# System Requirements Specification

# Robotic Waffles

Version 0.1  
Prepared by Tony Björkman

Table of Contents

[System Requirements Specification 1](#_Toc21687414)

[Robotic Waffles 1](#_Toc21687415)

[Revision History 3](#_Toc21687416)

[1 Introduction 3](#_Toc21687417)

[1.1 Document Purpose 3](#_Toc21687418)

[1.2 Product Scope 3](#_Toc21687419)

[1.3 Definitions, Acronyms and Abbreviations 3](#_Toc21687420)

[1.4 References 3](#_Toc21687421)

[1.5 Overview 4](#_Toc21687422)

[2 Overall Description 4](#_Toc21687423)

[2.1 Product Perspective 4](#_Toc21687424)

[2.2 Product Functions 4](#_Toc21687425)

[3 System Features Requirements 4](#_Toc21687426)

[3.1 Functionality 5](#_Toc21687427)

[3.1.1 Movement 5](#_Toc21687428)

[3.1.2 Dispensing ingredients 5](#_Toc21687429)

[3.1.3 [Extra] Detecting refill level 5](#_Toc21687430)

[3.1.4 Extract Waffles 5](#_Toc21687431)

[3.1.5 Frying the waffles 5](#_Toc21687432)

[3.1.6 Light barrier-safety 5](#_Toc21687433)

[3.1.7 [Extra] Multi-iron use 5](#_Toc21687434)

[3.2 User Interfaces 5](#_Toc21687435)

[3.2.1 Order Button 5](#_Toc21687436)

[3.3 Hardware requirements 6](#_Toc21687437)

[3.3.1 Waffle iron 6](#_Toc21687438)

[3.3.2 Robot tools 6](#_Toc21687439)

[3.3.3 [Extra] Light barrier 6](#_Toc21687440)

[3.3.4 Batter bowl 6](#_Toc21687441)

[3.3.5 Output tray 6](#_Toc21687442)

[3.4 Software requirements 6](#_Toc21687443)

[3.4.1 Virtual robot run 6](#_Toc21687444)

[3.4.2 Physical objects 6](#_Toc21687445)

[3.4.3 Waffle Iron 7](#_Toc21687446)

[3.4.4 Communication 7](#_Toc21687447)

[3.5 Quality attributes / Non-functional requirements 7](#_Toc21687448)

[3.5.1 Robot first 7](#_Toc21687449)

[3.5.2 Mechanical speed 7](#_Toc21687450)

# Revision History

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Document Purpose

This document is the first part of the engineering process and expands on the idea that a robotic arm shall make waffles for humans in a manner suitable for a wedding party. Exactly how this should be done is not known at the start of the project and thus the requirements and how they relate to features need to be thought out before the design can start.

After the features needed are conceived, they will be gathered in this document. This opposed the design-as-you-go method of starting designing as soon as a single feature has been identified (such as the robot needs to have a fixed-mounted scoop as tool) and when that is finished start looking for the next feature to complete without regards for the complete picture, perhaps there might be a feature requiring changing tools!. Problem is that requirements might show up, due to technical difficulties or other, that makes a previous fully implemented design obsolete.

Purpose of this document is to prevent unnecessary work that needs to change and to have a stable foundation with features that elegantly can cover the requirements and add fun and cool requirements when they are very easy to add.

## Product Scope

The system is from the start decided to revolve around the Mitsubishi MELFA RV-E3J and is thus a one off since this robot arm is end-of-life. The arm uses its standard controller CR-E116 for power and positioning, also end-of-life. The programming language used for robotic control is MELFA Basic III (don’t have a good source on that though).

The system includes everything required to receive waffle orders and deliver the finished waffles in an almost safe manner.

## Definitions, Acronyms and Abbreviations

## References

List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.

## Overview

Overall description TBD

# Overall Description

## Product Perspective

The system is part of a discovery for myself on how to apply robotics to solve real-world problems. It will be used and stress-tested under production-like conditions at once specific event, a wedding party serving guests on demand.

I hope to be able to re-use parts of this project for other similar projects in the future. Therefore, it is desired to have a design where the modules that might be used in other projects are loosely coupled.

The focus relies heavily on using the robot arm to solve tasks, even if other solutions would be more efficient and easier. This will become obviously clear when solving the dispensing problem, where pumps and tubes would solve the task a lot more efficient than robot arm tools.

All files relating to the project will be made available in a public repo, for others to use or get inspiration from and as a portfolio item showcasing a hopefully complete project.

## Product Functions

So, what are the core things we need to think about with regards to the system to find out the requirements we need?

This is what the system does:

* Physically grip and move batter, waffles and possibly other things
* Control power of multiple waffle-irons
* Safety measures to prevent mechanical accidents
* Notify the operator when refill/maintenance is required
* Schedule the waffle-orders and dynamically use the waffle-irons available
* Solution for serving the waffle to the user in a suitable way based on the circumstances.
* Receive orders for waffle

# System Features Requirements

Here all the requirements for the complete system is gathered, do note that requirements regarding the robot-arm, controller box are left out since they are already selected and will not change.

The commands used to control the robot are listed since it at moment of writing this specification, it is not clear what the capabilities of the robot control are.

Requirements tagged “Extra” is not needed for the most basic implementation.

## Functionality

### Movement

* Move ingredients without spilling
* [Extra] Funny movements
* [Extra] Stir the batter

### Dispensing ingredients

* Dispensing a fixed amount of ingredients
* [Extra] Dispensing the ingredients with different coverage, ie both pile and spread out
* [Extra] Dispensing multiple types of ingredients without mixing them (ie. No batter in the jam)

### [Extra] Detecting refill level

* Determine if ingredient stocks are good, need refill or critically low.

### Extract Waffles

* Open/close lid of waffle-iron in a controlled manner
* Retrieve hot waffles from the even hotter iron so they can be placed on output tray.
* If orders are pending, do not close the iron-lid since it will anyway directly be opened.

### Frying the waffles

* Turn on/off power on the iron (can use the robot to do it without relay!)
* Detect when a waffle is finished frying

### Light barrier-safety

* [Extra] Cover the customer facing area with a light-barrier to engage stop if crossed

### [Extra] Multi-iron use

* Allow for a dynamic number of waffle irons, such that multiple waffle cycles can execute in parallel but towards different irons.

## User Interfaces

### Order Button

* Button which initiates/queues a waffle to be made

## Hardware requirements

### Waffle iron

* High power to enable quick cooking
* Easily opened and closed with the robot arm
* Easy to retrieve waffles with robotic gripper ie. Fit with plastic turner or be non-teflon.
* [Extra] Easily add thermometer
* Give “Finished” signal that can be interfaced to the

### Robot tools

* In one transport carry the full amount of batter needed for a single waffle
* Open/close waffle iron-lid without letting it smash
* Remove the waffle without dropping it mid-way
* Waffle-mover shall be heat-resistant if it comes in contact with the iron
* Tools interacting with iron should be long enough to keep the robot from needing to position itself over the iron to avoid heat and dirt.

### [Extra] Light barrier

* Cover the outer perimeter of the system, but open for waffle-retrieval.

### Batter bowl

* Contain enough batter for 30 waffles (single iron 60 minutes/dual 30min)
* When putting a ladle straight down, should empty the bowl as much as possible. Ie having a hole/compartment for the ladle. This is to minimize refills and keeping the batter fresh.

### Output tray

* Room for a stack of 10 waffles

## Software requirements

### Virtual robot run

* The movement pattern shall be possible to pre-run virtually or some other way to make sure that the robot will not crash when moving due to faulty pathways and positions.

### Physical objects

* Each physical object in the robot environment, such as a tool-stand on which certain operations can be performed, shall be self-sufficient in its co-ordinate system so the robot only needs to know origin of that object to perform the objects operations.
* The physical objects have one or more operations
* The physical objects operations can include IO’s, such as “close grip” and read a proximity sensor that affects behaviour.
* The physical objects operation should also be able to read the robot state to determine actions, ex if robot is in unexpected position.

### Waffle Iron

* The dual waffle iron can make either one or two waffles. If only one waffle has been started and the lid closed, it needs to be finished before it can be opened again.

### Communication

* Validating the response for the messages sent to the robotcontroller. Ie, was the response returned correct? If not, throw an exception to give the option of cancelling the send-queue.
* Send single message and wait for response before sending the next

### Waffle making multitasking

* Should use as few tool-changes as possible. I.e. once the robot is equipped with a certain tool, it shall perform all waiting tasks on any currently running job that requires that tool before switching tool.

### Waffle making responsiveness dynamics

* The software implementation should allow for a waffle iron to get batter from a bowl selected at run-time. The task of serving waffle should also be dynamic in that sense, which means that the tray on which it serves (if multiple exists), can be chosen at run-time. The reason for this is that a bowl might be empty, a tray might be full.

## Quality attributes / Non-functional requirements

### Robot first

* If a task can be solved with either extra equipment or the robot arm, the solution with the robot arm should be chosen. Such as letting the robot flip a switch instead of using a relay. A human should always be able to take over the working station.

### Mechanical speed

* Move at a fast speed, robots moving fast are cool and adds challenge. Most robots on Youtube are too slow!