5 Implementation

In this chapter we describe the software system which has been implemented based on the requirements elicited in the previous chapter. The requirements have only been described superficially, and can now be further elaborated upon for the most important ones. When(if) the requirements have been elaborated upon to a degree which is deemed sufficient for the development process to begin, we will start by making the technology choices for the different parts of the system. It is important to remember that the development of the system for this project happens iteratively [16c], where a rapid prototype testing the feasibility of high risk system features is tried out first, and then developed upon by adding features according to their priority of importance. This is the done in sprints, where after each sprint, the developer(s) can demo the system to the external customers, or the internal users of the system and correct misunderstandings introduced under the requirement elicitation, or customers who realize that they actually want something completely different than they thought that they asked for[06].

5.1 Business process delimitation

As mentioned in the previous chapter, we will only focus on a smaller subset of the business process in the poc implementation. This subset is shown in 5.1, and the individual steps are the same as in the fully modeled process, except that we have simplified the whole process for the poc system.

5.1.1 Fraud Prevention

The system must handle fraud prevention, by checking that customers adhere to the trust that is placed in them by the supermarket, when they are using the self-service system. This check can be implemented and combined in several possible ways.

Scan and Place Item

When the customer scans an item, she must immediately place the item scanned on the scale before she can scan another item. This will help ensure that the number of items scanned matches the number of items the customer actually must buy. This helps the customer to not make mistakes in the procedure.

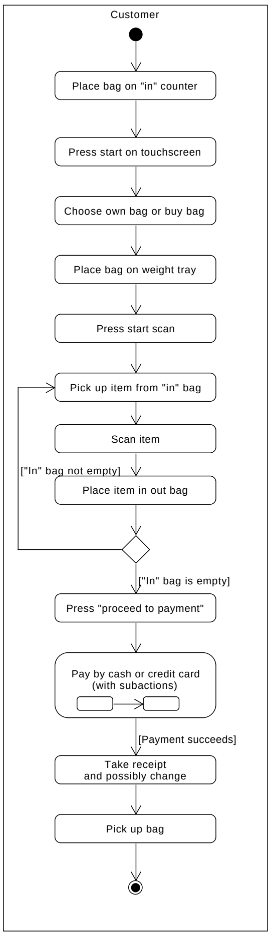

Customer
Place bag on "in" counter
Press start on touchscreen
Choose own bag or buy bag
Place bag on weight tray
Press start scan
Pick up item from "in" bag
Scan item
["In" bag not empty]
Place item in out bag
["In" bag is empty]
Press "proceed to payment"
Pay by cash or credit card (with subactions)
[Payment succeeds]
Take receipt and possibly change
Pick up bag

Figure 5.1: The digitized process for implementation.

Checking the Weight of an Item against Database

When an item is placed on the scale after being scanned, the system verifies that the weight of the item scanned, is approximately the same as the weight listed in the database that contains all items. This helps prevent fraud if a customer repeatedly scans an inexpensive item, and places more expensive items on the scale after scanning. This type of fraud prevention will not stop a customer from scanning for example one liter of cheap milk, and then placing a more expensive yogurt on the scale after scanning, but it would still prevent some forms of fraud. Also this anti-fraud system requires the supermarket to actually have the weights of all their groceries entered in the database. This might incur significant cost on the supermarket, and perhaps not be implemented in the real system.

Random Inspections by Staff

Even the customer should try to fool the system by placing more expensive items of the same weight on the scale after scanning, the system might still encourage customers not to do this, by blocking the payment step in the business process and require manual approval by the supervising staff at the self-service checkouts. This blockage of the payment would require staff to manually approve the scanned items, and let payment proceed. These checks could happen on random, but could also be semi-random, by for example happening more often if a customer triggers certain patterns when scanning items. An example of such a pattern might be: (yogurt:1050g)->(milk:1050g)->(milk:1050g)->(milk:1010g)->(milk:1010g), or some other pattern, wherein the median weights of two products with similar weights but different prices are interchanged.

Checking that no items are removed during scanning

The last method to verify that customers are following the correct scanning procedure, would be to ensure that no items are removed from the scale after they have been scanned. This would both allow the staff to inspect all the items on the scale during a random inspection, but also ensure that a customer could not exchange cheaper items with more expensive ones after scanning has taken place, unless they weight exactly the same (within some narrow margin).

5.1.2 Serverless architectures

For the overall server architecture we have considered a standard approach using a server instance installed on a cloud server versus a new trend in system building; the serverless[Boy00c] architecture. A serverless architecture allows us to forgo the administration of servers in the cloud. Instead we can focus on writing individual software functions for execution by a cloud computer whose underlying OS and application server environment complexity is abstracted away to us. A service or program which is developed by us, is then comprised of many small services (microservices [Boy00b]) which are orchestrated together to form the whole program/service to the end-user/customer. The services are not seen by the end user, but only the whole system as a blackbox. The serverless architecture allows us to outsource the server, OS, and applications server environments, and instead only focus on our business logic. Naturally we must place trust in the provider who manages the servers and software stack, but as was the case with hosted/cloud computing, some use cases make serverless/hosted systems best suited for the task, whereas other use cases make locally hosted servers the most optimal choice. We have chosen to pursue the serverless architecture for this project as it is encouraged and supported to varying degree by all the major cloud computing providers (Google, Amazon, Microsoft). The most immediate downside is that the field is somewhat new in comparison to the classical way of setting up a cloud infrastructure, so there is not much standardization in regards to technologies and practices for serverless computing, but the whole field is nonetheless in rapid development and as the tech giants themselves are using this to a high degree [Boy00a], serverless computing is being pushed forward rapidly. Using a serverless architecture encourages using microservices, as the serverless approach lets you split up your program in smaller independent parts, and this split helps developers to naturally divide the whole project into smaller specialized parts. If the developers manage to follow this practice, the system will have high cohesion and low coupling build in, and be easy to update piece by piece by independent developent teams.

5.2 Component Diagram

In modelling the system we begin by breaking the system as a physical entity into subsystems or sub-components, which are easier to manage in regards to modeling them in software, but also in relation to the physical location the different components are going to be placed in the physical world. In 5.2, you can see the different components of the system in a superficial view.

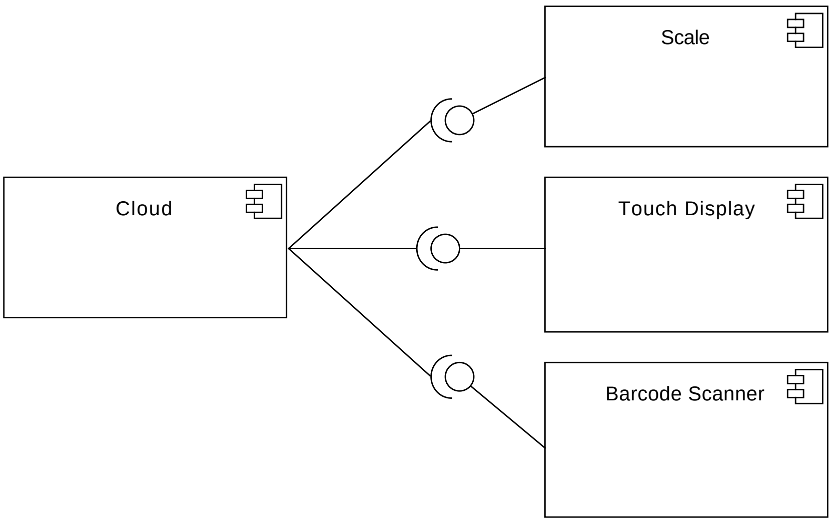


Figure 5.2: Component Diagram of the whole system.

5.3 Barcode Scanner

The barcode scanner component is a semi-independent part of the system which allows to scan the groceries before, after or during their placement on the scale. The component can be further divided into two sub-components; the scanner and the controller.

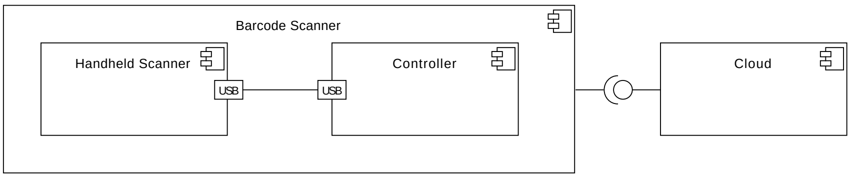


Figure 5.3: Component Diagram of the scanner.

5.3.1 The scanner

The primary requirement of the scanner is that it only needs to be a common (and relatively cheap) 2D scanner for scanning barcodes used for groceries in supermarkets. The idea is to find a scanner using a common interface which represents scanners in general. Some scanners appear to the operating system as a keyboard, as this is the protocol that we as a minimum requirement wish to use, but other protocols are common as well, so it will need further investigation to be able to settle on a protocol which is the best suitable one for this specific purpose.

5.3.2 The Scanner Controller

The barcode scanner is connected to a controller unit, which connects the scanner to the cloud infrastructure, by forwarding the scanned barcodes to the cloud. Also the controller will be programmed to have high resilience towards power outage, program failures, and hardware failures.

5.4 Touch Display

The touch display contains the user interface for the user to enter data and receive instructions from the system during the expedition. Even though there is a display and input buttons installed on the electronic scale which is part of the setup, we have chosen not to use it in the project, as this is not as flexible as using a dedicated touchscreen for user interaction. For example the touch screen allows to update the user interface on a grand scale using updates to the software, whereas the scale, though it can write text instructions to the user and receive number input from the user, would limit the possibilites for interaction to the features offered by the scale. For example in our setup we could show an instruction video on the touchscreen of how the user is supposed to scan the groceries using the hand scanner. This would be impossible to do on the ASCII display of the scale. To keep the experience of the user interaction with the system consistent, we will try not to use the input and output capabilities of the scale but instead use the touch screen exclusively if possible.

5.5 The Scale

The scale component represents the whole setup of the scale as a blackbox which communicates with the cloud. This entity also takes care of recovery in the event of power failure or communication failure etc. I reality the scale component is made up of several subcomponents as can be seen in 5.4.

5.5.1 The Hardware Scale

The scale which has been borrow for this project is a Mettler Toledo Scale model Geo 23, which contains an ethernet connection that is using TCP/IP for communication, and also supports a rs-232 connection. The scale uses the communication protocol SISC [99], so this is a given choice of protocol for the project.

5.5.2 The Scale Controller

As with the handheld scanner, the scale controller acts as an intermediary between the scale and the cloud. As the scale essentially is a server to which the cloud must connect to, to be able to request a weighing or control the settings of the scale. A lot of difficulties arise if the cloud was to connect directly to the scale as the scale is build to do. Therefore, a controller is installed in between the scale and the cloud. This is also a confession to the fact that the scale is not a true Internet of Things device in the sense that it can not itself connect to the cloud infrastructure, and that the communication between them is not secured.

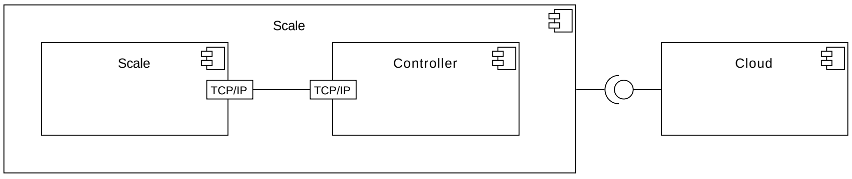


Figure 5.4: Component Diagram of the scale.

5.6 The Cloud

The Cloud represents the business logic for the proof of concept setup. While the scale, the touchscreen and the scanner are just peripheral devices for the user to interact with, the software (business) logic controlling the scenario is carried out by the cloud. The challenge in the setup is to secure resilient real-time communication between the external devices and the cloud.

5.7 Choice of implementation technologies

For the implementation of the above modeled project, several technologies have to be considered for all components of the system. As the project is conductd by one person and under considerable time pressure, the choises of hardware and software will both be influenced by what which programming languages and frameworks he already knows. Also, the hardware choices will be guided by the hardware obtainable within reasonable costs by the author. Of course the above mentioned limitations will be weighted towards the required features of the project

5.7.1 Scale

Figure 5.5: The Mettler Toledo scale.

The scale is, as mentioned above, a Mettler Toledo Scale model Geo 23, which contains an ethernet connection that is using TCP/IP for communication and a serial port. The scale uses the communication protocol SISC, developed by Mettler, so this is a given choice of protocol for the project. As the scale has been borrowed from DTU and is the centerpiece around which the whole project evolves, this choice of hardware is a given one.

5.7.2 Scanner

For the scanner component we wish to find a cheap handheld scanner which uses a simple and common programming interface to be able to represent a broad range of scanners. Also the scanner may be easier to replace in case it is broken or a new supplier of scanner hardware is chosen. For the concrete project our choice of scanner is the Datalogic Touch TD1120, as seen in 5.6. During the research to find a suitable scanner, we found that several standard interfaces for handheld scanners have been development for the Point of Sale industry such as OPOS, JavaPOS and now UnifiedPOS (UPOS), though many vendors use their own proprietary interfaces. In the scanner model we ended up purchasing, we looked for a support for Javapos,

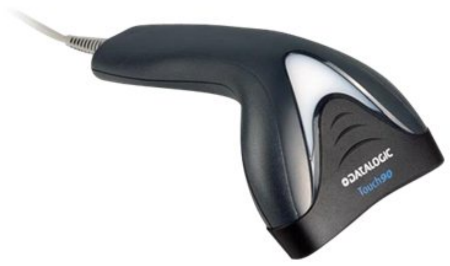


Figure 5.6: The handheld barcode scanner used.

as we are best at Java programming, but ended up actually being able to interface directly to the scanner using a serial interface over USB.

5.7.3 Touch display

The touch display for interaction between user and poc-system will be an iPad 3rd generation. We will not spend time developing an iOS application for the touch screen, but rather implement the UI as a webpage. This will allow us to use other kinds of tablets as touchscreens for the system, but also to update the app faster through the browser, than having to push updates through an app store of a specific platform. If performance or features are not satisfactory using a web-based approach, then custom applications which are either side-loaded or submitted to the app store must be implemented instead. But until performance has been established we would rather decouple the software interface from the hardware which is executing it by using the web browser engine for displaying the UI.

5.7.4 Cloud infrastructure

Choosing the cloud infrastructure for building you application is a big choice for a product, and maybe also for the company building the product. The same is the case for our poc system, but luckily we are neither developing a real product, nor do we have to consider scaling cost, when the project is this small. For investigating the best cloud provider, we choose to make a quick survey of the largest providers - Amazon, Microsoft and Google. All of them provide IoT frameworks. All of them provide lambda clauses (anonymous functions), so superficially they all seem to offer the services we might need. We have choosen to use the Amazon AWS cloud, as it is by far the largest provider [Ram16], with what seems as the largest amount of separate services. But we do recognize that we have not surveyed the market very carefully, so the other providers may offer specific features which might be helpful for our scenario, though we are quite confident that the poc would be implementable on all three providers' platforms.

5.7.5 Blinkt! LEDs

Figure 5.7: The Blinkt RGB diode array.

The Blinkt LEDs are to be used for debugging and user notification. In a more refined setup, the led would be placed more nicely, with labels attached, so that the person using the system had a chance to try to resolve the error, or tell remote support personnel which lights are blinking.

5.7.6 Choice of programming language

We have chosen to use Python3 as the language of choice, as it is sufficiently supported on both the Raspberry Pi platforms and on the AWS platform. If Python3 is not supported, we will use Java if possible. Also, for frontend web page coding, we will have to use Javascript as this is what the most modern frameworks such as Angular an node.js use for generating singlepage webpages.

5.7.7 AWS infrastructure components

For implementing the poc business logic in the cloud we orchestrate several AWS services together to form the system. Several different Amazon services must work together to fulfill the task of running the poc system.

The figure above describes the components comprising the system as it is build in the AWS cloud.

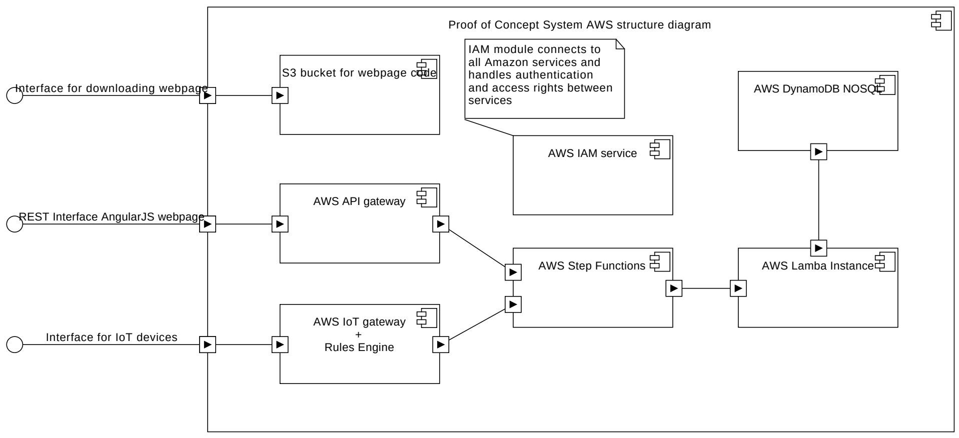

Proof of Concept System AWS structure diagram
IAM module connects to all Amazon services and handles authentication and access rights between services
Interface for downloading webpage
S3 bucket for webpage code
AWS DynamoDB NOSQL
AWS IAM service
REST Interface AngularJS webpage
AWS API gateway
AWS Step Functions
AWS Lamba Instance
Interface for IoT devices
AWS IoT gateway + Rules Engine

Figure 5.8: The Structure diagram of the AWS services containing the business logic.

IAM service

The IAM service is a component shared between all AWS services, and handles authentication and access rights between services, so that compromised services or errors in services are cannot cause havoc in other services which are controlled by the same owner. In a business critical system the rights would be of least privilege between systems, and developers would also be shielded from each other, if there are several teams with divided responsibilities.

S3 Bucket

The S3 service is a file sharing service, but can also be used to serve web-pages. Here it will be used to serve the basic html-page for the touch interface, which in turn invokes the node.js code that is the touch interface.

API Gateway

The touch interface mentioned above will call methods hosted using the AWS API gateway service. The service will be authenticated either by client certificates or by login and password on the web client.

IoT Gateway + Rules engine

The Raspberry Pi's connected to the scanner and the scale are communicating to the AWS cloud using the AWS IoT framework, which uses IoT communication channels based on MQTT [17c]. The clients authenticate using certificates which are also used for communication encryption.

Step Functions

The stepfunctions service can define business processes using statemachines. Normally development of serverless systems involves building the systems to be stateless, but in the case of the business process scenario we need to have separate states for knowing when the customer is in the different states of the process in 5.1. We could also have chosen to manage the state in the node.js client on the touch screen interface, and this would probably scale better in a scenario with several terminals, but we also have to consider the possibility of internet connectivity losses and being able to dynamically change the business process from the serverside instead of having to make huge changes to the touch screen client. If we can centralize the step functions to reside in the AWS we have more freedom to change business processes if the touch screen controller can be told by AWS which data it has to show in the given context. Given more time we would like to explore the two ways to implement the state machine further, with regard to performance, flexibility and reliability.

Lambda engine

The AWS Lamda engine allows the Step Functions engine to call anonymous functions which can be instantiated for short time periods to solve small stateless jobs. Lambda functions can be comprised to or themselves be a microservice. Lambda functions are intended to have low coupling and high coherence, and to only run for short periods of time if possible.

DynamoDB NOSQL

DynamoDB is a NOSQL intance hosted by AWS. For this poc a NOSQL database is sufficient as we will be fetching item names, prices and weight of the item based on the barcode scanned by the scanner. In a real-world setup we would consider using a real SQL database, if we had to do data warehousing, data mining or other tasks using complex data correlations. Sometimes you also use reading and dumping data into a NOSQL database to attain good performance in easily predictable read/write patterns and then copy the data to a SQL-database when you have time and the system load is lower.

In the section above we hope to have described the AWS subsystems and their interdependence in the poc system as we have chosen to model it.

5.7.8 The Scanner Implementation

The scanner is implemented using the scanners serial mode, which is recognized directly as a serial interface in the Linux distribution installed on the Raspberry Pi to which it is connected. The software running on the Raspberry Pi is waiting for the scanner to plece sufficient characters in the serial buffer before sending them to the AWS cloud using and IoT communication channel. The scanner does not receive commands from the business logic in the cloud in the current state of the implementation. We have tried to make the scanner component as simple as possible, and move the business logic to the cloud. When the scanner send an EAN-string to the cloud, it is the responsibility of the Step Function engine to decide if the string arrives at a correct time in the business process, and hence if the sting should be used, discarded, or if an error message should be printed on the touch display to inform the user to follow the correct process.

5.7.9 The Scale Implementation

In the implementation of the Scale component, the scale is sending measurements continuously to the Raspberry Pi computer over a serial-connection. The choice of the serial connection is only due to reducing complexity of the underlying connectivity between scale and computer. We might as well have chosen to use a TCP/IP-connection, but would then possibly have to consider security of the ethernet network, and the possible error modes in the TCP/IP-stack. Therefore, we chose to use the comparatively simple serial-interface until further the needs might change. The scale continuously sends measurements to the Raspberry Pi computer, which is then decides when to forward the measurement to the business logic in the cloud. The Pi computer sends the measurement to the cloud when the scale has settled on a steady weight, and when the cloud receives the measurement, it decides what to do with this measurement based on the state in the Step Functions Engine. The architecture of the scale sending continuously to the Raspberry Pi has been chosen so that the programming in the Pi must not implement a large subset of the SICS protocol to ask the scale for measurements. In stead a producer-consumer pattern has chosen, as this simplifies reduces the complexity of the system without straining the serial-connection bandwidth or the processor of the Raspberry Pi system. The Blinkt LEDs are configured to show in color-codes, when a measurement is sent to the cloud.

5.7.10 Current State of System Implementation

Currently only the Scale and the Scanner components are implemented while the AWS-system is still not completely functional. We have chosen not to test the system in the current state, as we only have separate parts working without the AWS to glue them together, but we will provide a demonstration of the finished pieces of the poc-system and attempt to finish the AWS-system to make a working demo of the whole system.

CHAPTER 6

Conclusion

Cloud technology moves fast with new and rapidly changing standards, hence when we move logic from the scale and scanner to the Raspberry Pi we are offered increased flexibility in adopting new standards and frameworks for integrating the devices into a cloud infrastructure. But a plugin system build into the devices, or a common industry standard for IoT devices would help adoption and longevity of devices by making them upgradeable. Though it is not currently a given thing that manufacturers of consumer electronics wish to extend longevity of products, whereas business and industry product manufacturers, like Mettler Toledo in this case, may wish to do so for long term customer relations and because the products they develop are in use in industry for many years.

What is needed is industry standards for securing the computing devices, making them easy to update, both to possibly add new features but more importantly to patch identified security holes or update security standards in these devices, if the already implemented ones are outdated or flawed. In case of the scale and the scanner, we need to move more control into flexible software, while still moving as much of the business logic into "the cloud" or other external hardware. Only in more specialized cases where latency or bandwidth limitations etc. are a problem to a centralized business logic, does it make sense to movie compute power into the peripherals instead of using "dumb" external devices in tandem with serverless architectures. An example of this may be a credit card terminal that does part of the integrity checks locally in a secure chip instead of sending all the credit card data to centralized servers for authentication etc. The cloud providers of course continue to develop their PaaS services, but we still need mature IoT frameworks to glue between the PaaS and the devices, and hopefully have commonly used industry standards so that provider lock-in will be less prevalent.

With regards to optimizing the business process of the groceries shopping scenario, we must conclude that we have already modeled the most optimal process, until smart phone scanning of the groceries is viable in the danish market. This change will fundamentally change the process, as the number of cash registers and the roles of the employees in the supermarket will change. Technologically there is no problem implementing this system as it is already being tested in Denmark, and is being actively used in other countries [KÜH00]. Here a scale for weighting the sum of the scanned groceries might be part of the solution as a fraud prevention mechanism, in contrast to the manual random inspections done today.

Sadly we did not implement as much of the modeled system as we hoped, and hence a formal test of the system did not make sense in its current state. We hope to be able to demonstrate a more complete system soon, where more of the resiliency and robustness mechanisms can be demonstrated to work.

CHAPTER 7
Glossary

Checkout line - Point of sale (or POS) - point of purchase (POP). This is the place where customers pay for their groceries after picking them up in the supermarket.

Escalator band - conveyor belt. This is the part of a classic checkout line where the customer packs the groceries on a belt, while an employee scans the barcodes on the items.

Self service - self-checkout. This is the system where a customer instead of going to a checkout line, goes to scan the groceries, pay for them, and pack them herself. We implement at Proof of Concept system doing this.

Scale - The scale can weight items, such as grocery items.

Handscanner - handheld scanner - barcode scanner or scanner. This is an electronic scanner which is able to scan standardized International Article Numbers (EAN) coded barcodes into a computer system.

Employee - Staff - Clerk. The persons who work at the supermarket and among many things man the checkout lines and the self service systems.

Bibliography

[06] *Project Cartoon: How Projects Really Work (Version 1.5)*. July 24, 2006. URL: http://www.projectcartoon.com/cartoon/2 (visited on January 15, 2017) (cited on page 17).

[16a] *Non-Functional Requirement*. In: *Wikipedia*. Page Version ID: 756105366. December 22, 2016. URL: https://en.wikipedia.org/w/index.php?title=Non-functional_requirement&oldid=756105366 (visited on January 11, 2017) (cited on page 12).

[16b] *Real-Time Computing*. In: *Wikipedia*. Page Version ID: 754713139. December 14, 2016. URL: https://en.wikipedia.org/w/index.php?title=Real-time_computing&oldid=754713139 (visited on January 11, 2017) (cited on page 12).

[16c] *Software Development Process*. In: *Wikipedia*. Page Version ID: 754769301. December 14, 2016. URL: https://en.wikipedia.org/w/index.php?title=Software_development_process&oldid=754769301 (visited on January 17, 2017) (cited on page 17).

[17a] *Bip Og Betal - Bip Dine Egne Varer Og Spring Køen over*. January 28, 2017. URL: http://superbrugsen.dk/bip-og-betal/ (visited on January 28, 2017) (cited on page 15).

[17b] *Information Security*. In: *Wikipedia*. Page Version ID: 759506588. January 11, 2017. URL: https://en.wikipedia.org/w/index.php?title=Information_security&oldid=759506588 (visited on January 11, 2017) (cited on page 13).

[17c] *MQTT*. 2017. URL: http://mqtt.org/ (visited on January 29, 2017) (cited on page 26).

[99] *Reference Manual - Standard Interface Command Set (SICS)*. 1999 (cited on page 22).

[Boy00a] Magnus Boye. *Fra Monolit Til Microservice: Sådan Genbyggede Netflix Sig Selv I Skyen*. 2016-12-15T05:11:00+01:00. URL: https://www.version2.dk/artikel/monolit-microservice-saadan-genbyggede-netflix-sig-selv-skyen-1070500 (visited on January 25, 2017) (cited on page 20).

[Boy00b] Magnus Boye. *Træt Af It-Monolitten? Prøv En Microservice.* 2016-12-16T05:03:35+01:00. URL: https://www.version2.dk/artikel/traet-it-monolitten-proev-microservice-1070559 (visited on January 25, 2017) (cited on page 19).

[Boy00c] Magnus Boye. *Slip for VM's Og Containere: Serverless-Arkitektur Skærer Alt Andet End Kode Væk.* 2016-12-21T05:09:00+01:00. URL: https://www.version2.dk/artikel/slip-vms-containere-serverless-arkitekturen-skaerer-alt-andet-end-kode-vaek-1071023 (visited on January 19, 2017) (cited on page 19).

[Cat14] Mary Catherine O'Connor. *Can RFID Save Brick-and-Mortar Retailers after All?* April 17, 2014. URL: http://fortune.com/2014/04/16/can-rfid-save-brick-and-mortar-retailers-after-all/ (visited on January 28, 2017) (cited on page 16).

[KÜH00] JOACHIM KÜHLMANN SELLIKEN. *Kvickly Og Super Brugsen Vil Afskaffe Køer: Sådan Fungerer Den Nye App.* 2015-11-05T09:01:00+01:00. URL: http://www.computerworld.dk/art/235424/kvickly-og-super-brugsen-vil-afskaffe-koeer-saadan-fungerer-den-nye-app (visited on January 28, 2017) (cited on pages 15, 29).

[MdM17] David Mckenzie and Suresh de Mel. *Lessons from a Technology Flop: RFID Works for Walmart but Not for Small Business Owners in Sri Lanka | Higher Education and Development | U.S. Agency for International Development.* January 9, 2017. URL: https://www.usaid.gov/education/development-higher-education/aug-2016-lessons-technology-flop-rfid (visited on January 28, 2017) (cited on page 16).

[Ram16] David Ramel. *Microsoft No. 2 Behind Amazon in Cloud Market Share -.* August 2, 2016. URL: https://rcpmag.com/articles/2016/08/02/microsoft-behind-aws-in-cloud.aspx (visited on January 27, 2017) (cited on page 24).