



University of Luxembourg
Faculty of Sciences, Technology and Communication

Master's Thesis

Assembly Tracking and Scheduling Improvement for Assembly Lines based on RFID technology

by

Tom Schmitz

Supervisor :

Dr. Meysam Minoufekar (University of Luxembourg)

Luxembourg, 18th January 2018

Abstract

The main goal of this paper is to improve productivity by measuring and adapting the assembly process for any given product. In order to track the production chain, each part in the supply chain is equipped with an RFID-tag which will be recorded during its travel through the facility. Also each worker has their RFID-tag to localize oneself and record the performed activities. The workstations are equipped with an RFID-scanners which will be used to record activity and product flow through the stations. All the gathered data is collected on a server and the real-time status of the assembly line is processed and displayed to the dispatching agents. Upon this data analysis, the dispatchers can take actions, update the manufacturing setup and ultimately increase productivity.

Acknowledgements

There is a number of elements which lead to me being able to perform and adequately document this work. I always like to refer to the thesis as a "project" because it felt more like leading a two-man team consisting of the always motivated Luc Ohles who was constantly ready to squeeze some code out of my concept ideas. Luc deserves my greatest respect for sharing his experience in the domain of micro controllers and software development. It is a blessing to have someone this motivated and capable on your side. Also to be thanked is Meysam Minoufekar who supported this masters thesis as if it was his own project. The trust put into Luc and myself is something reassuring and one of the biggest motivators for pushing out this quality of work during the short time available. Also Special thanks to Christof Oberhausen who allowed us to partly disassemble the existing test assembly line at the Lean-lab of University of Luxembourg and take what we needed for the tests performed. The hope also being that we can at one point also completely integrate our system into the setup of the Lean-lab. I also want to thank the volunteers which did proof-reading, *Balázs Pejó* and Mariia Zubova and those who participated in the testing of the system, Vishojit Bahadur Thapa and Patrick Pereira Dias.

Contents

1	Introduction	1
1.1	Background	1
1.2	Scope	2
1.3	Outline	2
1.4	Methodology	2
2	Fundamentals and State of the Art	4
2.1	RFID-Tags	4
2.2	Raspberry-Pi3	5
2.3	Windows IoT	6
2.4	Wi-Fi	7
3	Domain Analysis	8
3.1	The current State	8
3.2	Actors of the shop floor	10
3.2.1	Assembly worker actions	10
3.2.2	Supply-Chain Worker Actions	12
3.2.3	Quality Assurance Worker actions	13
3.3	Measurable factors	14
3.4	Performance Indicators	15
4	Solution Path	17
4.1	Solution Concepts	18
4.1.1	Requirements Analysis	18
4.1.2	Concept options	19
4.2	Evaluation of Platforms	20
4.2.1	R1: User identification technologies	20
4.2.2	R2: Object identification technologies	20
4.2.3	R3: Time measurement technologies	21
4.2.4	R4: Data collection Platforms comparison	21
4.2.5	R5: Data processing Platforms comparison	21
4.2.6	R6: Data communication medium between system nodes(here: rbpi3s)	22
4.2.7	R7: Visualization of the calculation and measurement re- sults	22
4.3	Evaluation of the target Solution	23
4.3.1	Visualization through use-case model	24
4.3.2	The updated assembly line structure	25

5	Implementation	27
5.1	Workstation Interface	29
5.1.1	Exemplary run of an assembly workstation	31
5.1.2	Exemplary run of a supply-chain workstation	39
5.1.3	Interactions between different workstations	41
5.2	Dispatching-Interface: Product	42
5.3	Dispatching-Interface: Line (VSM)	43
6	Experimental Validation	46
6.1	Experimental Setup	46
6.1.1	Assembly Stations	48
6.1.2	Testing Procedure	50
6.2	Results	52
6.2.1	Product Views	52
6.2.2	VSM View	54
6.3	Discussion	54
7	Conclusion	55
7.1	Summary	55
7.2	Future Research	56
	Bibliography	58

List of Figures

2.1	RFID equipment options.	5
2.2	The Raspberry Pi 3B Board with feature indication. ¹	6
3.1	Progress through this Chapter showing the approach from bottom-up.	8
3.2	An assembly line how it is traditionally implemented within a factory shop floor.	9
3.3	Assembly images for the treated example.	11
3.4	A package containing the required part of "ID-001"	12
4.1	The roadmap followed to identify the target solution.	17
4.2	Requirements analysis addressing the challenges to the system. .	18
4.3	The previous open challenges in Figure 4.2 have been replaced with technological solutions.	23
4.4	A specific use case showing the chosen technologies and typical user-interactions	24
4.5	An assembly line how it ideally is interconnected after we deployed our infrastructure layer.	25
5.1	Interfaces of this software and their paths of generation.	27
5.2	The system architecture following the Client-Server-model. . . .	28
5.3	The 7 inch touchscreen display, its back-plate and mounting components.	29
5.4	The Interface for each workstation touchscreen	30
5.5	Sequence diagram simulating	32
5.6	Sequence diagram state 0	33
5.7	Sequence diagram state 1	33
5.8	Sequence diagram state 2	34
5.9	Sequence diagram state 3	34
5.10	Sequence diagram state 4	35
5.11	Sequence diagram state 5	35
5.12	Sequence diagram state 6	36
5.13	Sequence diagram state 7	36
5.14	Sequence diagram state 8	37
5.15	Sequence diagram state 9	37
5.16	Sequence diagram state 10	38
5.17	Sequence diagram state 11	39
5.18	Sequence diagram state 12	39
5.19	Sequence diagram state 1	40

5.20	Product flow of the workstations.	41
5.21	The View of the product assembly Interface	43
5.22	The real-time assembly-line status Overview	44
6.1	The product assembled during the test runs.	46
6.2	The layout of the Assembly line Stations.	47
6.3	Experimental Setup of the Assembly line Stations.	48
6.4	Instruction Set for Station 2.1 and 2.2	49
6.5	Instruction Set for Station 3.	49
6.6	Instruction Set for Station 4.	50
6.7	Side by side comparison of the paper-version instructions and the new interface.	51
6.8	Assembly trace and practical evaluation of the red puncher test.	52
6.9	Product view for the assembly trace of the blue puncher.	53
6.10	Product view for the assembly trace of the black puncher.	53

List of Tables

4.1	Technological options overview	19
4.2	Comparison matrix for user identification technologies.	20
4.3	Comparison matrix for object identification technologies.	20
4.4	Comparison matrix for data collection platforms.	21
4.5	Comparison matrix for communication solution.	22
4.6	Comparison matrix for visualization software.	23

Chapter 1

Introduction

1.1 Background

The world of manufacturing is for many years already on the turn where the consumer requests more personalized products that fit exactly their needs. The one who offers this product in the requested variety will get the market share. This has been elaborated already years ago by Macduffie, 1996 [1].

Additional to the requirement of functional personalization comes the requirement of products that fit the style of the customer and reflect their mentality and social status. This behavior explains why there is a strong need to manufacture even everyday day objects like bottles in different colors, shapes and materials with no direct improvement in quality or usability. Those products still belong to the same product family and will be assembled often in the same plant and follow the same assembly line even though they use differently sourced parts which can also differ in color and shape.

Furthermore the manufacturing world is strongly moving away from a push-to a pull-manufacturing which requires highly flexible on demand assembly lines. Such lines feature the challenge of stock management and just-in-time supply management for the assembly lines.

A topic also addressed by costumers and governments is the need for a more sustainable resource management and protection of the environment. The main aspect here is waste. From the point of view of an assembly line, waste is scrap material and damaged or wrongly assembled end products. The cost of reassembling, reworking such products can many times outweigh the cost of the product itself and result in direct waste. Also it has to be understood that some products can hardly be disassembled as they are not designed for this use-case because their recycling is ignored or only possible after destruction. Many objects are therefore shredded and then sorted by material type solely.

These are problems that can be addressed by introducing RFID technology into the assembly process and supply chain. The last argument about recycling potential plays a role in research performed by Saar, 2002 [9]. The supply chain management has partly already been addressed by Huang in his paper [3] which deals with the step of resource management and tracking of parts from the store to the workstations. Although there is no experimental result shown in Huang's paper [3], we will try to experimentally determine how viable the presented approaches are.

1.2 Scope

The Scope of this thesis is to make use of RFID technology to record live data of assembly lines, process this data and visualize it to facilitate decision making, layout and scheduling optimization for the given assembly line. This will involve tracking the parts and workers of the facility, introducing a digital real-time interface for workstations and visualization of performance statistics on large screens for dispatching personnel to see.

The target is to develop software which fulfills the presented requirements. The submission version of the software should be stable enough to perform tests and assess a test assembly line.

1.3 Outline

The Structure of this paper will follow the same timeline that the Project follows in general which is oriented along the "KAIZEN" [4] approach from Toyota. The latter one can be characterized by the four main steps of "plan", "do", "check", "act" (PDCA [10]). The software development is centered to serve the means of KAIZEN best while making use of RFID technology. The steps of improvement of the assembly-line are similar but increased in efficiency and information flow.

1.4 Methodology

Following the above mentioned structure, the first step is to understand the challenges of manufacturing plants. For this purpose we have a fully operational small-scale model at hand here at the University of Luxembourg which is called the "Lean-lab" [6]. The Lean-lab represents an assembly line for punchers which is complex enough to accurately simulate the effects that play a role in a real plant. The workspace is equipped with state of the art tools like an Andon Board [5], ergonomically designed and features the same problems that also arise at any given facility. The main reason for the existence of this assembly line apart from research is its educational factor. Students can experience by themselves how assembly lines are operated and improved by modern research

and technologies. The contribution of this work will ultimately be to integrate the RFID-based system within the local educational assembly line and provide new ways to visualize the assembly process, its bottlenecks and in the end provide useful statistics and analytics.

The first step is a planning phase in which we analyze the existing technological boundaries and introduce the RFID tracking wherever possible and useful to achieve a reliable measurement. After presenting the plan, its time to move towards the first implementation. This represents the development of the measurement and workstation unit and its deployment on the local assembly-line or on a smaller test assembly-line. The main goal is here to track the workers activity and the assembly steps undertaken. It is important to us to provide an intuitive and interactive instruction-set to the worker which will adequately inform about the next tasks.

Upon a first implementation, this one needs to be tested and evaluated. The measurement results from a shift are interpreted periodically and displayed to the dispatching team of the assembly line. Here it is crucial to determine which metrics best reflect the overall productivity of the assembly line. The better the statistics, the more ease the dispatching team will have to reorder the assembly-line and achieve a higher production rate. The final and open step of this work will be to propose ways to further develop the software and integrate it in larger assembly-lines. The improvement of the assembly-scheduling based on the gathered data and developed analytics tools shall be compared. The whole software back-end features a modular design which will allow for quick changes and reallocation of assembly steps.

Chapter 2

Fundamentals and State of the Art

This chapter will cover the basic hardware knowledge required to understand the concept of this paper and introduce their distinctive features which are discussed later in more detail.

2.1 RFID-Tags

RFID, Radio-Frequency-Identifier tags are manufactured in different sizes, shapes and functionalities. There are active and passive tags. Active tags are equipped with their own energy source and send an identification signal periodically or triggered from the outside. A passive tag will only function when near to an RFID-reader. Such readers feature an electromagnetic field which will make a current flow within the circuit of the RFID-tag and activate it. Upon activation, the reader can communicate with the tag and retrieve the information stored on the chip. When the tag is removed from the energy source, it will stop sending. Those passive tags can be manufactured in very small form-factors and shapes to match numerous applications. Those are small enough to be part of wrist-bands, rings, thin stickers of 1cm^2 and fit into standard key-ring-badges. The reading distance varies from direct contact between reader and tag up to 5 centimeters depending on the reading device. The distance of detection is limited by the form factor of the tag and by the strength of the reader.

Three types of RFID-tags will be used during this work:

1. The RFID-Wristband which serves the means of identifying an employee. Shown in Figure 2.1a
2. The RFID-Tag (keychain) shown in figure 2.1b. This tag type is the most resilient and practical for our testing purposes and was used to debug code

and test new features with mocked parts.

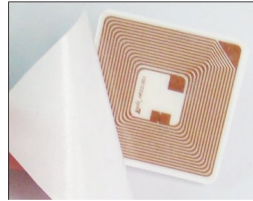
3. RFID-Stickers are the smallest variant used for this work. The stickers can be adhered to any part and are available in different shapes and sizes. One type used is shown in figure 2.1c.



(a) RFID-Wristband.¹



(b) RFID-Tag.²



(c) RFID-Sticker.³

Figure 2.1: RFID equipment options.

2.2 Raspberry-Pi3

Raspberry Pis are small Boards which feature a processing unit, pin headers and other IO devices like audio, video and USB-ports. This all together on one board results in a very compact but powerful computing unit which can run dedicated software, apps and even some lightweight operating systems.⁴

For this thesis we use Raspberry Pi 3 (short RbPi) devices. Those are combined with a 7" touch-screens. The RbPi3s connection to the RFID-scanner and further usage has been previously elaborated by Patryk Pomykalskis in his Bachelors Thesis: RFID sensor based collaboration platform in IoT environment[7].

For the purpose that this project follows, the small RbPis are best suited since they take up a minimum of space, are lightweight, and they feature an attractive band of connectivity options. The integrated Wi-fi simplifies installation on larger assembly-lines and open spaces since it reduces the need for

¹Image from:<https://www.buyapi.ca/product/rfid-tag/>

²Image from:<https://www.electronic-shop.lu/DE/products/144923>

³Image from:<http://chuangxinjia.en.hisupplier.com/>

⁴Raspberry PI 3 Device information and official Manufacturer: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

physical connections between the RbPiS and the server. A further plus point is the low power consumption and therefore also low heat-production which does not require the chips to rely on 3rd party cooling solutions. This reduces noise levels and installation requirements. The RbPiS can even be deployed on moving workstations/platforms like dispatching trolleys, maintenance carts and cranes. In later Projects the Wi-Fi connection could also be used to connect augmented reality devices such as the Holo-Lens⁵.

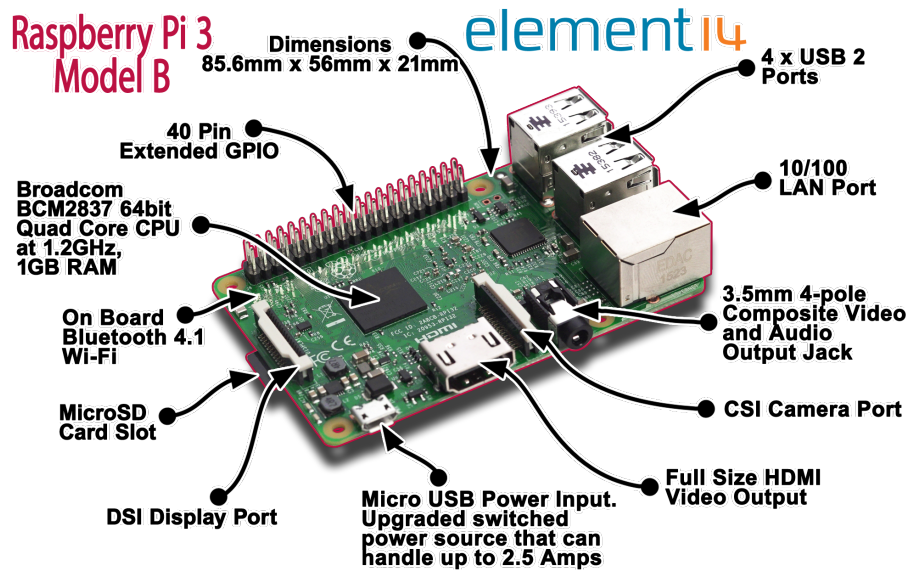


Figure 2.2: The Raspberry Pi 3B Board with feature indication.⁶

The 3rd version of the RbPi features also an HDMI port which we will be using to display the interfaces for each workstation. An audio-Jack in/output can be used to gather environmental data such as noise and evaluate working conditions but also to give audible feedback when a tag is scanned or if an error or special event has occurred. Last but not least 4 USB-ports can be used to connect further input/output devices such as Andon-buttons [5], keypads and photoelectric-barriers.

2.3 Windows IoT

Windows-IoT(Internet of Things) seems to be just made for the purpose of this project. The idea of Windows-IoT is to be deployable on lightweight, low power smart-devices like a smart-fridge which will order the supply on new food whenever its empty or a light that will remember when its most probable to be

⁵Explanation and description of the Microsoft Holo-Lens

<http://searchmobilecomputing.techtarget.com/definition/Microsoft-HoloLens>

⁶Image source: <https://www.element14.com/>

turned on and what intensity it should have depending on a set of inputs from various sensors from the same or other connected devices. Windows IoT features a complete set of libraries based on asp.net⁷ which can be extended with a broad set of extensions which allow for different use-cases. The communication between devices and the database is managed clearly and without hassle. The programming in c#⁸ in visual studio is straight forward and will allow for flexible and extensible software. Our project is by far not exceeding the possibilities of Windows IoT when deployed on the given RbPi model 3. This said the aim of this project is still to focus on quick user-interaction feedback and real-time tracking of the workstations, which follows the path of lightweight communication protocols like HTTP⁹. This procedure allows for minimal network-load and the reduction of buffer-times and pings which again will make the interaction with the devices more seamless and the feedback from tracking algorithms more responsive.

2.4 Wi-Fi

Wi-Fi is the means of connecting multiple devices in a wireless lan network. Generally one or more devices act as a router here and forward packages to their destination. The RbPi devices are equipped with this feature. For the connection to a server or Windows workstation, a Wi-Fi-router can be employed. Wi-Fi is an interesting means of communication for our purposes since the distance covered is fitting the needs of this thesis which is about 20 meters in radius but can be extended easily with multiple routers. The reliable package transmission without losses is also a plus.

⁷Official information about Windows-IoT: <https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx>

⁸C-sharp is a programming language based on the older C language. C-Sharp is proprietary to Microsoft and used in .Net, Windows etc.

⁹Hypertext tranfer protocoll - https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Chapter 3

Domain Analysis

The domain analysis will lead the research from the existing to the improved state. Figure 3.1 illustrates the intermediate steps and serves as road-map for this chapter.

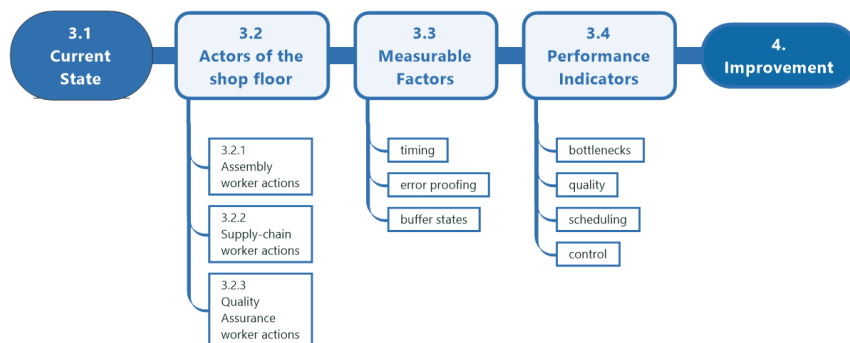


Figure 3.1: Progress through this Chapter showing the approach from bottom-up.



3.1 The current State

Assembly lines are in general rigid systems that are updated only rarely. This mostly if larger changes and improvements are taking place. The saying "never change a running system" is more than an unwritten rule since each stop of an assembly line will cause disruptions of the product flow. Since most industries focus on a pull rather than a push strategy, stock and buffer management is key to maintain a smooth running assembly line while allowing for human mistakes

and other low impact interruptions of the work-flow. Still those buffers should be kept as low as possible since they increase manufacturing time and startup-time of the complete line.

As for the continuous improvement process, CIP[2] or also called KAIZEN[4] the implementation is at best shift iterative. Changes have to be planned with great care and be trained with the employees before implementation. A change may be for example the addition of a workstation which adds more quality to the product surface by polishing. Another possible change would be to reschedule the assembly line by migrating tasks from one workstation to another one or restructure the sequence order or the assembly steps.

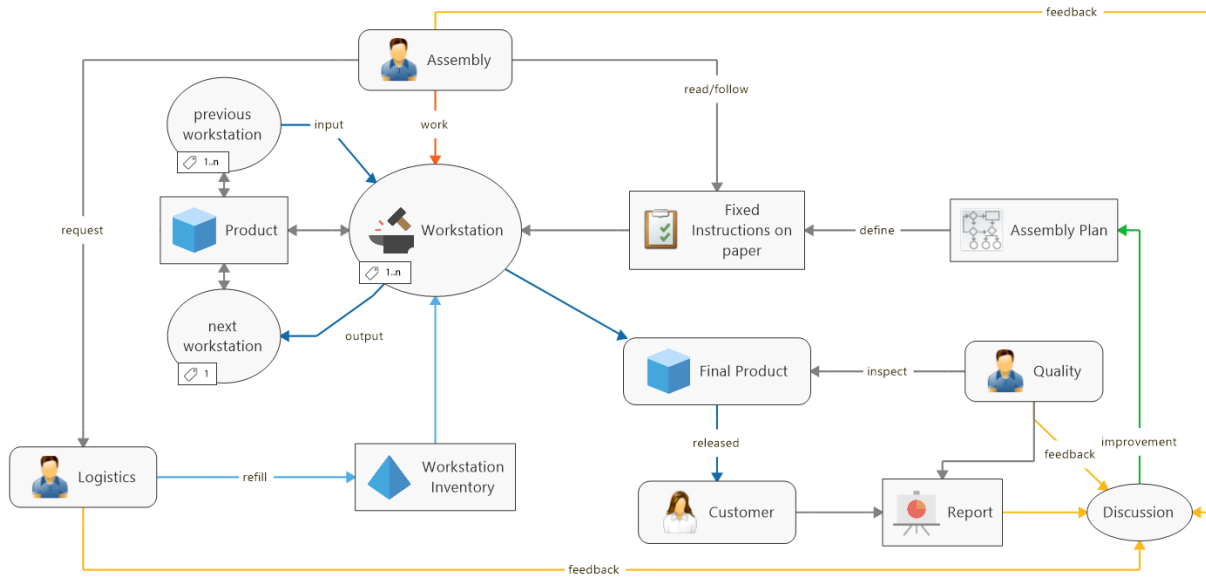


Figure 3.2: An assembly line how it is traditionally implemented within a factory shop floor.

Figure 3.2 depicts the material and information flow of a traditional assembly line which incorporates a continuous improvement process.

- Grey arrows are assembly or dispatching instructions.
- Yellow arrows show the flow of feedback from various players.
- The blue arrows indicate the movement of parts and products in the supply chain.
- The orange arrows show the manual labor input from a worker.
- The green arrow shows the performed improvement.

It is important to notice at which stages informations are passed, and when the discussion occurs, which will ultimately trigger an improvement of the assembly line. The discussion can not take place until all the different actors have given a complete report on the situation which is at hand. It should also be noted that the customer is involved in the process of feedback since this one will ultimately be the actor whose needs are targeted.



3.2 Actors of the shop floor

There are 3 main categories of workers which are directly influenced by the technology which will be addressed during this thesis.

Those are :

1. Assembly workers
2. Supply chain workers
3. Quality assurance workers

Other workers which are not affected by the scope of this thesis but may benefit from an RFID implementation are maintenance, safety and protection workers.

3.2.1 Assembly worker actions

The tasks of an assembly worker can vary strongly between applications and product which is being manufactured but generally the worker will manufacture a product which is made of a certain amount of parts and have a given set of basic instructions which guide him/her through the assembly process for his/her workstation and product. Those tasks can be broken down into atomic steps, atomic meaning that those steps can not be broken down any further for a human worker and require the most basic of concepts that a human can understand and execute intuitively.

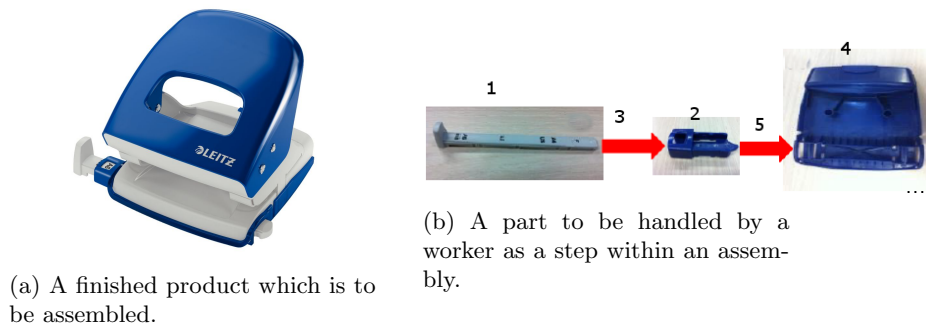


Figure 3.3: Assembly images for the treated example.

Example for an Assembly Worker:

"Assemble the paper-size indicator and the puncher bottom plate."

Following this principle, the steps for the task mentioned above could for example be:

1. Grab the paper-size indicator.
2. Grab the guidance piece.
3. Insert the shaft into the guidance piece.
4. Grab the backplate of the puncher.
5. Push the paper-size indicator with the guidance piece into the hole on the bottom of the backplate.
6. Test if the paper-size indicator slides without obstruction.
7. Put the finished Subassembly into the Out-box

These steps are to be executed within a given time frame and with a defined quality which addresses mostly human errors. An assembly error can result in a rework or scrap product but in any case creates waste in the form of manufacturing time since an error needs to be corrected which costs time. This time can in the best case be invested by the worker who performed the error if it is discovered in time or has to be found during a quality check at the end of the production chain which then becomes more costly in terms of time. Therefore it is of great use to track the assembly times and crosscheck at each step if the latter was performed correctly or if rework is required. The less errors are

produced, the better the quality and more smooth the entire manufacturing line will run. The ideal case is that all workstations perform within the set takt time and none to few Buffers are required.

3.2.2 Supply-Chain Worker Actions

A worker of the supply-chain has tasks which are always bound to delivering an object from a store to a consumer like a workstation. Such actions can also be mapped as tasks with certain steps to perform in a scheduled manner.

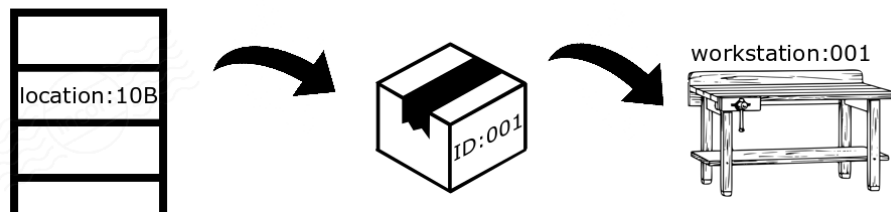


Figure 3.4: A package containing the required part of "ID-001"

Example for a supply-chain worker:

"Deliver 4 Shafts from the storage location 10B to the workstation 001 and 6 shafts to the workstation 002"

1. Go to the storage location 10B
2. Load a quantity of 10 of the Object "Shaft,ID-001"
3. Go to the workstation 001
4. Unload a quantity of 4 of the Object "Shaft,ID-001"
5. Go to the workstation 002
6. Unload a quantity of 6 of the Object "Shaft,ID-001"
7. Return to the dispatching station

With these simple steps at hand there is a way to have control over how the tasks are executed. For a delivery of multiple objects through the shop-floor there are optimal ways to address the problem which can be computed with a milk-man run algorithm for example. Also the paths between each delivery can be optimized based on the actual traffic on the shop floor and possibly occurring blockages of paths due to maintenance works. Once the paths have been optimized and the instructions are given in this detail, the errors which can occur are traceable and can be recovered in real time. Alternative solutions to the errors should be prepared. A decision tree can be computed for each possible solution solving each state of error. An example of a possible error is that the Payload is lost partly. How to recover such an error and which implications it has will be analyzed in the following chapter.

Many factories have automated a large number of their in-house supply-chain but still a human worker is required to finally bring the parts in their final place since autonomous robots are very good at retrieving items from large storage-shelves but perform worse for delivering goods and place them into a feeding box of a workstation while this one is being operated by a human. Especially since many parts have different shapes and weights. Some items are hard to navigate through the shop floor while others can cause a danger if dropped upon a crash due to their high weight. Also numerous tools are required to transport different types of objects and not every action can easily be automated.

In order to still manage and control the supply-chain many products are equipped with a bar-code which contains an identifier. This one is static and can not be changed without being printed and glued again on the object or its packaging. A bar code scan will not reveal the current state of the transport since it can not be altered along the delivery process. To identify the state, the reading device has to connect to a server and request this information.

3.2.3 Quality Assurance Worker actions

Quality assurance in our case is mainly the operations which check whether the product is assembled correctly and performs as expected during stress tests or functional crosschecking. The work of a quality assurance worker is most of the times the last step before the packaging of the products. This means that this work is key to guarantee a consistent high quality throughout the entire run of a given production line. The work needs to be documented for future use or in case of complaints and also reflect the state of the assembly line itself. Mistakes at this stage can not be recovered anymore and lead to high costs if products have to be replaced later on.

Like earlier done we also want to give an example for a job of a quality assurance worker.

Example for a quality assurance worker:

"Check the product xyz001 for functional integrity"

1. Retrieve the Product Number "xyz001".
2. Fix Product in testing station.
3. Wait until testing complete.
4. Check results.
5. If results positive/negative then perform action A/B.



3.3 Measurable factors

After having set up the actions which are generally performed on the shop floor and mapped them to possible mistakes which can occur, it is possible now to inspect how to check for errors. The factors to be recorded and measured will be triggered by an event like a worker interacting with an object or performing an action. Those factors represent the raw data collected during manufacturing.

Measurable indicators of actions are then:

- Worker ID who is processing the task.
- Object ID which is being interacted with.
- Product which is being worked on.
- Time at which the task is started.
- Time at which the task is finished
- Workstation ID at which the task is started.
- Workstation ID at which the task is ended. (for supply chain actions).

The above listed indicators reflect the set of data to be collected during the assembly process without obstructing the Action of the worker him-/herself too much while yet yielding a strong basis for future analysis.



3.4 Performance Indicators

To now assess the rate of errors and their magnitude in terms of time lost during manufacturing, the mentioned factors should be interpreted. The interpretation of the collected raw data leads then to performance indicators which clearly show the current state of matter and support the deciding actors on which change to make on the assembly line to increase its performance.

Performance Indicators answer to specific questions, therefore we want to first formulate the questions which arise from inspecting the assembly actions and then deduct the indicators to compute.

Variable definition for the following formulas:

T is the amount of tasks to complete an assembly of a product.

x is the id of a task between $0:T^1$.

P is the amount of Products produced so far.

y is the id of a specific product manufactured between $0:P$.

For the means of composing the performance indicator formulas, some functions are required. The required functions are in the code to be written in object oriented C# of which the first performance indicator expressing the task execution time is written in pseudo-code hereafter for demonstrative purposes.

```

1  Int GetTaskExecutionTimeMs(int task, int product, assemblyLine asl)
2  {
3      task t=asl.getProduct(productIndex).getTask(taskIndex);
4      int taskStartTimeMs = t.getStartTimeMs();
5      int taskEndTimeMs = t.getEndTimeMs();
6      return(taskEndTimeMs - taskStartTimeMs);
7  }
  
```

Listing 3.1: Function to retrieve the execution time from a Task performed on a sepcific product.

For the means of understanding and simplicity the following performance indicator formulas are expressed not under pseudo-code form but as mathematical equations with integrated functional notations. The notation is Matlab² inspired since this is the most commonly accepted mathematic programming language.

¹The " $0:T$ " expression describes the array of values from 0 to T in this case. This is equal to the set $[0, N]$ of natural numbers, \mathbb{N} "

²For more insight into the matlab syntax consult: https://nl.mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html

- How long did Task x take to be executed for product y?

$$TaskExecutionTime(X, Y) = TaskEndTime(x, y) - TaskStartTime(x, y)$$

- What is the average, Maximum and Minimum execution Time for Task x during the current run of the assembly line?

$$Avg(\Sigma TaskExecutionTime(x, 1 : P))$$

- Was the task performed correctly? i.e. was the right object interacted with?

$$ItemPickedCorrect(x, y) = ItemRequired(x, y) == ItemPicked(x, y)$$

- How long was Task x waiting before it could be executed?

$$TaskIdleTime(x, y) = TaskStartTime(x, y) - TaskEndTime(x, y - 1)$$

- Which worker performs best on Task X?
= WorkerID

- Which task is blocking Task x from execution?
Complex request which required to see the dependency tree for parts sourcing of the Task.

- How long does an item wait in the buffer before being processed?

$$ItemIdleTime(x) = TaskStartTime(x, y) - TaskEndTime(x - 1, y)$$

- How long is an item in transit between Workstations? (in this case the task is to supply the item from workstation A to workstation B)

$$TaskExecutionTime(x, y) = TaskEndTime(x, y) - TaskStartTime(x, y)$$

- How long did it take to manufacture Product y?

$$\Sigma TaskExecutionTime(1 : N, y)$$

- Were there any errors during the assembly of Product y?

$$ProductErrorCount(y) = \Sigma ItemPickedCorrect(1 : N, y)$$

- What is the relative waiting time during the Assembly of Product y?

$$waitingTimeRel(y) = \frac{\Sigma TaskIdleTime(1:N,y)}{\Sigma TaskExecutionTime(1:N,y)}$$

- Which Task has the highest error rate over all the products produced?

$$max(ProductErrorCount(1 : P))$$

- Which Task has the longest waiting Time before execution? (=Bottleneck)

$$max(TaskIdleTime(1 : P, y))$$

- etc.

Chapter 4

Solution Path



Following the research question the goal is to design a system which is capable of measuring and recording the factors mentioned in Section 3.3, process those factors and visualize the results in a way which is understood intuitively. Figure 4.1 projects the steps taken to funnel down from a given set of key requirements to a target solution. This is mainly done by comparing different technologies and determining the appropriate combination of those. The goal for further development is to be able to build a platform which is functional yet expendable in the future and can be tailored to the specific needs of each assembly line.

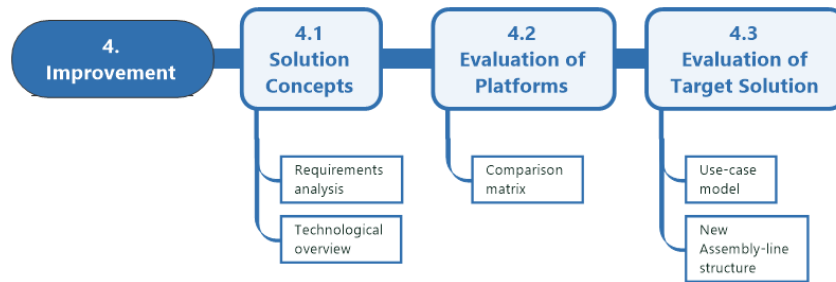


Figure 4.1: The roadmap followed to identify the target solution.

4.1 Solution Concepts



4.1.1 Requirements Analysis

Emerging from the determined Factors, the requirements to the system in question are formulated in Figure 4.2. Those are the most basic tasks, the system needs to be able to perform in order to compute the earlier mentioned factors.

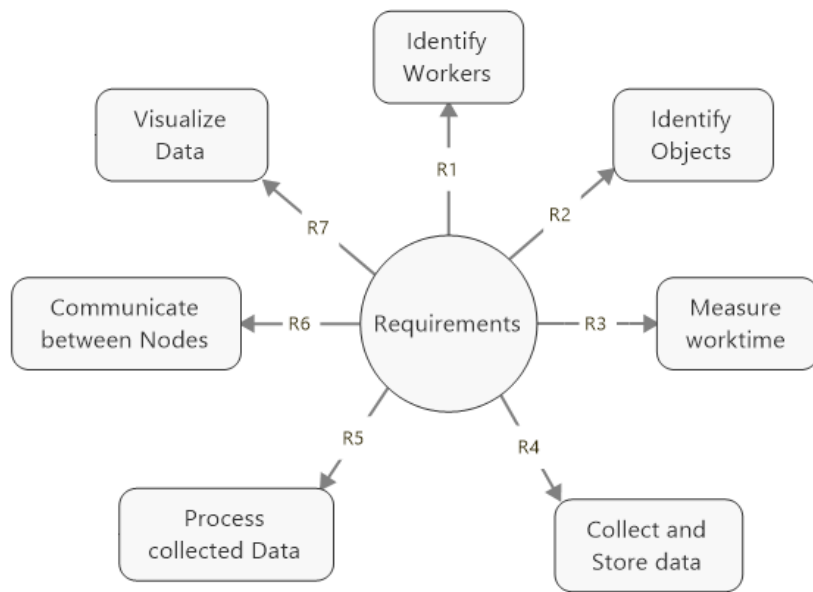


Figure 4.2: Requirements analysis addressing the challenges to the system.

The different requirements will be analyzed deeper in section 4.2 where we will also compare the different technologies. To introduce, let us briefly explain the different requirements and put them into perspective.

- R1-identify workers: Each worker must be uniquely identifiable by the system. This will assist with building statistics over user productivity. Knowing the real-time activity and status of each worker is viable for tracking an assembly line. There are a multitude of ways to identify Humans which range from cheap and quick like a "user-password"-combination to highly secure and customizable solutions such as an iris-scanner.
- R2-identify objects: Identifying objects means in this context to be able to identify parts, products, tools and other hardware required in the man-

ufacturing process. These objects can differ greatly in shape, size and portability. It is important to therefore find reproducible and flexible ways to recognize objects throughout the assembly-line.

- R3-measure work-time: The time a worker spends on a product, a task or which an object in question is dependent on factors which can not directly be known. It is vital though to track the assembly speed for each task and build statistics which enable to better understand the challenges of each assembly-line.
- R4-collect and store data: In order to electronically process data, it first needs to be acquired and stored in the system. This process should be automatized and cost the least amount of time possible. Each fluctuation in the acquiring of timestamps will bias the dataset.
- R5-process the collected data: The data which is stored in the system needs to be processed in real-time to be visualized for the assembly-line supervisors in time. This is taken over by an electronic component handling all the calculations of the system.
- R6-possibly communicate between nodes: Different nodes of the system in question need to be able to send data and triggers over a network to be defined. The less the latency and complexity of this network, the better the results.
- R7-Visualize Results: The computed results finally must be transformed into human understandable and readable output. There are different media in question of transferring information to a human worker. This depends on the complexity of the data and the capability of the worker. The latter is influenced by how busy the worker is or how susceptible he/she is to notice a change in information.

4.1.2 Concept options

There are different technologies at hand for solving each of these challenges. We want to provide a quick overview of what is considered for each challenge and later on explain why which technology was chosen.

Req-Description	Option 1	Option 2	Option 3	Option 4
R1-identify worker	Fingerprint sensor	RFID-Wristband	User password	iris-scan
R2-identify object	Bar-Code	RFID-Sticker	Shape recognition	Container sensors
R3-measure worktime	Physical button	RFID-Scan	stopwatch	Weight measurement
R4-collect data	PC-Workstation	RaspberryPi3	Smartphone/tablet	Arduino
R5-process data	Server	Workstation	RaspberryPi3	cloud service
R6-communication	Usb	Wi-Fi	Lan	Bluetooth
R7-Visualize data	Smartphone app	Html Page	Rendered Image	Windows App

Table 4.1: Technological options overview

4.2 Evaluation of Platforms



In the following subsections we will assess and compare the qualities of the technologies which are eligible for solving the key challenges. The selected technologies are presented more explicitly in Chapter 2.

4.2.1 R1: User identification technologies

The proposed options range from Simple user-input to cutting edge technology.

	Fingerprint sensor	RFID-Wristband	User password	iris-scan
Tampering resistance	++	++	+	-
Error margin	++	++	--	++
Time cost	++	++	-	+
Compatibility	+	++	++	-
Price	-	+	++	--
Total:	+6	+9	+2	-1
Rank:	2	1	3	4

Table 4.2: Comparison matrix for user identification technologies.

The most suitable option for identifying workers at their workstations is the RFID-Wristband which can be easily scanned contactless and with no further doing from the worker. The installation of a RFID-sensor at each workstation is at a low cost of less than €10 and directly compatible with the data-collection technologies of choice (section 4.2.4).

4.2.2 R2: Object identification technologies

Recognizing objects can not be done without further doing. Objects are more versatile in shape combinations than humans. This eliminates options such as fingerprint scanners and leaves us mostly with option which require the object to be marked.

	Bar-Code	RFID-Sticker	Shape recognition	Container sensors
Ease of use	+	++	-	++
Error margin	++	++	--	+
Compatibility	+	++	--	+
Price	+	++	--	-
Total:	+5	+8	-7	+3
Rank:	2	1	4	3

Table 4.3: Comparison matrix for object identification technologies.

In this section the best choice is again the RFID-technology nonetheless since it has already been chosen beforehand for the human identification. Using the same technology for different purposes saves time in deployment and software development lowers budget requirements and system complexity.

4.2.3 R3: Time measurement technologies

As the previous technologies are purely digital and based on electronic communication, the RFID-sensor technology is also the logical decision which will be used for measuring the times for executions, waiting and supply-operations of the assembly-line(s). The events registered when an RFID-tag is swiped will determine the time-window used for each operation. For operations which do not incorporate any tagged object or product, a physical button will be introduced within the Users workspace which he/she can operate to signalize the start and end of a timed-event.

4.2.4 R4: Data collection Platforms comparison

The comparison matrix in table 4.4 compares the key features which are of matter and assesses which platform has the optimal overall qualities.

	Raspberry Pi3	Arduino	Android Device	Windows Pc
Data security	++	+	-	+
Portability	+	+	++	--
Programming Ease	++	-	-	+
Connectivity	++	+	+	++
Hardware compatibility	++	+	-	+
Deployment Time	++	++	+	-
Price	+	++	-	-
Total:	+12	+7	0	+1
Rank:	1	2	4	3

Table 4.4: Comparison matrix for data collection platforms.

The Raspberry Pi 3 device offers many possibilities which are partly offered by other platforms too but the small form factor clearly beats the other options when comparing compatibility and functionality/size. One could argue that the adruino board is a good choice either but the biggest factor separating both choices is the native and straightforward Windows-IoT support of The Raspberry-pi3 board.

4.2.5 R5: Data processing Platforms comparison

Since this thesis is a time-limited work, the tests which are performed are not exhaustive and can in a real-world use-case scale up quickly in the case that

the network of workstations and products assembled is increased to shop floor size. The requirement of data processing is therefore scalable and can for small assembly lines like ours be overtaken by a workstation or even a raspberry Pi3 device. For real implementations on large assembly lines a powerful server is recommended for processing all the real-time information to keep the system run smoothly. For our tests we will use a Pc-workstation as the processing back-end for reasons of stability and control. The system shall not be bottlenecked by a slow back-end processor. Using a stronger unit as server allows us to put the workstation devices under full load. This is required for stability testing and benchmarking.

4.2.6 R6: Data communication medium between system nodes(here: rbpi3s)

	Usb	Wi-Fi	Lan	bluetooth
Portability	- -	++	-	+
Programming Ease	+	++	++	- -
Hardware compatibility	++	+	++	-
Deployment Time	- -	++	-	+
Price	-	++	-	+
Total:	-2	+9	+1	0
Rank:	4	1	2	3

Table 4.5: Comparison matrix for communication solution.

4.2.7 R7: Visualization of the calculation and measurement results

The Visualization can technically be done on all the proposed ways. Due to limited development time we only have the possibility to explore one concept deeper which will ultimately be the most compatible one which is the HTML page. The biggest advantage of HTML is the seamless integration possibilities for other technologies. To build a graph for example there is not one special library which is required but a vast amount of possible choices which all work out of the box through PHP or JavaScript and even HTML5 with CSS-scripts. Those are more flexible than a given add-on software which could be used within apps for example. We have this requirement since at the earlier stages of development it was still unclear which graphs and means of representing our results would be required and therefore did not want to be limited later on due to software compatibility-issues. Also the results should be scalable and quick to refresh. Rendering images puts a processor under high-load and causes imbalances. This drawback excludes the option to render figures each time an update to the state of the system has occurred. Such a change in state can potentially happen in second-wise intervals. HTML-pages are also very portable, meaning the size of the file itself is negligible and will load in very lightweight environments such as browsers of portable devices.

	Smartphone app	HTML Page	Rendered Image	Windows App
Portability	+	++	++	-
Programming Ease	-	++	+	+
Compatibility	-	++	++	+
interactivity	+	++	- -	++
Total:	0	+8	+3	+3
Rank:	2	1	3	3

Table 4.6: Comparison matrix for visualization software.

4.3 Evaluation of the target Solution



Resulting from the technology comparisons performed under section 4.2 the chosen option is "Option 2" from table 4.1. This choice consists of the following solution combination:

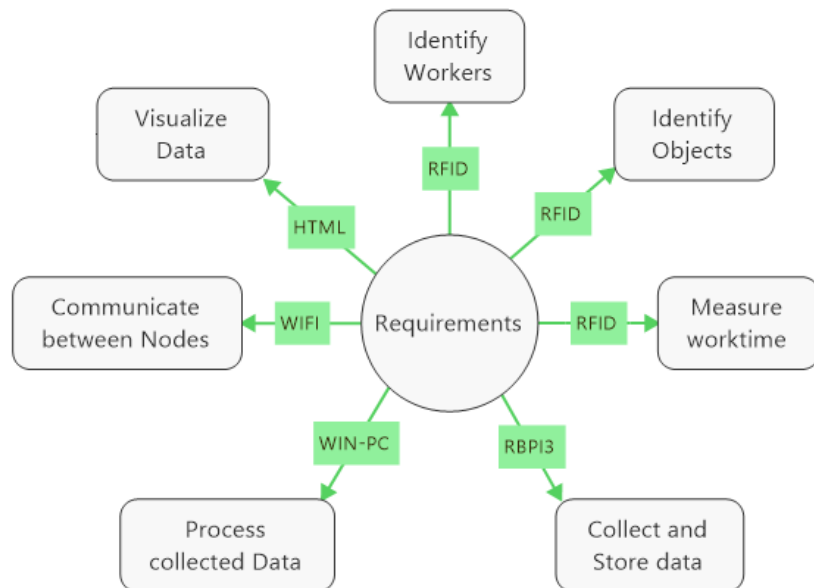


Figure 4.3: The previous open challenges in Figure 4.2 have been replaced with technological solutions.

- Explanations RFID: Section 2.1,
- Explanations RBPI3: Section 2.2,

- Explanations WIN-PC: Footnote ¹,
- Explanations Wi-Fi: Section 2.4,
- Explanations HTML: Footnote²,

4.3.1 Visualization through use-case model

To further explain the functioning of the system in development, the interactions with the implemented technologies are explained in this section. It is important to clearly state which elements communicate with each others, what the passed informations are and what the resulting task from the interaction is.

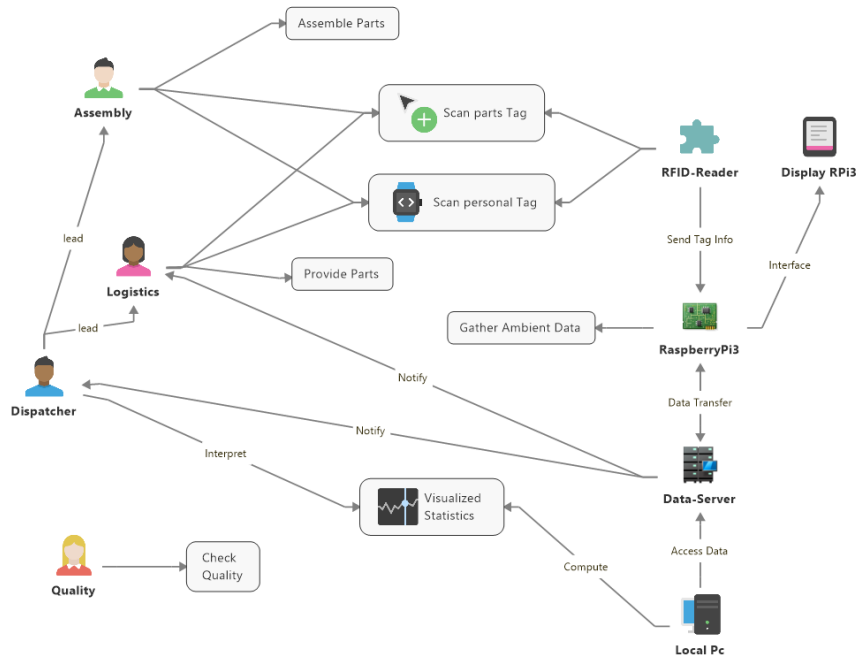


Figure 4.4: A specific use case showing the chosen technologies and typical user-interactions

Figure 4.3.1 shows the use-case model for the different types of workers with the new implementation of the assembly line. On the left hand side of figure 4.4

¹Under Windows Pc we mean a Workstation with a stronger processor and multiple cores which can handle server-applications without bottlenecking and store a set of data in a mysql database. The workstation Runs the Windows 7,8.1 or 10 operating system and supports the written applications which we design for our purposes as back-end server.

²HTML-Hypertext Markup Language is the most common web-development language. Nearly every web-page is based on HTML and then further expanded with more applied scripts. More details under <https://en.wikipedia.org/wiki/HTML>

the different workers are listed and linked to their tasks which are in the middle lane. The tasks have a back-end which is formed of digital communication to and within the devices such as RFID-Reader, RbPi and server(s).

4.3.2 The updated assembly line structure

In the Assembly line how it is imagined for the future 4.5 the drawbacks presented above are addressed. Mainly the information flow rate is now unbuffered and direct which leads to a number of consequences.

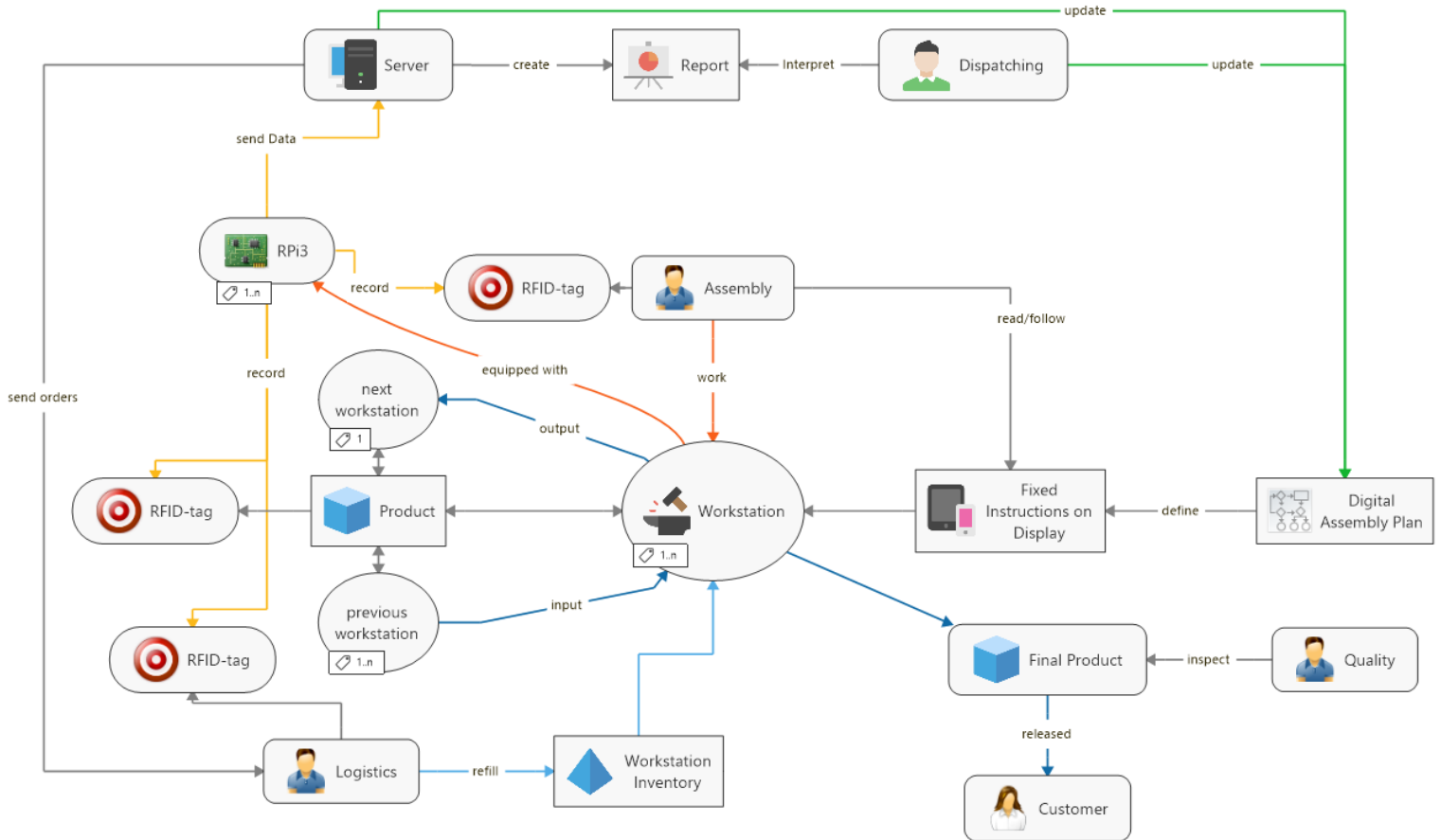


Figure 4.5: An assembly line how it ideally is interconnected after we deployed our infrastructure layer.

First of all by inspecting the schema the addition of actors is noted. There is now a server which stores all the data about the plant and sends triggers for new assembly pushes. In a traditional plant a server stores certain information

about the manufacturing process as well as basic manuals, while in our case the server is in direct communication to the modules installed in each workstation. Those modules are composed by a RbPi chip, an RFID-reader and a portable touchscreen. The workstations report to the activities performed to the server which then evaluates those and combines the input into overall statistics.

It is important to note that actors which have existed also in the traditional setup, are extended with RFID-chips which identify their identity and purpose. Depending on which location a Chip is scanned, different scheduled actions can be triggered, checked and closed. This also allows for errors to be detected and corrected immediately.

The assembly manuals and instruction sets are now generated electronically from the server side based on decisions taken by a new class of worker called the dispatcher(s). Their main task is to overlook the feedback gathered in the different views and make predictions on this basis about what to change in the production chain. Those alternations can immediately be pushed and implemented on the go by simply updating the information shown to the workers for each workstation.

Some parts of the plant can even be automated completely like the inventory management. Since each part is traceable now, it is possible to control the inventories for each workstation in real time. This means if the inventory falls below a certain threshold, it can immediately ask to be refilled by a worker assigned to logistics department. This can enable scheduling milk-man runs on a timed basis and requires less effort in managing the supply-chain.

Chapter 5

Implementation

A big part of the work performed under this thesis consists of software design and coding. All the coding work is later executed in the background and not accessible to the end-user(s). The functionalities of the software will be accessible through interfaces which are written as earlier stated in CSHTML which is a language allowing dynamic HTML pages as part of a c# software. These webpages will be called views in the later documentation. There are different views for different objectives.

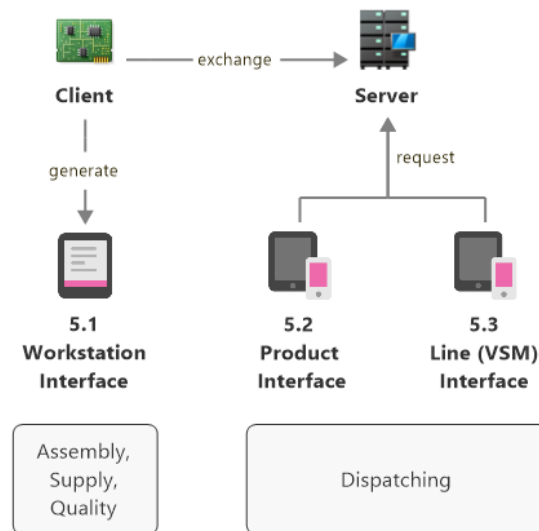


Figure 5.1: Interfaces of this software and their paths of generation.

These 3 Views are:

- Chapter 5.1 An interactive View showed for each workstation which enables the tracking of the assemblies.
- Chapter 5.2 A View generated by the server which shows live information about a given product which is scheduled for assembly, in the making or finished production. This view is shown on a big monitor for a dispatcher to consult for optimization of the line.
- Chapter 5.3 This View is also generated by the server and holds information on the overall status of the assembly line. The view is updated on refresh and shows a dynamic VSM [8] (Value stream mapping) holding only the key indicators which reflect the performance and health of the assembly line.

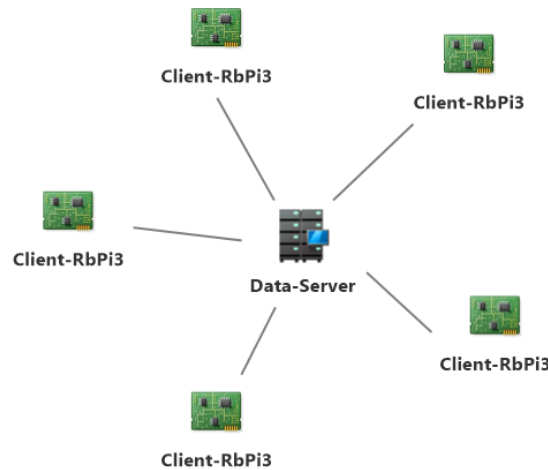


Figure 5.2: The system architecture following the Client-Server-model.

The structure of the architecture follows the client-server model. This means that the server(s) hold the data and provide it upon request to the different devices. For example can the Views for dispatching be requested like a normal web-page via HTML protocol and also be rendered on any device which has a web-browser which supports HTML. The Interactive view for the workstation which provides the instructions can only be rendered on the raspberry-pi3 devices equipped with the required back-end. This is done by design in order to use less network traffic and only transmit small packages of data. By minimizing the amount of data sent, it is possible to keep track of the line in real-time and deliver information about the current state of each workstation and assembly with the least delay to the server.

This choice of architecture also is more secure and tampering-resistant since smaller packages can be encrypted with more ease. The here presented version of the software does not feature traffic encryption, yet but an additional layer of ssl would not cause any challenge to the implementation.

5.1 Workstation Interface

Each workstation which performs tasks like assembly, logistics or quality assurance is equipped with a 7" touchscreen as shown in figure 5.3. The touch screen is directly connected to the RbPis and also draws the power from the portable devices.

The interface for the mentioned workstations is as shown in figure 5.4. This interface will be loaded on the RbPis and be shown on the touch-displays. The main purpose is to clearly visualize the next tasks and track the status of the workstation and the current product assemblies which are open. Therefore this screen is optimized for touch actions but should require as little interaction as possible and display only the non-intuitive part of the tasks. Those instructions aid the worker without binding him to the screen with overwhelmingly long and detailed instructions. Each task is simple and logically bound to the previous one. This view is updated in real-time while the product transitions through the workstations. This is possible because each part of the assembly is tracked and scanned when used by the workers.



Figure 5.3: The 7 inch touchscreen display, its back-plate and mounting components.

Figure 5.4 shows the concerned interface in its initial state, the "login" screen

waiting for a user to identify him/herself by scanning a personal RFID-Tag. Once logged in everything is controlled by User action meaning when an RFID-tag is scanned. The next 3 tasks are shown with steps which can be done in a certain order but don't have to be followed strictly. For example a task can be to assemble 3 parts together. The steps for this task are first to pick up each part and then to put them together following the manufacturing guideline(s). Each Item will be scanned on pickup and the step will be marked as finished once the resulting assembly is scanned. Then the screen will automatically move on to the next task of this workstation which usually means the next product to work on.

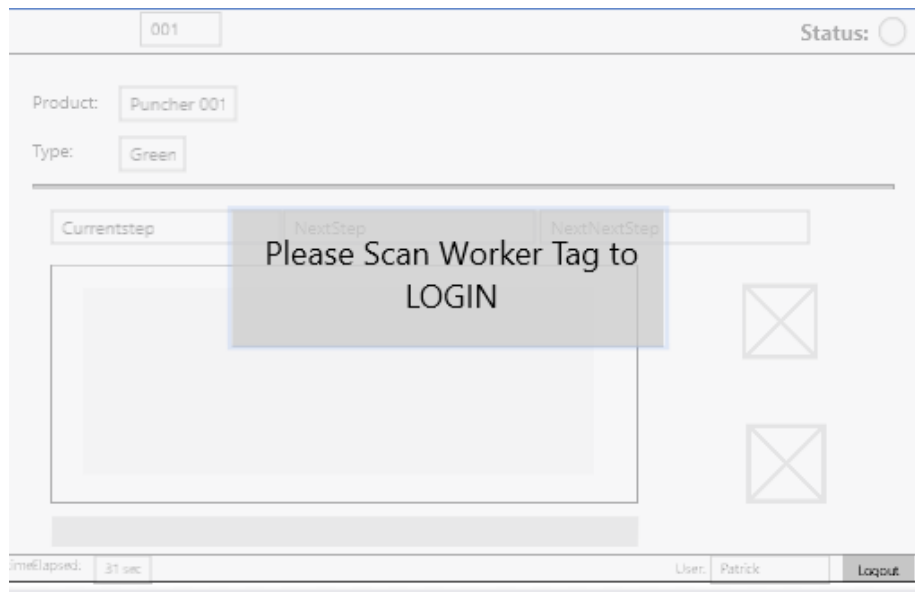


Figure 5.4: The Interface for each workstation touchscreen

There are different possible ways to register that the part is correctly assembled. One method is to define a key-part of the assembly at the beginning which could be one of the larger, easy to access parts. In our case of the puncher, this is the bottom plate. Each time when the bottom plate is scanned, this would mean that the task is now complete. Another way of handling this which allows for even more control is to use a scale at the input and output buffer. This can register if an element has been removed or added to a box/buffer. Furthermore we can determine which part has been picked up or placed by knowing the weight difference and weight of each part. This allows us to crosscheck if all the parts have correctly been arranged into the product/sub-assembly. Certainly it is an easy implementation to add one or more scales to constantly measure the weight of the input and output bin/buffer for each workstation. A partial implementation would be to place a scale under the display of the workstation and just use this one directly to measure the weight. This would help with keeping the "workstation-computer" more compact and accessible.

For each step there will be two possible images shown on the right hand side

of the screen. Those by standard feature a view of the part concerned and the product in its state after assembly.

On the top there will be a list of upcoming tasks. On the bar below, the progress through the steps of the current task will be shown in form of a progress bar.

The smart workstations are a lightweight way to track the assembly. The data gathered is purely analytical and aims to make best use of each workers abilities and time rather than trying to evaluate whether a worker is efficient or not. Everybody has different abilities, strengths and weaknesses. By knowing those one can identify the best workplace for each worker and provide useful help with the manufacturing process.

5.1.1 Exemplary run of an assembly workstation

To best explain the user-interaction with the workstation interface let us go step by step through the sequence diagrams and perform exemplary tasks. The covered examples include a typical run of an assembly workstation, supply-chain workstation and quality assurance workstation. Each state will be described with a screenshot and the possible outcomes discussed.

Figure 5.5 shows a sequence diagram for an assembly-workstation. The example consists of a worker logging in on a workstation, performing a task which consists of picking up an object, assembling it and then finishing the task and logging him/herself out of the workstation.

Workstation - Interaction Example

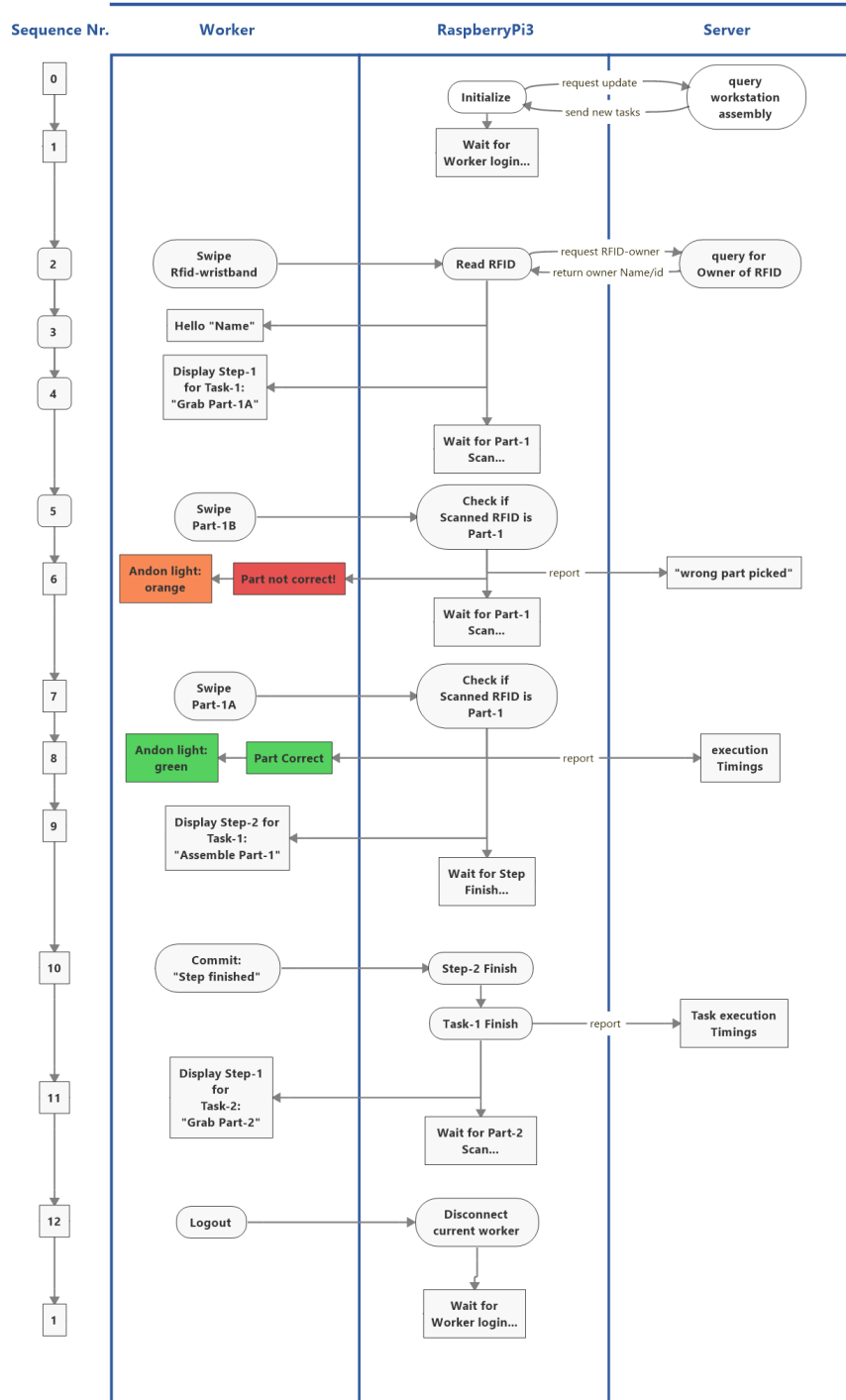


Figure 5.5: Sequence diagram simulating

State 0

The workstation in question has just booted up and first starts to initialize itself. The initialization consists of requesting an update of the assembly instructions from the server and synchronizing the time for future measurements.



Figure 5.6: Sequence diagram state 0

State 1

After the update is complete the workstation device will wait for a user/worker to log him/herself in by swiping their RFID-wristband. This is the Ready state of the workstation. The assembly steps and instructions are all loaded in the background.

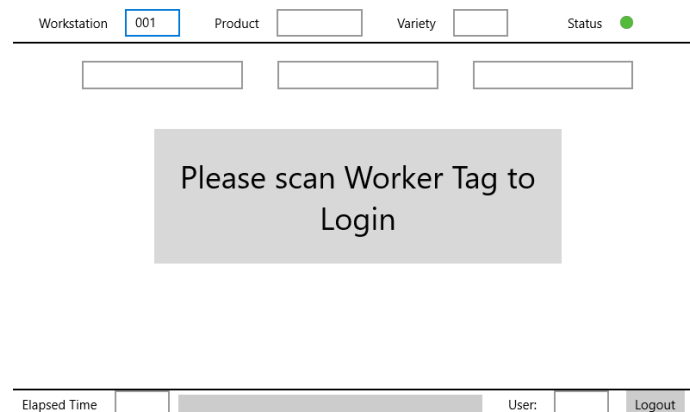


Figure 5.7: Sequence diagram state 1

State 2

Upon the worker swiping their RFID-wristband, the RbPi will read the content and make a request to the server to receive information about the identity of the worker. The server will make a query and respond with the information. If another RFID tag is scanned which does not belong to any worker of this shift, nothing will happen.

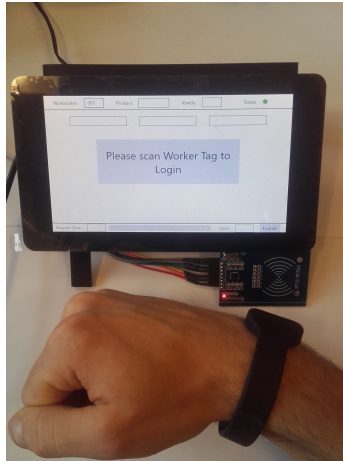


Figure 5.8: Sequence diagram state 2

State 3

Upon successful login, the workstation sends a "hello" to notify that the worker tag has been successfully registered. This message stays for about 1.5 seconds.

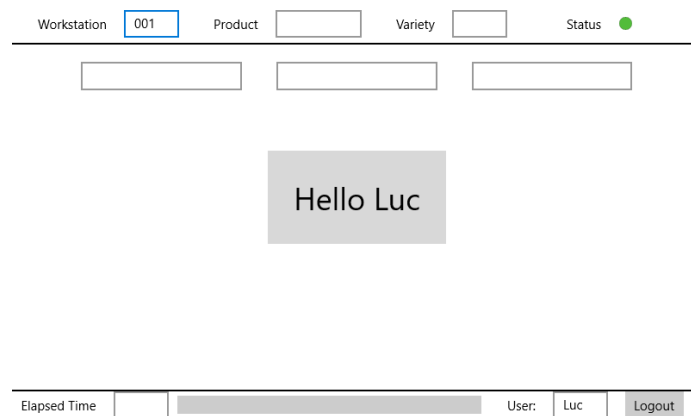


Figure 5.9: Sequence diagram state 3

State 4

After the "Hello"-message the work can begin. The interface will show the first step available in the queue for this workstation. Technically this step will be part of a task which is performed on a specific product. The next two steps are also visible for orientation so a fast worker can skip the instructions and work according to the header showing the next steps. For the step in the example, the worker is asked to take Part number 1A. The workstation will wait for an object-tag to be scanned.

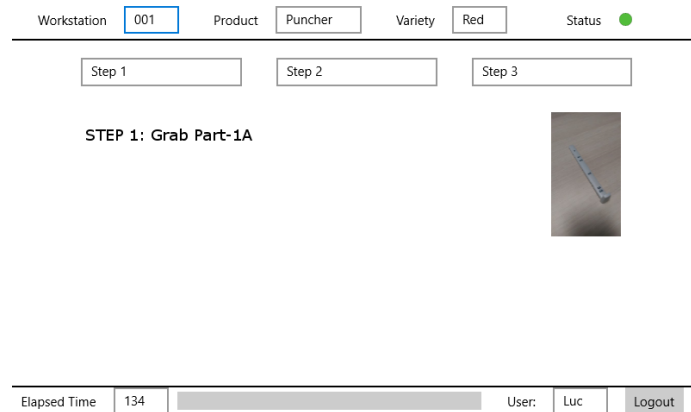


Figure 5.10: Sequence diagram state 4

State 5

The worker takes part number 1B and scans this Object. The RbPi will record which part was scanned and compare if this part is of the type which was required for this operation.



Figure 5.11: Sequence diagram state 5

State 6

The workstation-device discovered that the part of type 1B which was scanned was not the right one and rejects the step, by displaying a red warning feedback to the worker which indicates to take the right part, in this case 1A. Furthermore the workstation reports to the server that a wrong part has been scanned and a mistake was made. This will help with error-proofing later and help to build statistics on which parts are often mistaken or steps which are more prone to assembly mistakes. The workstation will wait for an object-tag to be scanned. As a consequence of the error, the Workstation is technically in still stand until recovery of the error. This will be shown by an orange Andon-light[5].



Figure 5.12: Sequence diagram state 6

State 7

The worker scans a part of type 1A on the workstation. The workstation will again check if this is the right type.



Figure 5.13: Sequence diagram state 7

State 8

This time the result of the check is positive indicating that the right part was picked for the step and the feedback on the screen will be green. This symbolized that the action was correct and the process will go on.

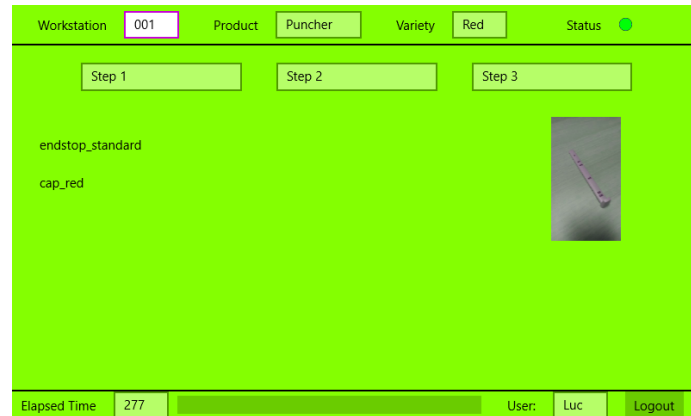


Figure 5.14: Sequence diagram state 8

State 9

After giving positive feedback, the interface will move on to display the next step in the queue which in this case is to perform a given assembly task on the part of type 1A. The workstation will wait for further user-input signaling that the step has been completed. The previously set Andon[5] light will be turned green again.

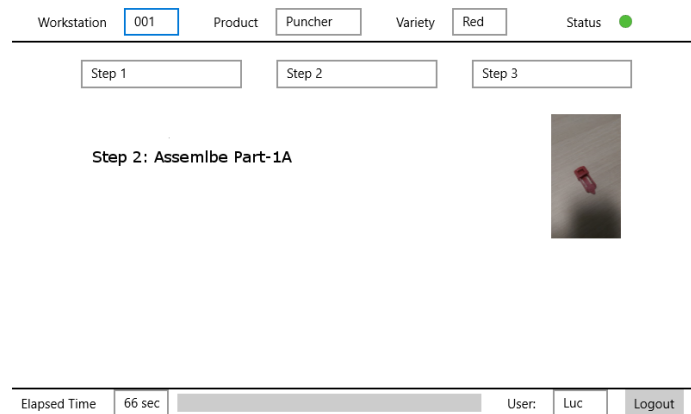


Figure 5.15: Sequence diagram state 9

State 10

As the worker finishes his work on step number 2, he/she will push a button on the interface signaling that the task is finished. In further development the target will be to solve this with a scan but for now we request a touchscreen feedback for simplicity and less error-margin during testing. The workstation will internally finish the step, check if this one was the last one for this task and if so, also finish the task on the current product. Since the task is finished, feedback on execution time is sent to the server for further calculations.



Figure 5.16: Sequence diagram state 10

State 11

Since the previous task has been finished, the workstation moves over to the next task in line which generally means the work on the next product also. Therefore now the step number 1 of task number 2 is shown to the worker. The workstation waits again for an object-tag to be scanned.

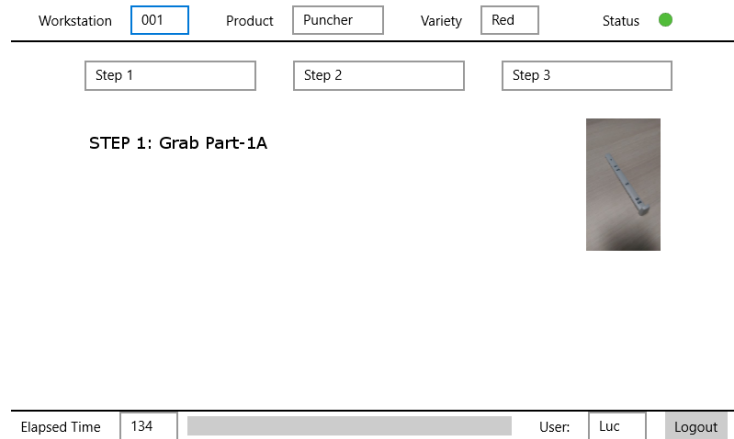


Figure 5.17: Sequence diagram state 11

State 12

Now the worker could continue the assembly process but in this special case, decided to log out from the workstation by pushing a "logout" button on the bottom right of the screen. It can be stated that the shift of the worker has finished for example. This action will put the assembly process where it was on hold and wait for a new worker to login on the workstation and continue where the last one has stopped.

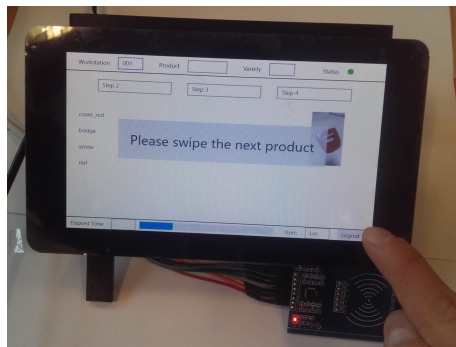


Figure 5.18: Sequence diagram state 12

5.1.2 Exemplary run of a supply-chain workstation

Figure 5.19 shows a sequence diagram for this case. The example consists of a worker logging in on a workstation, performing a task which consists of retrieving an object, delivering it to a given destination (a workstation) and returning back to his/her home-workstation. Finally the worker will log out from the workstation again.

Workstation 001 - Supply-chain Example

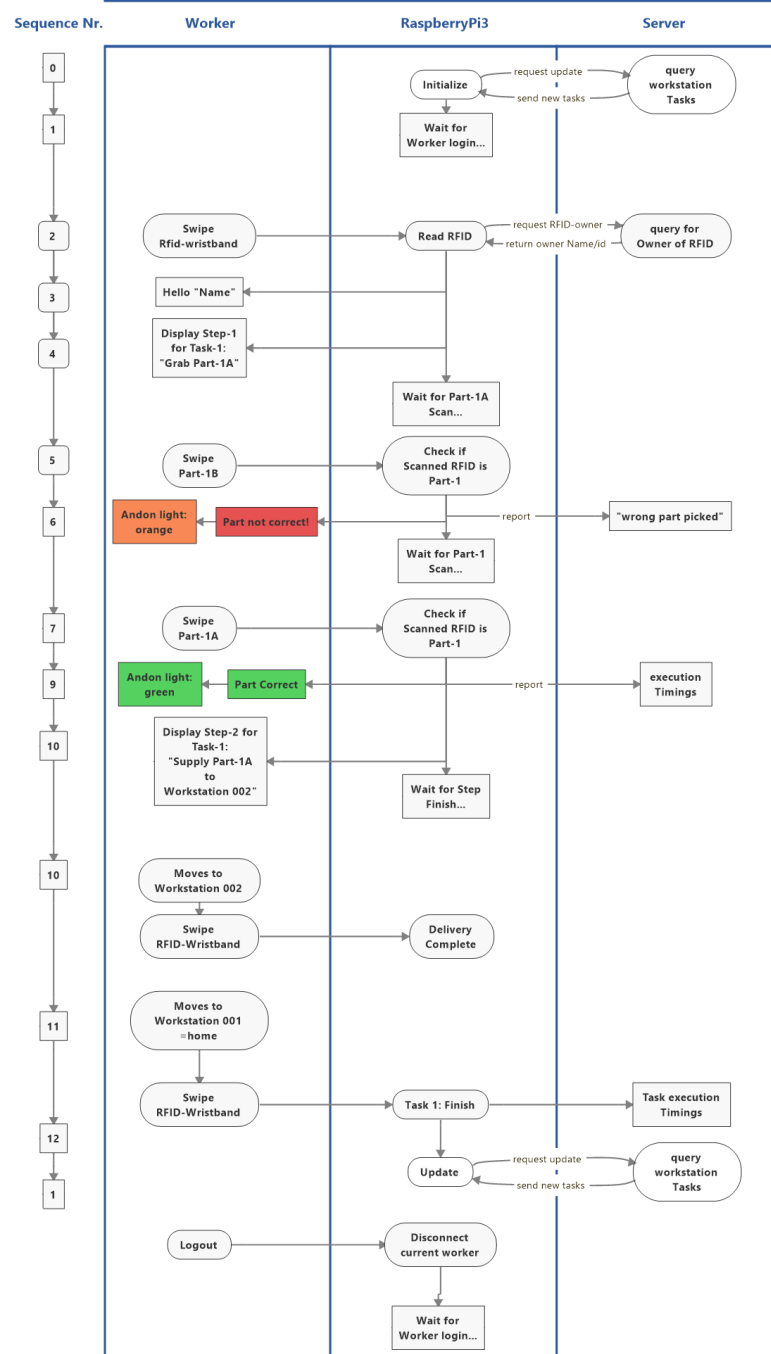


Figure 5.19: Sequence diagram state 1

States 0-12

The supply-chain interactions are not much different from those of an assembly workstation. The run is quite distinctive but can be interpreted the same way as the exemplary sequence diagram for the assembly station as there are no uniquely new features or actions which arise. Therefore the different states of the supply-chain workstation are not explained as elaborately as those of the assembly case. The steps which mainly differ here are the ones which request for interaction with the receiving workstation. The action which is used to measure successful delivery of the payload is performed under State 10 in the sequence diagram (Figure 5.19). The worker which is assigned the task scans his RFID-wristband at the workstation to signalize a successful delivery of the packet.

5.1.3 Interactions between different workstations

As is one workstation creates a sub-assembly from a set of parts. These parts can be sub-assemblies themselves and originate from earlier workstations of the production chain. The sub-assemblies created are then passed themselves to the next workstation which will add value to them.

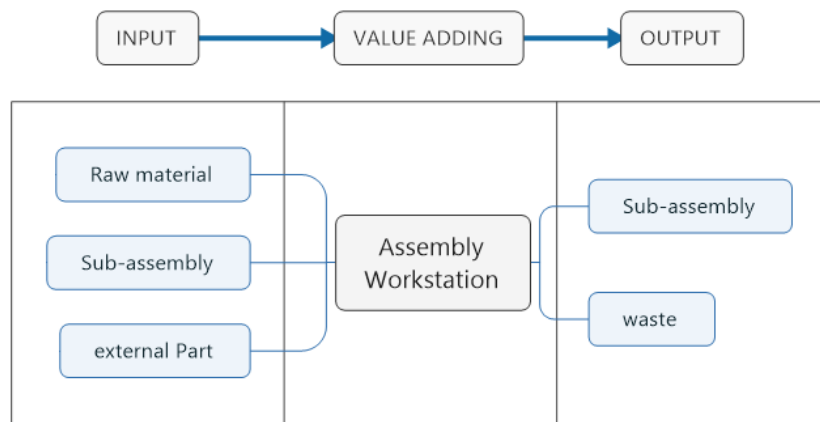


Figure 5.20: Product flow of the workstations.

The output of a workstation can either be a subassembly, waste or both. If a product is not assembled or handled correctly it can occur in some cases that the result is a faulty subassembly which can no longer be used for the final product as such. This will then be waste. As much as this thesis tries to fight waste by detecting wrongly executed tasks, the assumption that no errors are performed is still not viable.

5.2 Dispatching-Interface: Product

Apart from the workstation displaying tailored assembly information to each workstation there are dispatchers which are workers dedicated to maintaining good flow and progress of the overall plant processes. They can change the way the plant operates and take decisions based on what the workstations record. For this purpose there are for now two Views which are generated on the server-side and shown on two big screens for everybody else to see.

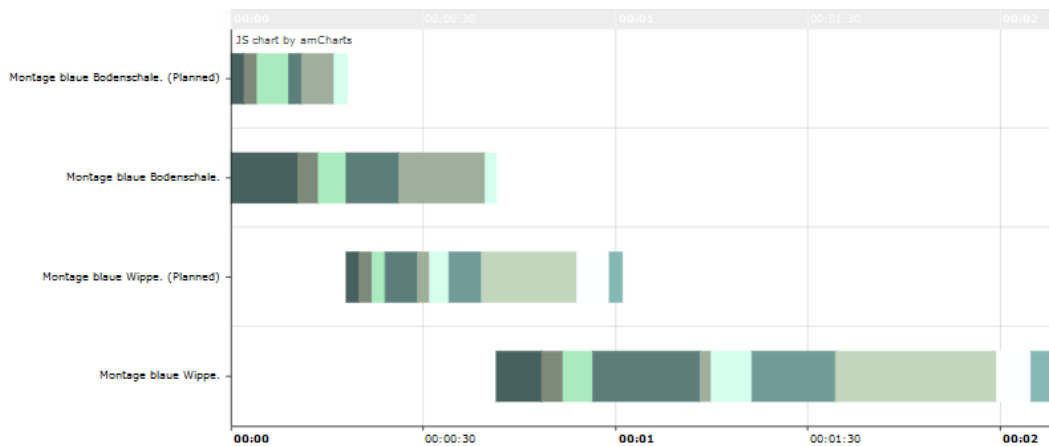
One of those is the "Product-View" which consists of showing a time-line in the form of a Gantt Chart representing a real-time assembly for a chosen product when it is in planning, being assembled or after its assembly is completed.

Figure 5.21 shows this View with a generic example. After each assembly Task, there is a certain time which is planned for the part to go through the buffer between stations or ideally a certain waiting time which has to be accounted for when the part is in transit from one station to the next one. The better the line is planned and operated, the smaller this waiting time will be. There are two cases of waiting time. The first one is if the previous station can not finish its task in time and the following one is waiting for this resource. The second case is if the previous station finishes in time but the following one is still busy processing other tasks and can not immediately begin working on the newly arrived part.

In the Gantt chart it will be clear to see which station is the source of the waiting time in each case since there is the possibility to crosscheck the actual task-time with the planned one. Now there is the possibility to try to speed up the assembly of the bottlenecking Task(s) but it is more efficient and quality maintaining to reschedule the assembly line and change the Takt-times. This can be done by parallelizing some workstations/processes which are time consuming or redistribute the tasks to workstations with less load. The columns can be toggled to refine the view depending on which is the current target of optimization.

Product view Puncher (Blue)

Puncher (4)
Running time: 202 seconds



© 2018 - My ASP.NET Application

Figure 5.21: The View of the product assembly Interface

Each row of the Gantt chart in Figure 5.21 shows on Task run. This can either be a scheduled task which shows the ideal times for the task or the measured execution times from the workstations. Typically the planned task is shown first, followed by the execution data for comparison. In the mentioned figure can also be seen that each task is subdivided into sections of different colors. Each section represents a measured action, which is called step (atomic) in this thesis. Those steps belong to the task and must be executed within the given order. This chart is interactive which means hovering over steps will show their individual title and times in milliseconds. Other information can also be shown such as the number errors which have occurred before the successful completion of the step.

5.3 Dispatching-Interface: Line (VSM)

Figure 5.22 is designed to partially represent a Value Stream Map(VSM). A VSM depicts the general outlay of a plant showing each workstation, the work-

ers, management and flow of parts. Only the fields which are the most useful are overtaken from the classical VSM design. It is possible to implement a complete version but the benefit would be mild compared to the existing version. Most important are the fields showing the workstation buffers as well as statistics on each workstation like uptime and errors/time for example.

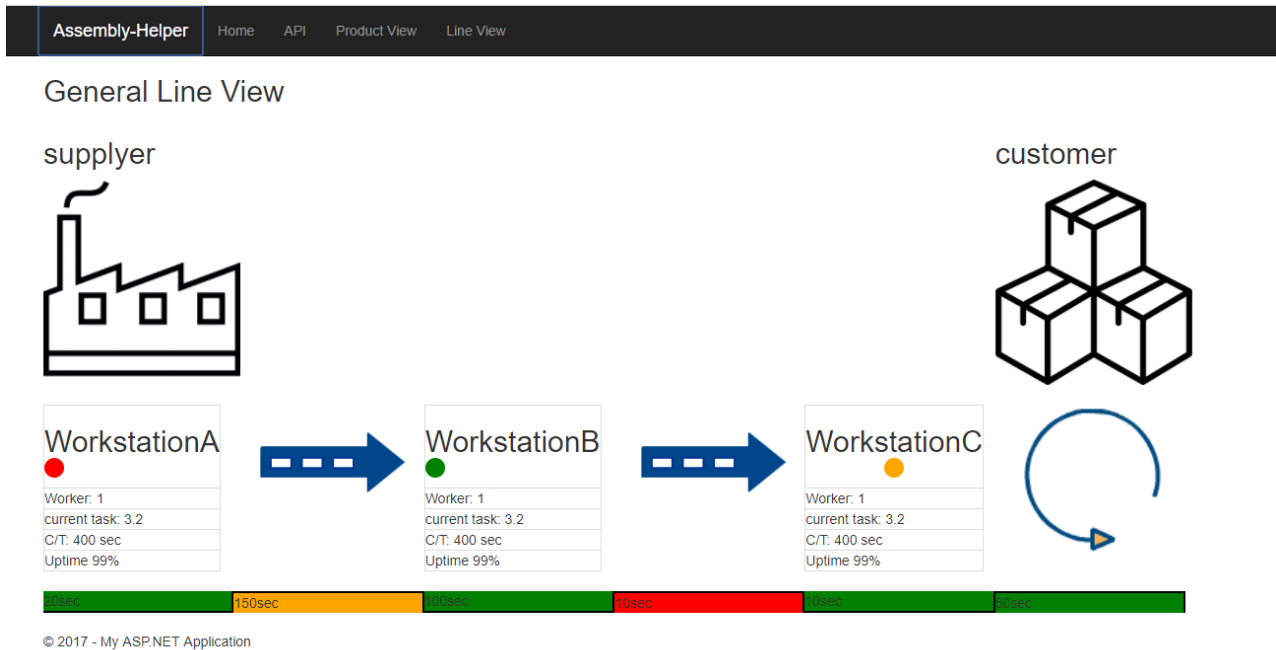


Figure 5.22: The real-time assembly-line status Overview

From right to left are the workstations from start to end of the product assembly. Each workstation list shows on top the name and number of the station, followed by an Andon[5] indicator which is mirrored with the Andon[5] light of each workstation interface. This helps to see the status which can be green (all ok), Orange (in waiting) or red if an error persists on a station.

Under the title is a small list with the core performance indicators of the given workstation. Those can be varied based on experience. Recommended are for example:

- Worker ID/Name currently logged in.
- The Task/product currently being performed and its progress in
- The error rate of this station.
- The uptime of the station for the current shift which is defined by 100% - waiting time/working time.

This list should not be overloaded to retain practicality. 4-5 items are sufficient to gain a basic awareness of the health of the workstation.

As the parts and unfinished products are passed from station to station, there are number of different types of ways to organize this transfer. In the current solution there are buffers in place. Those are depicted in the VSM-view with a blue arrow or cases showing the maximum size of this buffer. The smaller, the better but this also increases the waiting time.

The last indicator on the bottom of the Graph represents the time-scale followed by the assembly. Under each Workstation the average time to execute the task(s) of this workstation is shown in seconds or minutes. Between the workstation in the buffer-zone is shown the time which the parts wait in the buffer before they are accessed by the next workstation to be further worked on.

Chapter 6

Experimental Validation

This Chapter will explain the experimental setup, the testing methodology and then present the outcomes of the test runs. The visualizations are screenshots taken during and after the execution of the tests. These tests will be the base for further discussion and improvements to the system.

6.1 Experimental Setup

The experimental setup is inspired by the already running education and test assembly line called "lean lab" situated at university of Luxembourg. The advantages of this existing educational assembly line is that we can mostly base our tests on the already in use instruction-set, product-parts and Setup.

The example assembly is to produce a puncher. This puncher can be manufactured in 3 different colors: blue, red and black.



Figure 6.1: The product assembled during the test runs.

The complete assembly line for this puncher has 7 stations of which we cover stations 2-4. These essentially cover the production steps of the product. One

station will be black-boxed as it requires special tools which were not in the optimal condition and scheduled for maintenance at the time where our tests were performed. This allowed us to target other, more simple stations and also reduce the complexity of our test. The other Stations left away were dedicated to disassembly.

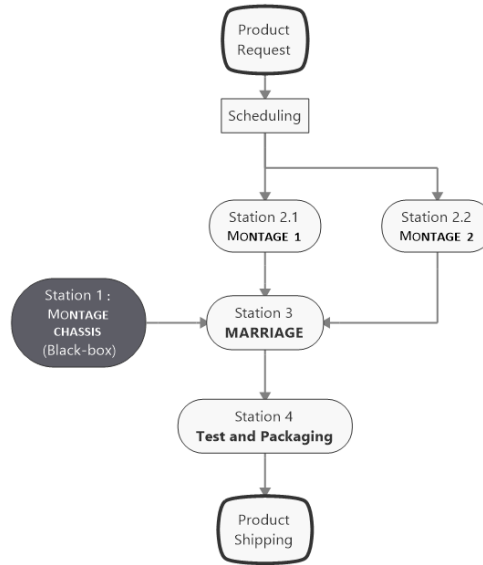


Figure 6.2: The layout of the Assembly line Stations.

Figure 6.2 shows the assembly steps followed for our test runs with each station having one worker assigned. The original Assembly instructions will be transcribed into the new database and be shown together with the images of the parts on the displays during assembly.

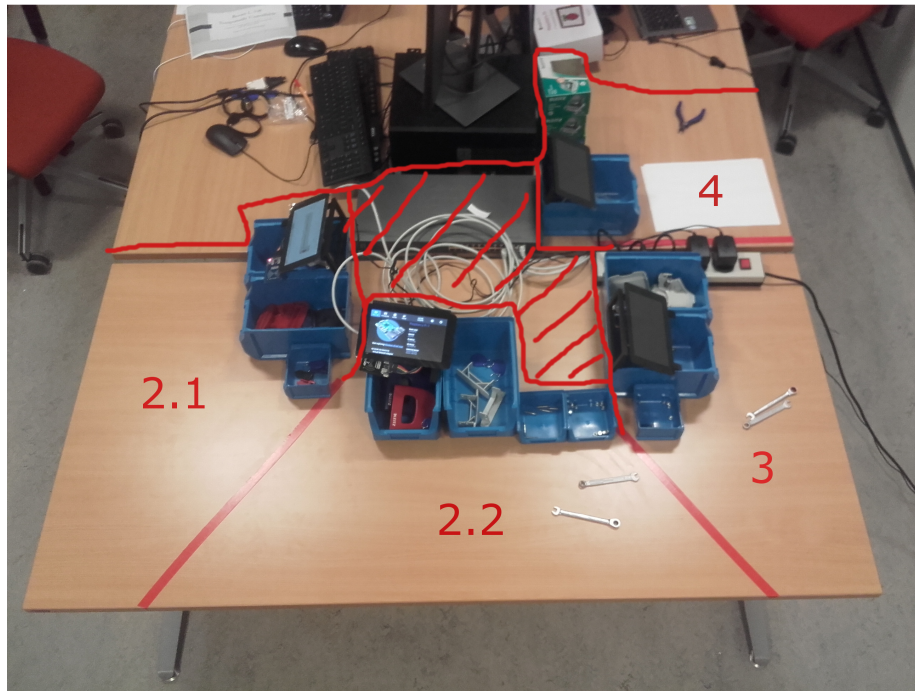


Figure 6.3: Experimental Setup of the Assembly line Stations.

6.1.1 Assembly Stations

The assembly instruction for the stations are in German and got translated to English for this thesis. Since the old instruction sheets used in the Lean-Lab[6] were already formatted and the adapted English ones only exist in our database, we will use the formatted German version to give an overview of the planned assembly steps.

Station 2.1-2.2


STATION 2		MONTAGE AUFBAU		uni.lu UNIVERSITÄT DU LUXEMBOURG		WORK ELEMENT SHEET	
		Symbol: ▽ Kontrolle ● Tricky ◆ Qualität					
Blatt 2	Stückliste: 1 x Wippe 4 x Mutter 1 x Raststück 1 x Formatschiene 1 x Bodenschale 2 x Stift						
Arbeitsnummer	Symbol	Beschreibung des Arbeitsvorgangs		Fertigungshilfsmittel		Zeit t [s]	
2.1		Einsetzen der Plastikblende in die Wippe, sodass die runden Ecken der Plastikblende die Wippe nicht berühren				5	
2.2		Einsetzen der beiden Stifte in die Wippe durch Loch 1 und Loch 3				5	
2.3	●	Sichern der beiden Stifte mit den Muttern (erst die Mutter an Loch 3 anschrauben und dann die Mutter an Loch 1)		Schraubenschlüssel		15	
2.4	▽ ◆	Beide Muttern nachziehen				5	
2.5		Weitergabe der Wippe an Station 3				2	
2.6	▽ ◆	gleichfarbige Bodenschale zur Wippe nehmen				2	
2.7	▽ ◆	gleichfarbiges Raststück zur Bodenschale				2	
2.8		Raststück (ins rechte Loch) der Bodenschale legen, bis zum Anschlag				5	
2.9		Formatschiene durch das Raststück in die Bodenschale schieben				5	
2.10		Weitergabe der Bodenschale an Station 3				2	
Fotos:							
							

Figure 6.4: Instruction Set for Station 2.1 and 2.2

Station 3





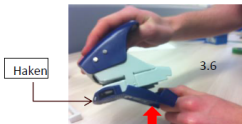


STATION 3		HOCHZEIT		<div> UNIVERSITÄT ZÜRICH FABRIK</div> <div>WORK ELEMENT SHEET</div>		
		Symbol: ▽ Kontrolle ● Tricky ◆ Qualität				
Blatt 2		Stückliste: 1 x Chassis (vormontiert) 2 x Mutter 1 x Wippe (vormontiert) 1 x Wippenstift 1 x Bodenschale (vormontiert)	 			
Arbeitsnummer	Symbol	Beschreibung des Arbeitsvorgangs		Fertigungshilfsmittel	Zeit t [s]	
3.1	●	Aufsetzen der Wippe auf Grundgestell mit Druck, sodass die Wippenstifte unter den Haken einfahren (leicht an der Plastikblende ziehen)			15	
3.2	▽ ◆	Wippenstifte sollen sich in der Mitte bzw. auf den Lochstiften befinden			2	
3.3		Einsetzen des Wippenstifts zur Fixierung der Wippe und des Grundgestells; Wippenstift durch das Loch 2 der Wippe			10	
3.4		Sichern des Wippenstiftes mit Hilfe von Muttern		Schraubenschlüssel 	15	
3.5	▽ ◆	Kontrolle, ob Bodenschale und Wippe die gleiche Farbe besitzen			2	
3.6		Befestigung der Bodenschale am Grundgestell durch einfaches Drücken			20	
3.7	◆	Bodenschale muss sich über dem kleinen Haken des Grundgestells befinden			1	
3.8		Weitergabe an Station 4			2	
Fotos:						
<div></div>						

Figure 6.5: Instruction Set for Station 3.

Station 4



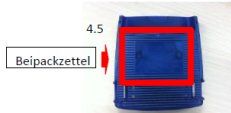


STATION 4		TEST UND VERPACKUNG			 WORK ELEMENT SHEET	
		Symbol: ▽ Kontrolle ● Tricky ◆ Qualität				
Blatt 2		Stückliste: 1 x Locher (vormontiert) 1 x Spannband 1 x Beipackzettel 1 x Verpackung				
Arbeitsnummer	Symbol	Beschreibung des Arbeitsvorgangs			Fertigungshilfsmittel	Zeit t [s]
4.1	▽	Kontrolle, ob das Raststück sich auf der richtigen Seite befindet				2
4.2	▽ ◆	Test des Lochers, indem man ein Blatt lochert und den Abstand mit dem des Teststreifens vergleicht				10
4.3		Falls der Locher fehlerfrei ist, Punkte 4.5 - 4.11 ausführen				-
4.4		Falls der Locher Fehler aufweist, Weitergabe ans Lager				7
4.5	◆	Abfall aus der Bodenschale entnehmen				15
4.6		Zusammendrücken des Lochers und Anbringen des Spannbands				10
4.7		Beipackzettel unter Spannband fixieren (Bodenschale Rechteck)				5
4.8		Locher in Verpackung schieben				5
4.9	▽ ◆	Leitz-Symbol muss sichtbar sein				2
4.10		Verpackung verschließen				4
4.11		Weitergabe an Station 5 und (falls nötig) neue Verpackungen ab Station 5 mitnehmen				5
Fotos:						
<div> <div>4.5</div> <div>  </div> </div> <div> <div>4.6</div> <div>  </div> </div> <div> <div>4.7</div> <div>  </div> </div>						

Figure 6.6: Instruction Set for Station 4.

6.1.2 Testing Procedure

The execution of the test run should mimic a real production scenario the best possible way. The goal is to provide a solid basis for evaluation of the solution and future improvements. For the given scenario, there are three different varieties of the product to be produced during the test runs. During a run which could represent a shift on a smaller scale, each variety will be assembled once. The complete execution will be performed three times to mimic three shifts of a working day. The run of the experiment will be recorded on video for future evaluation and error-proofing of the methods in place.

The instructions are to be followed by the test workers as they appear on the screen. The conversion of the instruction manual to the assembly interface is shown in figure 6.4

STATION 2			
		MONTAGE AUFBAU	
		Symbol: ▽ Kontrolle	● Trick
		Qualität	
		ini. In UNIVERSITÄT DUISBURG ESSEN	
		WORK ELEMENT SHEET	
Blatt 2	Stückliste: 1 x Wippe 1 x Raststück 1 x Bodenschale 4 x Mutter 1 x Formatschiene 2 x Stift		
Arbeitsnummer	Symbol	Beschreibung des Arbeitsvorgangs	Fertigungshilfsmittel
2.1		Einsetzen der Plastikblende in die Wippe, sodass die runden Ecken der Plastikblende die Wippe nicht berühren	
2.2		Einsetzen der beiden Sifte in die Wippe durch Loch 1 und Loch 3	
2.3	●	Sichern der beiden Sifte mit den Muttern (erst die Mutter an Loch 1) anschrauben und dann die Mutter an Loch 1)	
2.4	▽	Beide Muttern nachziehen	Schraubenschlüssel
2.5		Weitergabe der Wippe an Station 3	
2.6	▽	gleichfarbige Bodenschale zur Wippe nehmen	
2.7	▽	gleichfarbiges Raststück zur Bodenschale	
2.8		Raststück (ins rechte Loch) der Bodenschale legen, bis zum Anschlag	
2.9		Formatschiene durch das Raststück in die Bodenschale schieben	
2.10		Weitergabe der Bodenschale an Station 3	
Fotos:			
			

Workstation: 002

Product: Wippe

Partname: Wippe

Variant: BLUE

Status: ●

Elapsed Time: 00:00

Step 1

Step 2

Step 3

Parts

rocker x1

plastic blend x1

Instructions

Put the plastic blend into the metal rocker. I care to put the round edges flush with the metal shape!

Tools

Step 1

Step 2

Step 3

Parts

shaft x2

rocker x1

Instructions

Put the 2 metal shafts into the holes H1 and H3.

Tools

Step 1

Step 2

Step 3

Parts

nut x4

Instructions

Mount the 4 screws on the 2 shafts
1. First tighten the nut on Hole H2. Then tighten the nut on Hole H1!

Tools

screwdriver

Step 1

Step 2

Step 3

Figure 6.7: Side by side comparison of the paper-version instructions and the new interface.

6.2 Results

6.2.1 Product Views

The following is a documentation of the execution. During the test run a red, blue and black puncher have been assembled. For each product manufactured the tasks can be consulted which leads to one product view per assembly performed.

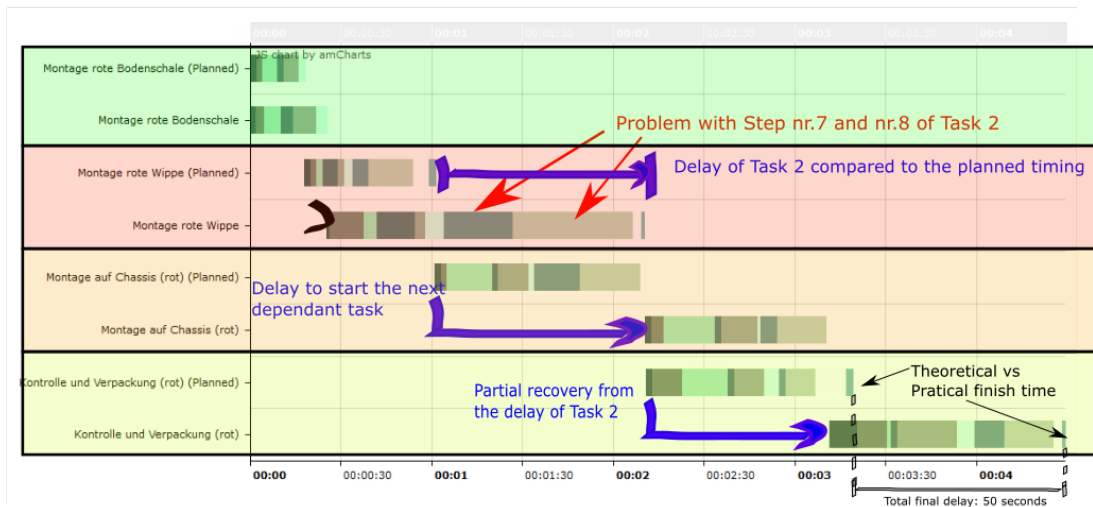


Figure 6.8: Assembly trace and practical evaluation of the red puncher test.

Figure 6.8 shows the base Gantt chart with some explanations towards how to interpret the result. The bottleneck occurs in the tasks #2 and #3, more precisely in the steps which deal with screwing the nuts on the screws and shafts of the puncher. This can be seen when hovering over the steps in the interactive page.

A closer observation would yield that an improvement has to be done for the concerned steps of task #2 and task #3. Another alternative would be to keep training with the worse performing workers on the assembly steps until the score makes the planned target. Definitely the assembly has to be rescheduled in this case to reduce the difference between planned and real timings for future tests. It has to be noted that the test was performed by unexperienced workers of which two out of four have never used or seen this system. Nevertheless the unexperienced workers performed very fluidly and it was indeed a challenge within the assembly line which lead to divergences.

The following two product assemblies in figures 6.9 and 6.10 show a similar scenario to the red puncher in figure 6.8. Again task #2 was underperforming.

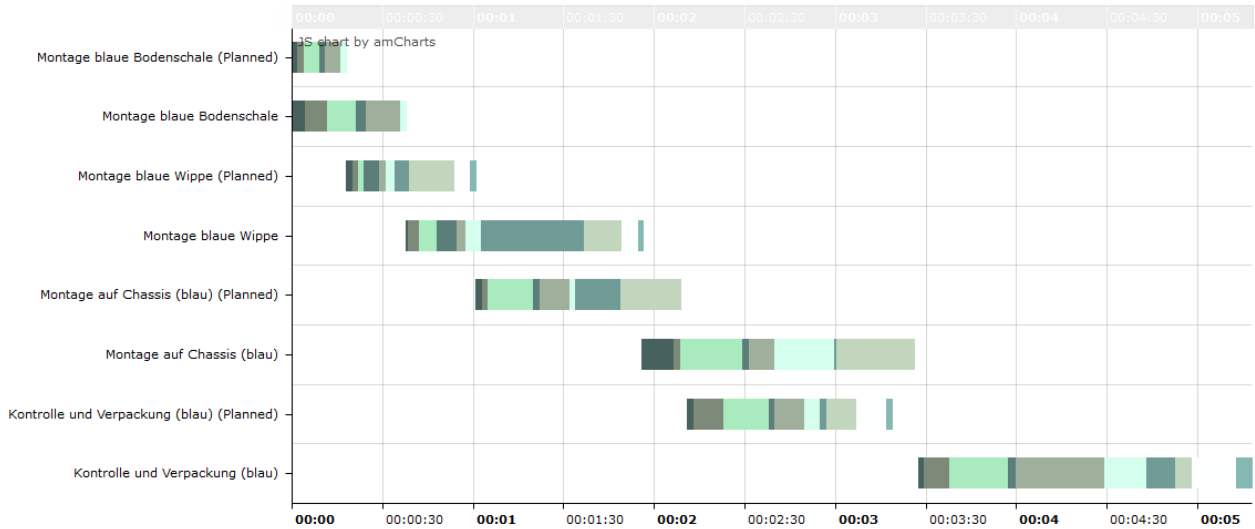


Figure 6.9: Product view for the assembly trace of the blue puncher.

In figure 6.10 it can also be seen that there seems to be an issue with the first task this time. From having seen the run of the test, it should be said that the assembly of the black puncher was the very first to be treated by the worker in question. This explains the long execution times for the new worker on the new system. In the following time this worker experienced a strong learning curve and managed to execute his task after only the third try and stay within the schedule for his workstation.

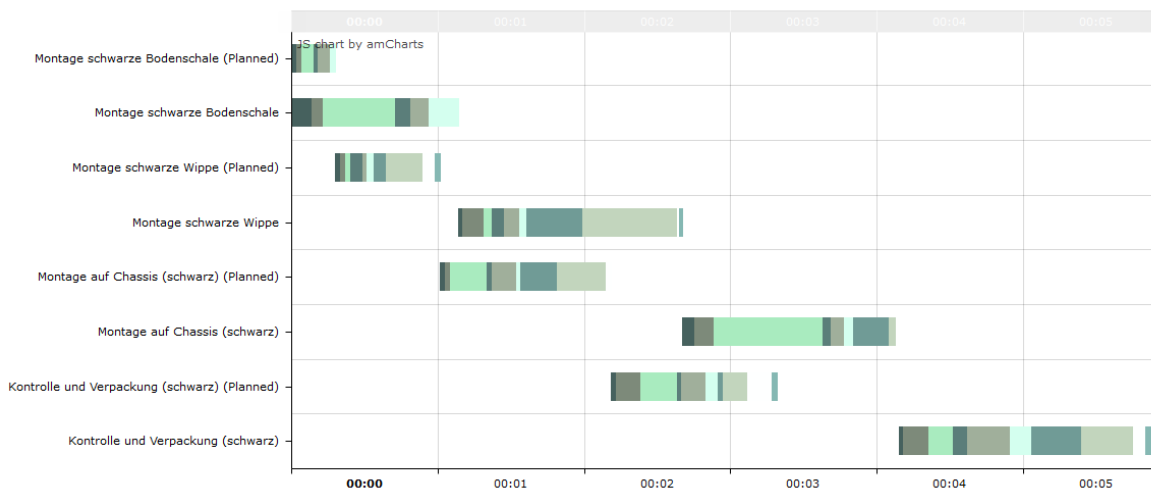


Figure 6.10: Product view for the assembly trace of the black puncher.

6.2.2 VSM View

The VSM view is implemented in CSHTML and ready to be populated with dynamic data. Due to the lack of time this part of the thesis could not be covered for the submission but awaits completion as the project is continued after submission.

6.3 Discussion

The integrity of the presented data was checked and rated plausible which means that the presented measurements are reliable and provide a useful basis to assess the run of the assembly line. The product view is still not the perfect tool but open for improvements. Something which was noticed immediately is the lack of indicators and recognition of the tasks and steps. This is to be clarified in the future but was intentionally not touched until the functioning of the view has not been fully tested. Annotating charts has to be done with care since an overloaded chart requires more time to be read. Information should be communicated through the symbolism and animations within the tool, not through explanatory text.

Chapter 7

Conclusion

Following the tests it is time to draw a line. We want to review which goals have been met during the three months of development, which goals have dropped out of scope and how the future of this work can be outlined.

7.1 Summary

The thesis starts from a state of the art assembly-line which features zero to no control of user-interaction and evolves into an assembly line which integrates the developed system seamlessly. The goal being to measure user-interactions and evaluate the productivity of the workers and quality of the assembly-line. The aim of the evaluation is to give insight on which options exist for improving the assembly-line. An improvement of the assembly line results in a reduction of manufacturing cycle times, quality and safety.

The thesis transitioned through project-like stages. Those have been "brainstorming", "technological discussion", "software conceptualization", "development strategy" and "testing". Those are followed in the exact same order in the paper but named slightly differently.

The brainstorming stage is mostly reflected within the chapter "solution path" 4 and based on a technological discussion, which comes up in the chapter "domain analysis" 3. After getting the concept map right, software development started. This first required a careful design of databases which hold future data and code the necessary back-ends for the communication with the front-end(s). In the meantime the front-end, meaning the GUIs were designed and scripted which are mainly the grid which is populated later on with the available data. The interface design choices are crucial to render the user experience intuitively and smooth. An unexperienced worker should not have to make up his/her mind about how to use the software but furthermore be guided through the assembly steps as if a coworker was explaining the steps to him/her.

The most important quality of the software apart from its ability to measure the execution times is the error recognition and immediate recovery. This means that if a worker is about to commit an error to the assembly, the software will notify the worker and wait until the right part is picked before any further instructions are given. This is especially useful in an environment where many parts look alike or are distinguished only by small changes like different size of holes in a part. This allows the worker to learn by himself which is wrong and which is right and allows for trial and error learning. While errors are not solely shown, they are also logged for statistical purposes. If an error persists then this can be seen in the final assembly reports and the situation has a chance to be improved in the future.

The complete system is at the basis tailored to facilitate improvements on the assembly line workstations, instructions, user interactions and scheduling. Small improvements can have an impact. For example reformulating the instruction of an assembly step can make this one easier to understand and result in less errors and better execution times for this step.

All in all we agree with the outcome of the software and the achievements reached during this thesis. The software serves to demonstrate the possibility of creating the software in question with the technologies discussed earlier. Not all aspects which have been planned for the duration of the thesis have been fulfilled but this was to be expected since the original schedule of the project was very eager. Two out of three targets have been reached. The third goal was to finish the VSM view which is already designed and ready to be populated with live data but the back-end is still under development and not ready for release. Features like the Andon-lights [5] and certain performance indicators are missing. As said those performance indicators are defined in this thesis as functions in section 3.4 but not yet scripted within the software back-end.

7.2 Future Research

The thesis started off from the work of Patryk Pomykalski[7] which was the basis. The opportunities for a system to be built on the RFID technology have been large and the path chosen by this thesis had no intention of funneling down on those opportunities. The library of Patryk has been integrated within the software and extended to fit our purposes.

The software as is can be extended in many directions. The main ideas for the future of this work are:

- Complete the VSM-view and implement it within the line.
- Optimize the Product-View and expand the current statistics.
- Handle more complex and completely unique Product combinations.
- Include a library within the code which gathers data on shop-floor conditions such as the noise-levels and temperature.

- Overthink the behavior of the Andon-indicators[5] and create a view which gives a better indication on the status of the workstations.
- Test Wi-Fi-behavior with multiple workstations. (In the tests performed for this thesis, the workstations were connected via ethernet cables).
- Integrate the system completely within the Lean-Lab assembly line. This work includes the design of a better holder for the screens of the RbPi devices as well as power cable management and a clean integration of the sensors connected to the RbPis.
- Perform a large scale test with all the operators of the Lean-Lab assembly line serving all the workstations in a test which is exactly the same as those performed on a monthly basis for educational purposes.
- Extend the implementation to support the automatic buffer handling which is introduced in this thesis. This includes automatic events when the workstation buffers reach critical levels and also then the scheduling of supply-chain tasks at the corresponding workstations.
- Include the error-timings within the Gantt chart in the form of thin red lines.

Bibliography

- [1] Product variety and manufacturing performance: Evidence from the international automotive assembly plant study. *Manage. Sci.*, 42(3):350–369, mar 1996.
- [2] Joseph C. Chen, Ye Li, and Brett D. Shady. From value stream mapping toward a lean/sigma continuous improvement process: An industrial case study. 48:1069–1086, 01 2010.
- [3] George Q. Huang, Y.F. Zhang, and P.Y. Jiang. Rfid-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts. *Robotics and Computer-Integrated Manufacturing*, 23(4):469 – 477, 2007.
- [4] Masaaki Imai. *Kaizen*. BRISA, 1994.
- [5] Jeffrey K. Liker. *The Toyota way: 14 management principles from the worlds greatest manufacturer*. McGraw-Hill, 2004.
- [6] Christof Oberhausen and Peter Plapper. Value stream management in the lean manufacturing laboratory. *Procedia CIRP*, 32:144149, 2015.
- [7] Patryk Pomykalski. Rfid sensor based collaboration platform in iot environment.
- [8] Mike Rother and John Shook. *Learning to See - Value Stream Mapping to Add Value and Eliminate Muda*. Lean Enterprise Institute, NC, 2003.
- [9] Steven Saar and Valerie Thomas. Toward trash that thinks: Product tags for environmental management. *Journal of Industrial Ecology*, 6(2):133–146, 2002.
- [10] Peter R. Scholtes. *The leaders handbook: making things happen, getting things done*. McGraw-Hill, 1998.