System Description

*Sub-category: robot*

Contents

[Introduction 3](#_Toc98695161)

[Project overview 4](#_Toc98695162)

[Robot’s role 4](#_Toc98695163)

[Should/Could 5](#_Toc98695164)

[Hardware 6](#_Toc98695165)

[Advanced: Maths 7](#_Toc98695166)

[Glossary 8](#_Toc98695167)

[Sources 9](#_Toc98695168)

# Introduction

In this document all information surrounding the robot is gathered. This document can be used to dive a little bit deeper into the functionality of the robot, and how certain mechanics are used to employ the robot to its fullest.

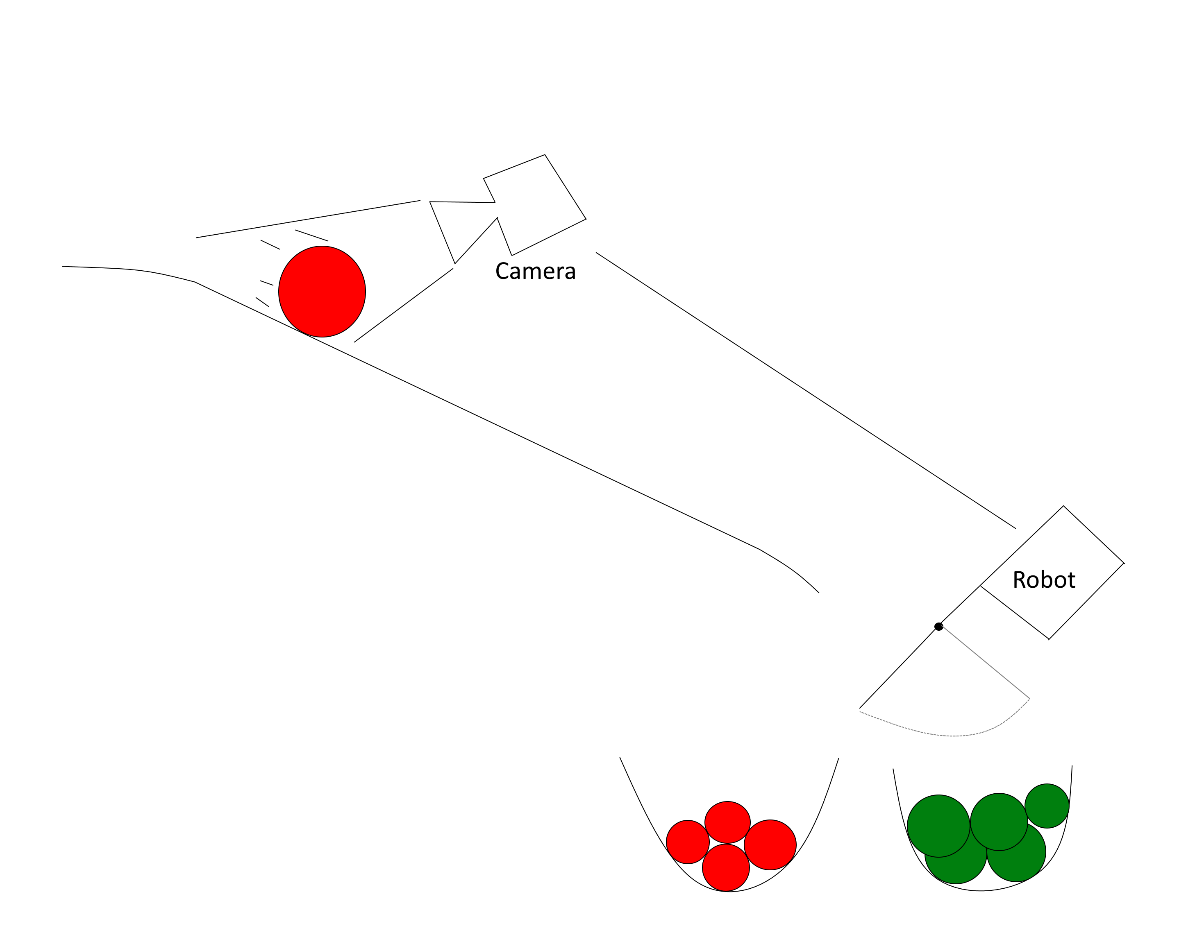
# Project overview

This project aims to provide a quick and easy to implement sorting machine for fruits and vegetables coming down on a conveyor belt. This is achieved by a collaborative effort between multiple microcontrollers, a camera and a robot. Our camera uses machine learning to recognize fruits and vegetables, and even distinguishes them based on quality. Does the quality **not** meet the set expectations, our robot will make sure this item is removed from the selection.

Due to the smart nature of our system, it can be, taking size constraints into account, installed virtually everywhere.

## Robot’s role

The robot makes sure the fruits/vegetable ends up in the correct spot. Either the “Good” or “Bad/rotten” bin. It does this by moving its arm, and thus, via its attachment, affecting the trajectory of the fruits/vegetables. The robot receives this “choice of trajectory” via the camera-unit.



*Early brainstorm sketch of the system, and specifically how the robot chooses trajectory.*

# Should/Could

The robot should be able to do some things in our system. However, additional/extra features or “QOL, Quality Of Life” implementations can make the system more accessible or even easier to work with.

Should:

* Be able to quickly change the directory of the conveyor belt
* Be able to seamlessly interpret signals coming from the camera
* Be able to store multiple pre-set conveyor positions

Could:

* Be able to sent data over MQTT
  + Extensive data gathering
* Be able to be controlled manually both physical and digital

# Hardware

* Morobot

This is a robot acquired from the technical college in Austria. Elaborate pictures and movement dynamics will follow.

* ESP32/Arduino Uno

This robot can be ran with either an ESP32 **or** an Arduino Uno. The choice depends solely on the fact if the user would like to use the internet capability the ESP32 offers. Since our implementation of the robot is bound to use internet, our choice will in most cases be the ESP32.

* A handful of jumper wires
* End affector

The end affector of our robot will be something that is able to adjust the trajectory of the conveyor belt. Since this is subject to both testing and change, this is not sure yet.

# Advanced: Maths

Since jointed robots use a lot of math in order to determine position, research about these kind of formulas is trivial for a quick and responsive robot. Hence we chose to start gathering data on this from day one. So far we are in the primitive state of understanding inverse kinematics, a method used to get the angles of certain links.

Tutorial: <https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/>

# Glossary

# Sources

<https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/>

To do:

* System description (what does the system do, and why)
  + List the different parts of the system in this paragraph
  + List of ideas associated with developing this project
  + Hardware list
* Glossary (meanings of terms used which aren’t “conventional”)
* Should/could list
* Maths
* Hardware
  + Morobot
    - Movement/aiming
  + ESP32/Arduino Uno
    - Controlling Morobot
    - Communication via MQTT
  + Zed 2 Stereo Camera
    - Input for Machine learning
  + Jetson Nano
    - Process Zed Camera input
    - Berry/bug detection

Short explanation of the system flow:

* a bug is on the Zed 2 Camera feed
  + bug detection algorithm running on Jetson Nano detects the bug
    - message is sent to ESP32/Arduino which controls Morobot
      * for example, Morobot aims and shoots bug repellent at the bug