IMPERIAL

Vision-Assisted Mechatronic Component Sorter

MEng Final Year Report

Author
Shaheen Amin
CID: 01866464

Supervised by
Dr. Ed Stott
Prof. -

A thesis submitted in fulfilment of requirements for the degree of
Master of Electronics and Information Engineering

Department of Electrical and Electronic Engineering
Imperial College London
2024

Plagiarism Declaration

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.
I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.
I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me but is represented as my work.
I have used GPTv4, as an aid in the preparation of my report. GPTv4 was used for LaTeX formatting and spellchecking, however, all technical content is original.

ABSTRACT

To address the issue of electronic waste and cluttered workspaces in the Level 1 Labs, this project aims to develop an automated system for identifying and sorting electronic components. Utilising computer vision techniques, the system will classify electrical components such as resistors, capacitors, ICs, and MOSFETs, and sort them into designated bins. The project comprises three core components; the mechanical design of the system, the integration of electronics and software, and the development of a computer vision system. Supervised by Dr. Edward Stott, this project aims to solve the problems faced by the Level 1 Labs.

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

add acknowledgements

Contents

Todo list

1 Project Specification

The motivation for this project is three-fold; to solve the problem of electronic waste and cluttered workspaces in the Level 1 Electrical Engineering Labs; to alleviate the time-consuming process of sorting electronic components from the Labs' technicians; and to bring the Lab closer to meeting the requirements for the LEAF (Laboratory Efficiency Assessment Framework) certification [30] that the Lab is currently working towards. The LEAF certification is a framework that aims to improve the efficiency of laboratories by reducing waste, energy consumption, and costs. This project aligns with the LEAF certification's goals by reducing the amount of electronic waste produced by the Lab.

The project aims to achieve these goals by employing state-of-the-art computer vision techniques to classify various electronic components, and then sort them into designated bins. The project's deliverables are as follows:

- **Vision System:** A computer vision system capable of real-time classification of electrical components.
- **Mechatronical System:** A mechatronical system that can sort the classified components into designated bins set by the user.
- **User Interface:** An interface from which to observe and control the system's state.

Samples of the different electrical components commonly used in the Level 1 labs were collected, and they are as follows:

- Resistors
- Capacitors
- Ceramic Capacitors
- Inductors
- Diodes
- MOSFETs
- Transistors
- LEDs
- Wires
- Integrated Circuits

Conform more to Clarke's guidelines

fix all pictures

2 Background

This chapter will delve into the background of the project, exploring the existing literature on computer vision systems for component identification, real-time computer vision architectures, and the mechanical design of existing sorting machines. This research will inform the design of the project and its various systems. Research into these topics will inform the decisions made in the design of the project and its various systems.

2.1 Existing Computer Vision Systems

A range of computer vision systems have been explored in the literature, ranging from PCA (Principle Component Analysis) Dhenge et al. [9] to more modern computer vision techniques like CNNs as used by Chand and Lal [5], Xu et al. [48]. However, given the rate of advancement of computer vision techniques, the techniques used in the literature reflect the state of the art at the time of writing, and so are not necessarily the most viable current techniques for the project, but they inspire novel approaches to the problem of component identification.

PCA with ANN (Artificial Neural Networks) as used by Dhenge et al. [9] is a relatively simple but outdated statistical technique that was successful in identifying nuts and bolts, which are very distinct

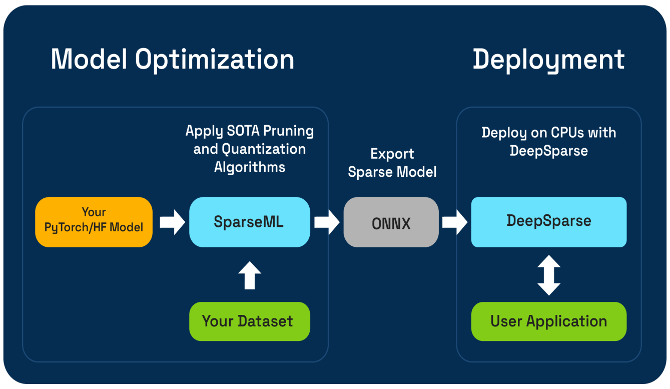


Figure 1: SparseML Pipeline [25]

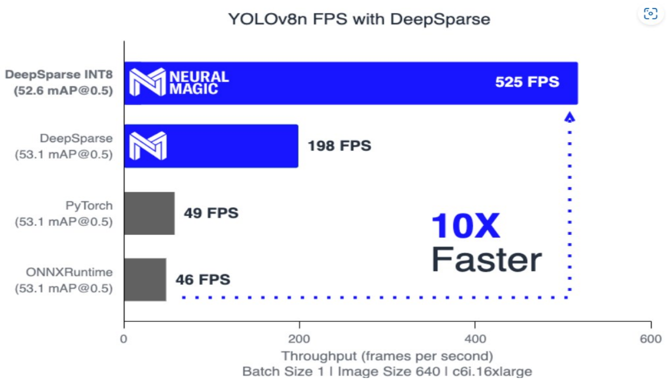


Figure 2: DeepSparse Performance [24]

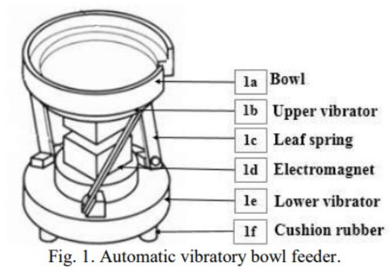


Figure 3: VBF[28]

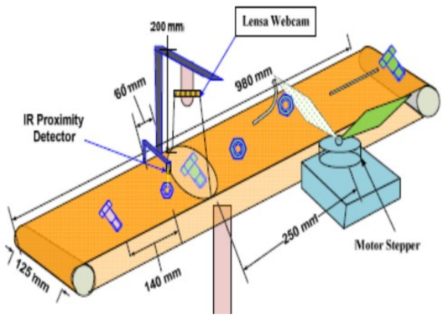


Figure 4: Conveyor Belt[9]

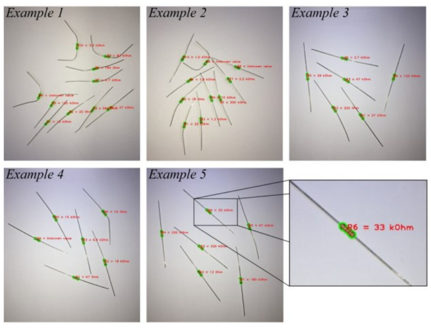


Figure 5: Resistor Test Set[26]

components. In this paper, PCA is used as a feature extractor and then fed directly into, though not explicitly mentioned, a FCL (Fully Connected Layer), as the classifier. Although a valid approach, CNNs (Convolutional Neural Networks) are known to be much more effective at feature extraction, and do not "have the problem of low recall and accuracy", as mentioned by Xu et al. [48], that PCA may suffer from, so this approach is not viable for the project.

The authors Xu et al. [48] use the SqueezeNet CNN architecture to identify 22 different subcategories of electronic components, specifically resistors, capacitors, and inductors, achieving a TPR (True Positive Rate) of 99.999% with only a 2.67ms average inference time on a GTX 1050 2GB GPU (released in 2016) which is an impressive result. This work shows that a CNN is a viable approach to the problem of component identification, helping to inform the design of the project's computer vision system.

The paper by Muminovic and Sokic [26] discusses the use of an SVM (Support Vector Machine) to characterise resistors, by identifying the resistor's centroid (centre of mass), determining the resistor's orientation, and then analysing the bands to determine the resistor's value. This is directly applicable to the project, as it is a viable approach to identifying resistors, which are very distinct from other components, given the presence of colour bands. The paper's novel approach to identifying the resistor values allows it to achieve high accuracy (86%, however, the test images used are resistors that are positioned far away from the camera, as shown in Figure 5, and so the bands are very small, which will not be the case in the project's vision system, so the accuracy will likely be higher) which makes it a very promising resource for the project.

2.2 Computer Vision Architectures

For the problem of component identification, the computer vision system must be able to identify components in real-time, as the components may eventually be moving on a conveyor belt, captured using a camera facing down at the components. This means that the computer vision system must be able to identify components in a very short amount of time, and so the system must be computationally efficient. The envisioned system is designed to be self-contained, and as described in Section 4 (Implementation), the system will run on a Raspberry Pi 4, which has a 1.8GHz quad-core 64-bit ARM Cortex-A72 CPU and up to 8GB of RAM [14], and must also be accurate and able to control the other systems in the project in real-time.

This requires the Computer Vision system to be both computationally efficient and accurate, which is a difficult balance to strike.

For the explicit purpose of identifying electronic components, the paper by Chand and Lal [5] compares a range of different object detection architectures, including YOLOv3, YOLOv4 and Faster SqueezeNet, with YOLOv4 achieving a mAP (mean Average Precision) of 98.6%.

YOLO (You Only Look Once) is a real-time object detection architecture, that is used very commonly in computer vision applications and is very effective at identifying objects in real-time Terven and Cordova-Esparza [44]. It makes use of a single CNN that takes in an image and outputs a list of bounding boxes and class probabilities for each bounding box. This is in contrast to other object detection architectures, such as R-CNN, which uses a CNN to propose regions of interest and then uses a second CNN to classify the regions of interest.

Other papers like Guo et al. [19] also comment on YOLOv4's effectiveness at identifying electronic components, achieving 93.94% mAP on a dataset of 20 different components, and the paper also comments on YOLOv4's ability to run in real-time, achieving 67 FPS, albeit on a powerful NVIDIA TITAN Xp GPU. For the Raspberry Pi 3B, the paper by Sismananda et al. [43] has shown to run YOLOv3 at a very low 1 FPS, with an IoU (Intersection over Union) accuracy of 86.7%, which is relatively low.

However, efforts made by Neural Magic [24] in the optimisation of YOLOv5 and YOLOv8 show performance improvements of up to 10x on CPUs, through their open-source optimisation toolkit SparseML [25] and CPU inferencing runtime, DeepSparse [23]; a very promising result.

2.3 Training Methods

When training a neural network, it is important to have training data. For a computer vision system, this means having a dataset of images that are representative of the conditions that the system will be used in and are well-labeled with bounding boxes and class labels. It is important to have a large dataset to ensure that the model is robust and generalises well, which is time-consuming to create.

To solve this problem, a paper by Yang et al. [49] proposes many different training techniques, the most promising of which is semi-supervised learning. In this approach, the model is trained on a small labeled dataset, and then used to label a large unlabelled dataset, which is reviewed and corrected by a human.

This process is repeated until the model is sufficiently accurate, and then the model is trained on the large labeled dataset. This approach is very promising, as it allows for the training of a robust model with a large dataset, without the time-consuming process of manually labeling the entire dataset.

2.4 Mechanical Design

Transport Mechanisms
The paper by Dhenge et al. [9] depicts a conveyor belt system that transports nuts and bolts to a computer vision system for identification, which aligns with the goals of the project. The hardware prototype is shown in Figure 4, and a down-facing webcam is used to capture images of the components as they pass by. The webcam uses Principle Component Analysis (PCA) to identify the components, which will be discussed in the next section.

The components then separate into two chutes, one for nuts and one for bolts, using a stepper motor, which may be useful for the future design of the sorting system. Additionally, the approach taken by the paper Quilloy et al. [37] to sort Philippine table eggs also features a similar conveyor belt system, a down-facing camera and an arm to sort the eggs into different categories.

The approaches taken here are similar to industrial sorting systems seen in videos while researching for this project; this approach seems to answer three major design questions for the project; how to transport the components to the computer vision system; how to transport the components from the computer vision System to the sorting system; and the placement of the camera.

Other approaches include vibratory feeders[21], which use precise vibrations to orientate and transport components, and pneumatic systems from Abe et al. [1] (also suggested by project Supervisor Dr. Stott when dicussing transporting mechanisms in project meetings), use air pressure in tubes to transport components. These approaches require more complex hardware, and lack the easily achievable precision of the conveyor belt system, for example in the case of the pneumatic system, a complex network of tubes and valves and an air compressor is required, which is not practical for the project. Robotic arms are commonly used in the industry for sorting, however, this is not viable as this would either require an expensive robotic arm, or a complex system of motors and actuators to move the components.

From the approaches above, it seems that a conveyor belt with a down-facing camera is the most viable approach to transporting the components. Initially, the aim was for the system to be semi-autonomous, with a design allowing the camera to face upwards. This configuration would enable the user to place the component on an acrylic plate directly above the camera for immediate identification, enabling easy user interaction as the view of the component is not obstructed. Having the camera face-down would block the user's view of the component, which would be an inconvenience, and as such, the current design of the system features an up-facing camera. However, this will be changed to reflect the research outlined above, in the next stage of the project.

Bowl Feeders
While researching for this project, and especially in videos[46], it seems most industrial sorting machines use a Vibratory Bowl Feeder (VBF) to help feed components into the sorting system; as shown in Figure 3, the VBF consists of a bowl that vibrates coupled with a spring and electromagnet. The paper by Nam et al. [28] explores the optimal design of a VBF for USB keycaps, by attempting to identify the ideal parameters for the structure of the bowl, sorting track, mounting adapter, and suspension system. The paper also uses modal analysis to determine the natural frequencies of the system and uses this to avoid resonant conditions that might cause inefficient or erratic operation.

This paper is useful as it provides a comprehensive overview of the design of a VBF, and provides a good starting point for the design of the VBF. In the future, the project may make use of one for fully autonomous sorting. The paper also provides a good overview of the design considerations for the VBF, and so can be used as a reference during the design process.

Additionally, Reinhart and Loy [39] delve into a mathematical model of a VBF, optimising more on the overall performance of the VBF rather than efficiency, and Silversides et al. [42] provides a good overview of the forces involved in the operation of a VBF, strengthening the basis for its design and viability.

The paper by Zhengyang Zhang [50] outlines a sorting system for vials and does not make use of a VBF, instead opting for a turntable design that mechanically orientates the vials. It primarily operates by using a design that is specific to the geometry of the vials, and so does not apply to this project, however, it does provide a good insight into the design of a sorting system.

An alternative to the VBF could be a robotic arm; fine control over movement would be necessary as the components are small and may be tangled, however, as mentioned before would require its own set of sensors and camera systems. As the components can be tangled, there are not many viable alternatives to the VBF or a robotic arm, so between these two solutions, it seems that the VBF is the most viable approach if the system is to be fully autonomous, as the vibrations of the VBF would help to untangle the components.

2.5 Commerical Systems

complete this section with a few youtube videos

2.6 Key Takeaways

After reviewing the literature, the following key takeaways were made:

The most viable approach to the problem of component identification is to use a CNN, specifically the YOLOv8 architecture, as they are very effective at classification tasks, which translates to being able to identify electronic components. With the ability to optimise the model using SparseML and DeepSparse, the model may be able to run on the Raspberry Pi 4 in real-time, which is a key requirement of the project. Additionally, the YOLO architecture may be covered in the modules Deep Learning COMP70010 and Machine Learning COMP70014, enriching the understanding of this architecture.

The most viable approach to the problem of component transportation seems to be a conveyor belt system with a down-facing camera, and so the design of the computer vision system will be based on the research outlined above in the next stage of the project.

A VBF is the most viable approach to the feeding mechanism of the sorting system, and so the design of the VBF (if employed in the project) will be based on the research outlined above.

Additionally, the YOLOv8 architecture will be employed, making use of the SparseML and DeepSparse optimisation tools with semi-supervised learning to train the model.

3 Design

This chapter will discuss the design and system architecture of the project. The project is divided into three main components: the mechanical design, the integration of electronics and software, and the computer vision system. The mechanical design will be discussed in Subsection 3.3, the electronics and software integration in Subsection 3.5, and the computer vision system in Subsection 3.4.

3.1 System Architecture

The project's design is underpinned by a modular design philosophy; each subsystem is designed to be as independent as possible, allowing for easier debugging and maintenance, and expose only high level interfaces to other subsystems. This not only simplifies the development process but also allows for easier integration of new features in the future.

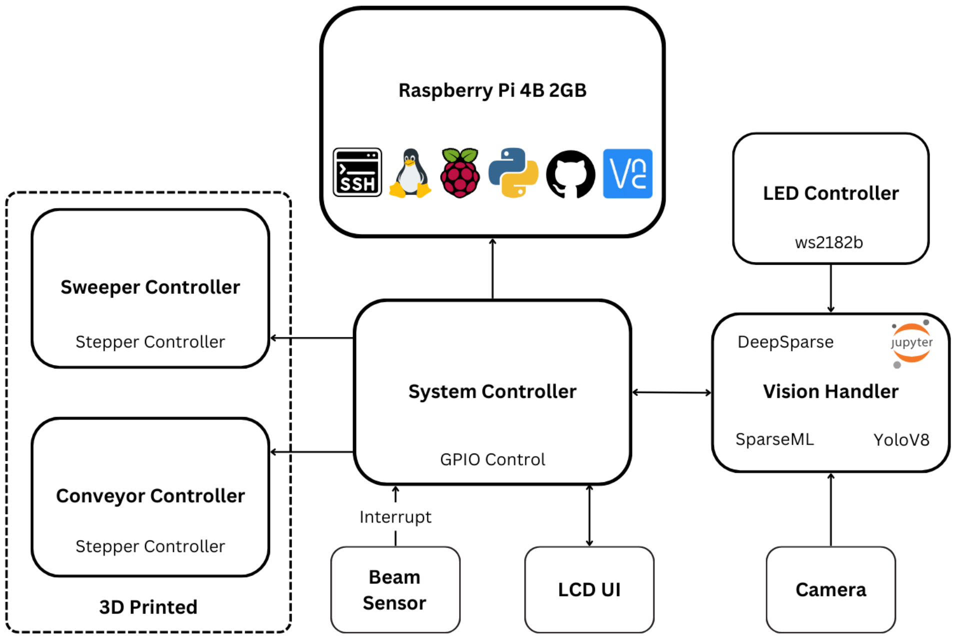


Figure 6: System Architecture Diagram

A System Controller oversees the entire system, interacting with all other components and controlling all communication. It is responsible for interacting with the Vision Handler, and passing commands to the Conveyor Controller and Sweeper Controller. The Vision Handler is responsible for processing images from the camera and performing inference for classification. The Conveyor Controller is responsible for controlling the conveyor belt, while the Sweeper Controller is responsible for controlling the sweeper. Both of these controllers also control a TMC2209 stepper motor driver, which in turn controls the stepper motors that drive the conveyor belt and the sweeper.

There is two-way communication between the LCD UI and System Controller as the user may input commands to the system, and the System Controller may send status updates to the LCD UI.

3.2 Hardware

When selecting hardware for the project, it is vital to consider the requirements of the system and to evaluate the trade-offs between alternatives. This section will detail the choice of hardware and provide justification for the selection.

3.2.1 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B [14] was chosen as the main component of the system and is the central hub that all other components are connected to. This model was chosen as it is regarded as a reliable SoC (System on Chip) and is widely used in the industry. It has a large amount of software and driver support, ensuring confidence in finding solutions to problems encountered over the course of the project. Additionally, it has GPIO pins that can be used to control the other components in the system, such as stepper motors and LEDs. It also has a dedicated CSI port which allows for a camera to be connected directly to the SoC, which is necessary for the computer vision system.

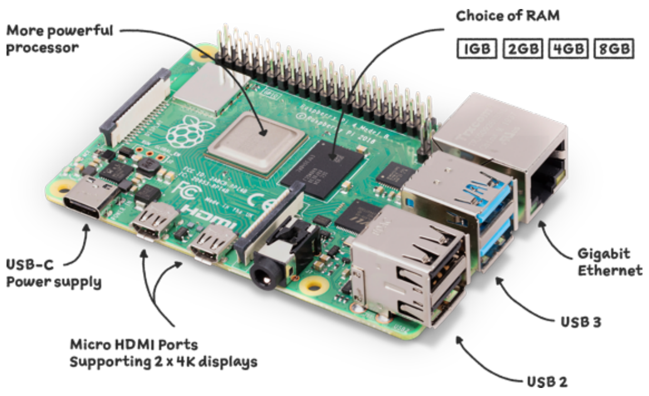


Figure 7: Raspberry Pi 4 Model B 2GB [14]

It also features Wi-Fi support, allowing for SSH and remote development software, as well as an HDMI port for the display. Originally, the 4 GB model was to be used however an order issue resulted in receiving the 2 GB model. This was found to be not an issue as the 2 GB model still performed well, but there was now a heavier need for efficient code to ensure the system runs smoothly.

An alternative to the Pi family of SoCs could be an Arduino-compatible microcontroller, but while they are capable of controlling the components of the system, and potentially drive an LCD, they lack the resources to run the Computer Vision system and stream video from the camera. Due to this, they cannot be used as a replacement for the Pi. It was considered to use an Arduino-compatible microcontroller to delegate control of certain components to overload the Pi, but this was found to be unnecessary as the Pi was able to handle the load of the system.

A viable alternative to the Pi could be a Nvidia Jetson Nano [29], as it is designed for Computer Vision applications and has a dedicated GPU. The dedicated GPU would reduce the likelihood of the Computer Vision system being a bottleneck, and it has a CSI port for the camera. In terms of CPU performance, the Nano is weaker than the Pi, having a quad-core ARM Cortex-A57, whereas the Pi has a quad-core ARM Cortex-A72[14]. However, the worse CPU performance is offset by the dedicated GPU, and the Nano has 4 GB of RAM, making the Nano a very attractive alternative to the Pi. The Nano's drawback is that it is significantly more expensive than the Pi (up to 3x from retailers), and given the ability to optimise the computer vision system as discussed in Section 2 (Background), the Pi is still a viable choice for the computer vision system.

To address the lack of a dedicated GPU for inference on the Pi, an article from Medium[2] benchmarked the Pi's performance on various computer vision tasks with various accelerators like the Intel Neural Compute Stick 2 and the Google Coral USB Accelerator against other SoCs like the Jetson Nano and the Coral Dev Board. It was found that the Coral Dev Board was the best performing, however the Pi using optimisation libraries like TensorFlow Lite showed promising results, which helps to strengthen the argument for using the Pi for the computer vision system, especially with the SparseML and DeepSparse libraries that were discussed in Section 2 (Background).

The benchmarks can be found in Appendix Figure 21.

3.2.2 DFRobot 7″ Touchscreen Display

The DFRobot 7″ Touchscreen Display was chosen for the LCD panel as it is a relatively cost-effective display that is compatible with the Raspberry Pi. It has touchscreen support and features Raspberry Pi 4 mounting holes on its back, as well as HDMI adapter boards to connect to the Pi. This means that a physical mount for the Pi does not need to be designed, and only a mount for the display is required. The display is also powered by the Pi, so no additional power supply is required.

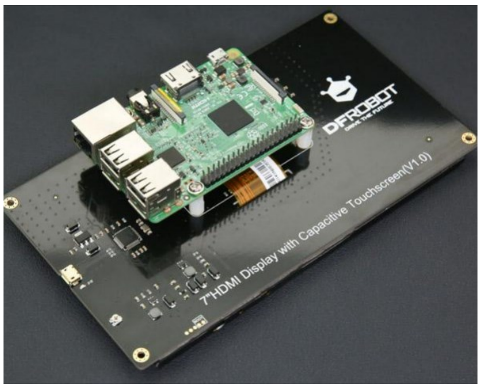


Figure 8: DFRobot 7″ Touchscreen Display [8]

Other alternatives to the DFRobot 7″ display were various 10″ displays however many required proprietary drivers and cables, and were more expensive. The inclusion of mounting holes on the back of the DFRobot display was also a major factor in its selection, which seems to be uncommon in other displays.

3.2.3 Okdo Adjustable Focus OV5647 Camera

For the Computer Vision system, a camera is required to capture images of the components. The Okdo Adjustable Focus OV5647 Camera was chosen as it is CSI compatible, meaning it can be connected directly to the Raspberry Pi, and has a manual focus ring, allowing it to be used as a macro camera to capture images of small components placed directly above it. The manual focus ring allows the focus of the camera to be specifically tuned to the design of the system, allowing for the best possible image quality. It also has a 5MP sensor[32], which is sufficient for the Computer Vision system, as high-resolution images would be preprocessed and reduced, only adding to the amount of processing required.

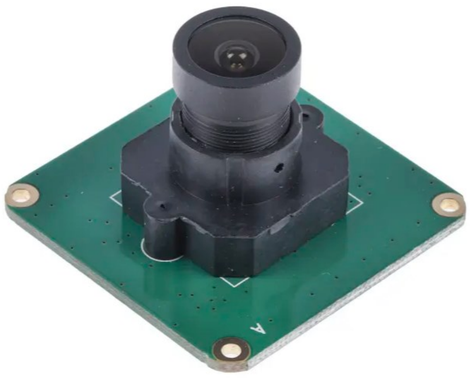


Figure 9: Okdo Adjustable Focus OV5647 Camera [31]

Other cameras were considered, such as the Raspberry Pi Camera Module V2 [16], however, it lacks a manual focus ring, which may result in a Minimum Object Distance (the closest distance at which an object can be captured) that is too far for the design of the system. Another camera, the Raspberry Pi High Quality Camera [15] also has the option of replacing the lens with a dedicated macro lens, however these are incredibly expensive.

3.2.4 WS2812B LEDs

To illuminate the components on the conveyor belt, a solution was required that could provide bright, controllable light. The WS2812B LEDs [47] were chosen as they are individually addressable, meaning that each LED can be controlled independently, allowing for a wide range of colours and patterns to be displayed. They are also relatively cheap and easy to source, and are compatible with the Raspberry Pi and have a large amount of support due to their popularity. They also can run off 5V, meaning it can share the input power rail with the Raspberry Pi and are also easy to control, requiring only a single GPIO pin to control many LEDs as they can work with either SPI or PWM signals, depending on the chosen library.

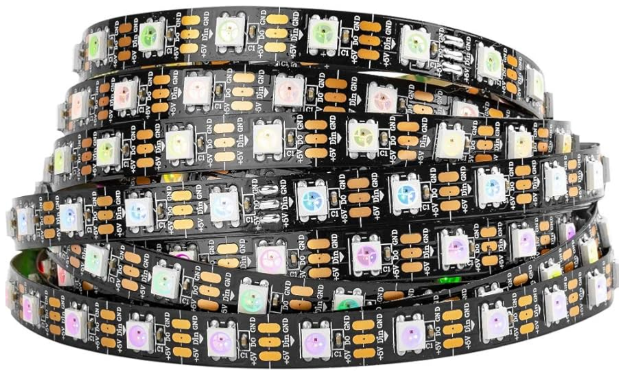


Figure 10: WS2812B LEDs [47]

Figure 11: 12V COB Ring Light

Originally, a 12V COB ring light was considered, as it could produce a uniform light source, however the specific ring was not dimmable and resulted in strange flickering when attempting to do so with a MOSFET. It also did not produce a clean white light, but a more blue light. The WS2812B LEDs were chosen as they could produce any light colour, and could be cut and placed in any configuration, allowing for a more customisable design.

3.2.5 Stepper Motors

Stepper motors possess the ability to move in precise increments, making them ideal for the Sweeper System which requires precise control. The Conveyor System is not as demanding, but stepper motors were chosen for consistency across the system. The NEMA17 series of stepper motors were chosen [22] as they are widely used and have a large amount of support and documentation, also typically used in FDM printers. With the ability to rotate at 1200RPM/900RPM at 12V, and a holding torque 48/63Ncm (for single/double stack motors), they are more than capable of driving the conveyor belt and sweeper arm.

Figure 12: NEMA17 Stepper Motor [22]

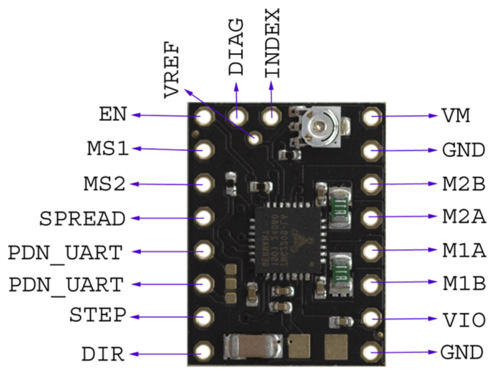


Figure 13: TMC2209 Stepper Motor Driver [4]

For driving the motors, TMC2209s were selected due to their microstepping support and silent operation. Due to their popularity in FDM printing and the ease of use, they have a large amount of support and documentation. They are also capable of driving the NEMA17 series of stepper motors, and are compatible with the Raspberry Pi. Originally, the DRV8825 was considered [33], however they added significant noise to the system and were unpleasant to use.

3.2.6 Break Beam Sensor

Due to the complexity of the system, it is vital to ensure that the system is as efficient as possible to ensure real-time performance. The design choice was made to use an IR Beam Break Sensor [20] to detect the presence of objects on the conveyor belt to reduce the computational overhead of constant inference on the Vision Handler. The System Controller connects directly to the IR Beam Break Sensor, which is used to detect the presence of objects on the conveyor belt, making use of interrupts to detect changes in the sensor's state. This allows for a fast response from the system without much computational overhead. The sensor itself does not change lighting conditions as it operates with infrared light, which is not visible to the human eye. The sensor is also easy to install and maintain, as it only requires a power supply and a digital input pin to operate.

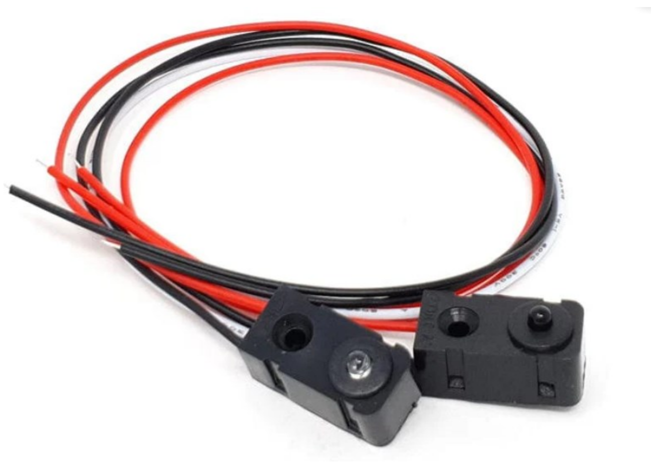


Figure 14: IR Beam Break Sensor [20]

An alternative to the IR Beam Break Sensor would be to use software approaches on the Vision Handler to detect the presence of objects on the conveyor belt, for example using OpenCV to detect large changes between successive frames. This could be implemented in a number of ways such as background subtraction or histogram comparison. However, this would also require the Vision Handler to constantly process frames, which would increase the computational overhead of the system. The IR Beam Break Sensor is a more efficient solution as it allows the system to only process images when an object is detected on the conveyor belt.

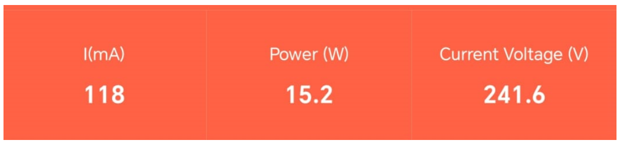
An approach to dynamically adjust the frame rate of the camera could be considered, increasing frame rates when there are no objects on the conveyor belt in anticipation of objects being placed on the conveyor belt, and reducing frame rates when objects are detected to reduce computational overhead. This could potentially impose resource contention between the Vision Handler and the System Controller while it is controlling the Sweeper and Conveyor Belt. The framerate also needs to take into consideration the conveyor speed to not miss objects, requiring both more complexity and computational resources.

An alternative to the IR Beam Break Sensor would be to use a proximity sensor to detect the presence of objects on the conveyor belt, however this requires constant polling which would be less efficient than using an interrupt-based approach. Additionally, the IR Beam Break Sensor is a more reliable solution as it is less prone to false positives than a software-based approach. The IR Beam Break Sensor is a more robust solution as it is less prone to interference from external factors such as lighting conditions or shadows.

For these reasons, the IR Beam Break Sensor was chosen as the best solution for detecting the presence of objects on the conveyor belt.

3.2.7 Power Supply Unit

To power the system, two parameters were taken into account when choosing a PSU; the power rating and voltage.



I(mA)	Power (W)	Current Voltage (V)
118	15.2	241.6

Figure 15: Power consumption of the system

Figure 16: 24V DC 6.25A Power Supply Unit [7]

Firstly, the power supply must be able to drive all components in the system with overhead in case of spikes in power consumption. The system consists of a Raspberry Pi 4 Model B, a DFRobot 7″ Touch-screen Display, two NEMA17 stepper motors and TMC2209 stepper motor drivers, an IR Beam Break Sensor, an Okdo Adjustable Focus OV5647 Camera, and 16 WS2812B LEDs.

In the early stages of the project, with only the Raspberry Pi, the DFRobot 7″ Touchscreen Display, WS2812B and Okdo Camera, the power consumption was measured to be under 20W, recorded using a smart plug with a power meter as shown in Figure 15. It was thought that the addition of the NEMA17 motors and TMC2209 stepper motors would contribute 50W (running at 2A 12V) to the power consumption, making for a power consumption of under 70W. A 30W overhead would have been added to the power supply, resulting in a 100W power supply. However, a 150W power supply was used instead due to an overestimation on the power consumption of the system before finalising the design. In hindsight, the 150W power supply was excessive, and a 100W power supply would have been sufficient.

add power consumption measurements from usb for complete power - implementation maybe?

Additionally, the voltage of the power supply must be greater than or equal to the highest voltage of any component in the system to enable the use of step down convertors to power the components. The highest voltage of any component in the system is 12V and the lowest is 5V. Initially, 24V components were a possibility due to the variety of LEDs and stepper motors, so a 24V power supply was chosen.

This resulted in selecting a 24V DC 6.25A Power Supply Unit.

3.2.8 Step Down Convertors

To power the components of the system, step-down convertors were used to convert the 24V output of the PSU to the required voltage of the components. The components that required a 12V supply were the NEMA17 stepper motors and TMC2209 stepper motor drivers, and the components that required a 5V supply were the Raspberry Pi, DFRobot 7″ Touchscreen Display, Okdo Adjustable Focus OV5647 Camera, and WS2812B LEDs.

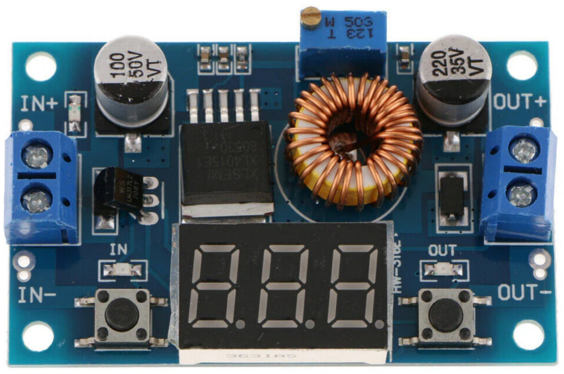


Figure 17: 24V to 12V Step Down Convertor [11]

The DFRobot and Camera is powered directly from the Pi, however it is necessary that the WS2812B LEDs are powered from the PSU to prevent the Pi from being overloaded. Each LED can consume up to 60mA at full brightness, and with 16 LEDs, the total current draw could be up to 960mA, which may be too much for the Pi to handle. Instead, the LEDs and Pi are connected to a USB hub that is powered by a step-down convertor at 5V, which is connected to the PSU. Likewise, the NEMA17 motor and TMC2209 stepper motor driver are powered by a step-down convertor at 12V.

The XL4015 High Power Step Down Convertor was chosen as it is capable of handling the current draw of the components, and is adjustable, allowing for fine-tuning of the output voltage. It is able to draw up to 75W, which is more than enough for the two 5V and 12V systems.

3.3 Mechanical Design and Electronics

As the system has a major physical component, the mechanical design is a crucial aspect of the project. The mechanical design is responsible for the physical structure of the system, including the conveyor system, the sweeper, bins, the housing for the LCD and PSU (Power Supply Unit), and the camera mount. It provides the foundation for the system to operate and interact with the environment, and as such it is important to ensure that the mechanical design is robust and reliable.

The system also has a major electronics component due to the requirement of moving parts, namely the conveyor and sweeper systems. Care must be taken to ensure that the power delivery to these systems is stable and reliable, and connections are secure. Careful consideration must also be taken to ensure electrical safety, preventing damage to the system and harm to the user.

The system has been designed in FreeCAD [17], a free and open-source parametric 3D CAD software. Aligning with the modular design approach of the project, the parametric nature of FreeCAD allows for easy modification of the design by modifying numerical parameters, which is useful in designs that require frequent changes. It was decided to use FDM printing to produce the components of the system using PLA, a biodegradable thermoplastic derived from renewable resources such as corn starch or sugarcane [13]. PLA is easy to print, and does not require ventilation or post-processing, unlike other materials like ABS. The known brittleness and low heat resistance of PLA are not a concern for this project, as the system is not exposed to high temperatures or direct mechanical stress. An alternative to PLA would be PETG, which has higher heat resistance and is less brittle than PLA, however, it is more difficult to print and is more prone to warping - it is also less environmentally friendly than PLA and less cost-effective.

All components are designed to be easily assembled and disassembled, allowing for easy maintenance and repair. They are easily assembled using standard M3 screws and brass inserts, which are easy to source and replace.

3.3.1 Conveyor

As discussed in the Background (Section 2), a conveyor belt system with a face down camera seems to be the most promising approach. For the design of the conveyor system, it is imperative that the following requirements are met:

- The conveyor belt must be able to move objects from the input to the output.
- The conveyor belt must be able to move objects at a consistent speed.
- The conveyor belt must be able to move objects in a controlled manner.
- The belt must be detachable for maintenance and repair.
- The conveyor must include a tensioning mechanism to ensure the belt is taut.
- The conveyor must include a mount for the break beam sensor.
- The design of the conveyor must mount the stepper motor that drives it.
- The idler that allows the belt to move must be mounted and is free to rotate to reduce friction and ensure smooth operation.
- The actively driven roller must be easily coupled and decoupled from the stepper motor.

The belt will be made of thick paper as it is cost-effective, easy to replace, and is reasonably rough and can withstand tension forces. It can also be in different colours which may help with contrast in the images for the Vision Handler. Other alternatives to paper would be rubber or silicone, however, these are more expensive and difficult to source and cut to size. Alternatively an FDM printed belt could be used, but this would require more maintenance and may lead to louder operation.

The foundation of the conveyor system will be 2020 aluminium extrusion, which is lightweight, strong, and cost-effective. Custom designed FDM printed PLA parts will be used to fulfil the requirements of the conveyor system. To facilitate smooth rotation on the idlers and driven roller, ball bearings will be used. A NEMA17 stepper motor will be used to drive the driven roller, as it is powerful enough to drive the belt and is easy to source; torque is not a concern as the belt is lightweight and the speed of the NEMA17 series of stepper motors is sufficient to not require a reduction gear. The stepper motor will be controlled by a TMC2209 stepper motor driver, a silent, and efficient stepper motor driver that is easy to use and is compatible with the NEMA17 stepper motor.

NEMA17 stepper motors typically require 12V, meaning a separate means of delivering power to the motors from the Pi is required.

include nema17 and tmc2209 specs

3.3.2 Sweeper

To sort components travelling along the belt, many approaches were considered.

One such approach would be compressed air to blow components off the conveyor belt into bins as seen in commercial systems. However, this approach is not feasible as it would require a large and potentially loud air compressor to generate the compressed air, as well as a system of nozzles and valves. Alternatively a single nozzle and valve with a moving arm could be used, however this poses the same issues.

Another approach could be to use a series of actuated arms to push components off the conveyor belt into bins. The arms would be designed in such a way where they have two states; either open or closed, making them operate more like a gate. This approach would require a series of actuators and a complex control system to ensure that these gates are in the correct position at the correct time. Each gate would need a motor which would quickly become expensive and difficult to route with the limited GPIO pins on the Raspberry Pi.

An improvement to this design would be to instead carefully design the gates such that the end position of their rotation is either in the open or closed state. An arm on a moving track with a flexible rod could then be used to rotate the arms as it travelled past, effectively allowing the system to carefully control which bin the component goes into. This would require a single motor and a carefully designed arm, which would be more cost-effective and easier to control. This is an approach that is discussed in Section 4, but was not chosen for the final design due to the next approach.

The approach that was taken was to simply attach a static arm to a moving carriage that ran alongside the conveyor belt. The arm is angled, allowing the movement of the conveyor belt to push components off the belt and into the bins. This approach does not require intricately designed gates or complex control systems, and is simple and cost-effective. It requires the use of a NEMA17 stepper motor and an endstop for precise control of the arm's position, and a TMC2209 stepper motor driver to control the stepper motor.

Again, the issue of power delivery arises, as the NEMA17 stepper motor requires 12V.

3.4 Computer Vision System

To identify and classify components on the conveyor belt, a Computer Vision system is required. As discussed in the Background (Section 2), the approach taken will be to use the YOLOv8 object detection model, due to its notable inference speed, performance, and ability to optimise to deploy on the Raspberry Pi.

3.4.1 Image Processing

talk about camera distortion correction

YOLOv8's training framework [45] allows image augmentation to be applied to the dataset in-flight, which can be used to artificially increase the size of the dataset and improve the model's performance. This can be used to apply transformations to the images such as rotation, scaling, and flipping, as well as more complex transformations such as perspective warping. This can be used to artificially increase the size of the dataset and improve the model's performance.

It is important to select data augmentation schemes that would result in the images being in the same domain as the images that the model will be inferring on. For example, grayscale images should not be used as the model will be inferring on colour images. Scaling augmentations should be used sparingly as to not remove the component from the image, while rotation augmentations should be used to ensure that the model is invariant to the orientation of the component.

For the task of component identification, the following augmentations were selected:

Augmentation	**Value**	**Reasoning**
HSV_H	0.05	This determines the range of hue values that the image can be augmented by. A 5% change in hue can represent lighting conditions changing, which is important for the model to be invariant to. Changing it too much will result in the training data to not be in the same domain as the inference data.
HSV_S	0.3	This determines the range of saturation values that the image can be augmented by. Justification is the same as above.
HSV_V	0.2	This determines the range of value (degree of whiteness) values that the image can be augmented by. Justification is the same as above.
Degrees	180	This determines the range of rotation values that the image can be augmented by. Allowing any degree of rotation is important as the components may be placed on the conveyor belt at any orientation.
Translate	0.1	This determines the range of translation values that the image can be augmented by. Allowing a 10% translation is important as the components may not be placed in the centre of the image, however too much translation may result in the component being cut off.
Scale	0.8	This determines the minimum scaling factor that the image can be augmented by. This provides robustness to the specific mounting height of the camera.
Shear	10.0	This determines the range of shear values that the image can be augmented by. Shearing is important as the components may not be placed in the centre of the image.
Perspective	0.0	This determines the range of perspective values that the image can be augmented by. Perspective warping is not required as the camera is always placed directly above the conveyor belt.
Flipud	0.5	This determines the probability that the image will be flipped up-down. Flipping the image up-down is important as the components may be placed on the conveyor belt in any orientation.
Fliplr	0.5	This determines the probability that the image will be flipped left-right. Justification is the same as above.
Mosaic	0.5	This determines the probability that the image will be augmented by a mosaic. According to YOLO [45], this greatly improves model performance.

Table 1: Data Augmentation Parameters

3.4.2 Real-time Object Detection

To perform real-time object detection, it is necessary to deploy a model with low inference latency, especially for the Raspberry Pi where computational resources are scarce. As discussed in the Background (Section 2), the YOLOv8 object detection model was initially chosen due to the ability to be optimised by optimisation libraries DeepSparse [23] and SparseML [25] to run on the Raspberry Pi. Pretrained 'sparsified' models are available for the YOLOv8 model, which have been pruned and quantised to reduce the number of parameters and the amount of computation required to run the model, preventing the need to 'sparsify' the model manually. In the event that the model could not be successfully deployed using these libraries, the model could have been run normally at a reduced frame rate, or could have been hosted on the cloud and accessed via an API — though this would have introduced latency and would have required an internet connection. However, during the implementation phase, the typical YOLOv8 model not found to be suitable for the task.

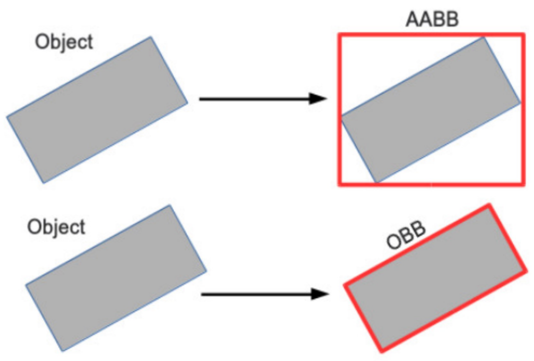


Figure 18: OBB vs AABB Bounding Boxes [3]

Common Computer Vision architectures typically draw AABBs (Axis Aligned Bounding Boxes) when detecting the object in the camera frame, as shown in Figure 18. However, for the specific task of component identification it is more useful to draw OBBs (Oriented Bounding Boxes) which aligns with the orientation of the component, allowing the component to be cropped out and further inspected, for example in the case of value identification. There is a variant of the YOLO architecture that supports OBBs, called YOLOv8-obb [45], which will be used for this task.

Model	**mAP^{50}**	**Speed CPU ONNX (ms)**	**Params (M)**	**FLOPs (B)**
YOLOv8n-obb	78.0	204.77	3.1	23.3
YOLOv8s-obb	79.5	424.88	11.4	76.3
YOLOv8m-obb	80.5	763.48	26.4	208.6
YOLOv8l-obb	80.7	1278.42	44.5	433.8
YOLOv8x-obb	81.36	1759.10	69.5	676.7

Table 2: YOLOv8-obb Model Performance [45]

The OBB models come pretrained on the DOTOv1 dataset [10], which contains 15 classes of objects, including components. Pretrained models are useful as they have already learnt useful features from a large dataset, and can be fine-tuned on a smaller dataset to improve performance [12]. The performance of the different sized YOLOv8-obb models is shown in Table 2. The most crucial parameters to take into consideration is the mAPval (mean average precision) and the CPU ONNX (inference latency). The mean average precision is defined by the following formula:

$$\text{mAP} = \frac{1}{n}\sum_{i=1}^{n} \text{AP}_i$$

Where n is the number of classes, and $\texttt{AP}_i$ is the average precision for class i. The average precision is defined as the area under the precision-recall curve, and is an objective measure of the model's performance. For mAP^{50}, the IoU (Intersection over Union) threshold is defined as being 50%, meaning that as long as there is at least 50% overlap between the predicted bounding box and the ground truth bounding box, the prediction is considered correct. In the formula below, A is the ground truth bounding box, and B is the predicted bounding box, with A ∩ B being the intersection of the two bounding boxes, and A ∪ B being the union of the two bounding boxes.

$$\text{IoU} = \frac{\text{A} \cap \text{B}}{\text{A} \cup \text{B}}$$

From Table 2, the YOLOv8x-obb model has the best performance, with a mAP^{50} of 81.36 and a CPU ONNX of 1759.10ms. The YOLOv8n-obb model has the worst performance, with a mAP^{50} of 78.0 and a CPU ONNX of 204.77ms. However, the Pi is likely significantly slower than the CPU used in the benchmarks, so the actual inference latency will be higher, making the 0.5 FPS of the YOLOv8x-obb model infeasible, but the 5 FPS of the YOLOv8n-obb model more promising, a 9x improvement in speed for only a 3.1% decrease in mAP^{50}.

As mentioned in Subsubsection 3.2.6, the system will use a break beam sensor to detect the presence of objects on the conveyor belt, reducing the potential of the Vision Handler to be a bottleneck due to constant inference. With the promising performance increase of using the DeepSparse [23] and SparseML [25] libraries to optimise the model for the Raspberry Pi, the YOLOv8n-obb model will be used for the project.

3.4.3 Dataset Annotation

To train the YOLOv8-obb model, a dataset of images of components is required. The dataset must be annotated with the bounding boxes of the components in the images, which can be done using a tool such as Roboflow [40]. However, the components will only be visible from the Raspberry Pi, so it can become tedious to take images from the Pi, transfer them to a computer, annotate them, and repeat for every distinct value within the component and then for every type of component.

Instead, the decision was made to design a custom dataset labelling tool that could directly capture images from the Raspberry Pi, annotate them, and save them locally for training. The dataset would be correctly structured for training with YOLO, facilitating the training process.

3.5 Software

The software of the system is responsible for controlling the hardware components, processing images from the camera, and interfacing with the user. The software is divided into several subsystems, each responsible for a different aspect of the system. The software is designed to be modular, with each subsystem exposing a high-level interface to other subsystems, allowing for easy integration and maintenance.

Each component has the capability to run independently to facilitate debugging and maintenance, and communicates with the System Controller to receive commands. The System Controller is responsible for managing the communication between the components, and is the central hub of the system.

The software is written in Python [36], due to its ease of use and large amount of support and libraries available, and will be run entirely on the Raspberry Pi. Git [18] is used for version control to ensure that changes to the software can be tracked and reverted if necessary, Pylint [35] is used for code linting to ensure that the code is clean and readable, RealVNC [38] is used for remote access to the Raspberry Pi, and Visual Studio Code [6] is used as the IDE for development. As mentioned in Subsection 3.3, FreeCAD [17] has been used for the mechanical design of the system.

3.5.1 LCD UI

The LCD UI is responsible for displaying the status of the system, such as the current state of the system, the status of the conveyor belt and sweeper, and the classification of components. It also allows the user to interact with the system, such as starting and stopping the system, and manually controlling the conveyor belt and sweeper. The LCD UI communicates with the System Controller to receive status updates and send commands.

The LCD UI is written with Pygame [34] and Pygame GUI [27], a Python library for creating graphical user interfaces. Pygame GUI builds upon the Pygame library, providing UI elements so that they do not need to be created from scratch. Pygame also provides camera support, allowing the Vision Handler to receive frames for inference or displaying the feed on the LCD UI. It also allows for precise control over what is being drawn and the event loop, allowing for a responsive UI.

3.5.2 Data Annotation Tool

As mentioned in Subsubsection 3.4.2, a custom dataset labelling tool was designed to capture images from the Raspberry Pi, annotate them, and save them locally for training. The tool is responsible for capturing images from the camera, displaying them to the user, allowing the user to draw bounding boxes around the components, and saving the images and annotations to disk.

This tool was written using CustomTkinter [41], a modern fork of the Tkinter library that provides a more modern look and feel, and is more customisable. CustomTkinter provides a high-level interface for creating GUIs, allowing for easy creation of UI elements and event handling. Draw loops are not needed as the UI is event-driven, allowing for easy development.

4 Implementation

5 References

[1] Maria Concepcion Abe, Gabriel Angelo Gelladuga, Chirstine Joy Mendoza, Jesseth Mae Natavio, Jeanella Shaine Zabala, and Edgar Clyde R. Lopez. Pneumatic conveying technology: Recent advances and future outlook. *Engineering Proceedings*, 56(1), 2023. ISSN 2673-4591. doi: 10.3390/ASEC2023-16267. URL https://www.mdpi.com/2673-4591/56/1/205.

[2] Alasdair Allan. Benchmarking the intel neural compute stick on the new raspberry pi 4, model b. https://aallan.medium.com/benchmarking-the-intel-neural-compute-stick-on-the-new-raspberry-pi-4-model-b-e419393f2f97, 08-2019. Accessed: 06-02-2024.

[3] Namas Bhandari. Aabb obb photo. https://www.namasbhandari.in/post/the-era-of-oriented-bounding-boxes, n.d. Accessed: 01-06-2024.

[4] BIGTREETECH. Bigtreetech tmc2209 stepper motor driver. https://botland.store/stepstick-stepper-motor-controllers/19883-bigtreetech-tmc2209-v13-stepper-motor-driver-5904422379667.html, n.d. Accessed: 24-05-2024.

[5] Praneel Chand and Sunil Lal. Vision-based detection and classification of used electronic parts. *Sensors*, 22(23), 2022. ISSN 1424-8220. doi: 10.3390/s22239079. URL https://www.mdpi.com/1424-8220/22/23/9079.

[6] Visual Studio Code. Visual studio code. https://code.visualstudio.com/, n.d. Accessed: 22-01-2024.

[7] RS Components. Delta electronics power supply, pmt-24v150w2ba, 24v dc, 6.25a, 150w, 1 output, 90 132v ac input voltage. https://uk.rs-online.com/web/p/switching-power-supplies/2411652?gb=s, n.d. Accessed: 24-05-2024.

[8] DFRobot. 7 inch hdmi display with usb touchscreen. https://www.farnell.com/datasheets/3162025.pdf, n.d. Accessed: 19-01-2024.

[9] Amol I. Dhenge, Nāgpur, and A. S. Khobragade. Mechanical nut-bolt sorting using principle component analysis and artificial neural network. In *n.a*, 2013. URL https://api.semanticscholar.org/CorpusID:201742258.

[10] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/TPAMI.2021.3117983.

[11] Eelectronicparts. Xl4015 5a high power 75w dc-dc adjustable step-down module+led voltmeter power supply module. https://www.eelectronicparts.com/products/xl4015-5a-high-power-75w-dc-dc-adjustable-step-down-module-led-voltmeter-power-supply-module, n.d. Accessed: 24-05-2024.

[12] encord. Pre trained model. https://encord.com/glossary/pre-trained-model-definition/, n.d. Accessed: 01-06-2024.

[13] FormLabs. Guide to 3d printing materials: Types, applications, and properties. https://formlabs.com/uk/blog/3d-printing-materials/, n.d. Accessed: 24-05-2024.

[14] Raspberry Pi Foundation. Raspberry pi 4 model b. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/, n.d. Accessed: 19-01-2024.

[15] Raspberry Pi Foundation. Raspberry pi high quality camera. https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/, n.d. Accessed: 24-05-2024.

[16] Raspberry Pi Foundation. Raspberry pi camera module v2. https://www.raspberrypi.com/products/camera-module-v2/, n.d. Accessed: 24-05-2024.

[17] FreeCAD. Freecad. https://www.freecadweb.org/, n.d. Accessed: 19-01-2024.

[18] Git. Git. https://git-scm.com/, n.d. Accessed: 22-01-2024.

[19] Ce Guo, Xiao ling Lv, Yan Zhang, and Ming lu Zhang. Improved yolov4-tiny network for real-time electronic component detection. *Scientific Reports*, 11(1):22744, 2021. doi: 10.1038/s41598-021-02225-y. URL https://doi.org/10.1038/s41598-021-02225-y.

[20] The Pi Hut. Break beam sensor. https://thepihut.com/products/ir-break-beam-sensor-3mm-leds, n.d. Accessed: 24-05-2024.

[21] Sigitas Kilikevičius and Algimantas Fedaravičius. Vibrational transportation on a platform subjected to sinusoidal displacement cycles employing dry friction control. *Sensors*, 21(21), 2021. ISSN 1424-8220. doi: 10.3390/s21217280. URL https://www.mdpi.com/1424-8220/21/21/7280.

[22] PBC Linear. Nema 17 stepper motor. https://pages.pbclinear.com/rs/909-BFY-775/images/Data-Sheet-Stepper-Motor-Support.pdf, n.d. Accessed: 24-05-2024.

[23] Neural Magic. Deepsparse. https://github.com/neuralmagic/deepsparse, n.d.

[24] Neural Magic. Yolov8 detection 10x faster with deepsparse—over 500 fps on a cpu. https://neuralmagic.com/blog/yolov8-detection-10x-faster-with-deepsparse-500-fps-on-a-cpu/, n.d.

[25] Neural Magic. Sparseml. https://github.com/neuralmagic/sparseml, n.d.

[26] Mia Muminovic and Emir Sokic. Automatic segmentation and classification of resistors in digital images. In *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–6, 2019. doi: 10.1109/ICAT47117.2019.8939034.

[27] MyreMylar. Pygame gui. https://github.com/MyreMylar/pygame_gui/tree/0cbf7056518377b455d51a8d20167f4029756ad9, n.d. Accessed: 22-01-2024.

[28] Le Nam, Nguyen Mui, and Dang Tu. A method to design vibratory bowl feeder by using fem modal analysis. *Vietnam Journal of Science and Technology*, 57:102, 02 2019. doi: 10.15625/2525-2518/57/1/12859.

[29] NVIDIA. Jetson nano. https://developer.nvidia.com/embedded/jetson-nano-developer-kit, n.d. Accessed: 06-02-2024.

[30] University of Bristol. Leaf green lab certification. https://www.imperial.ac.uk/sustainable-imperial/resource-management/energy-use/laboratory-efficiency-assessment-framework-leaf/#:~:text=In%202021%2D22%2C%20we%20awarded,leaving%20procedures%20were%20in%20place, n.d. Accessed: 06-02-2024.

[31] Okdo. Okdo, camera module, csi-2 with 2592 x 1944 resolution. https://uk.rs-online.com/web/p/raspberry-pi-cameras/2020456/, n.d. Accessed: 19-01-2024.

[32] Okdo. Okdo, camera module, csi-2 with 2592 x 1944 resolution specification. https://docs.rs-online.com/b064/A700000006917308.pdf, n.d. Accessed: 19-01-2024.

[33] Pololu. Pololu drv8825 stepper motor driver. https://www.pololu.com/product/2133, n.d. Accessed: 24-05-2024.

[34] Pygame. Pygame documentation. https://www.pygame.org/docs/, n.d. Accessed: 22-01-2024.

[35] Pylint. Pylint. https://github.com/pylint-dev/pylint, n.d. Accessed: 19-01-2024.

[36] Python. Python. https://www.python.org/, n.d. Accessed: 19-01-2024.

[37] Erwin Quilloy, C. Delfin, and M. Pepito. Single-line automated sorter using mechatronics and machine vision system for philippine table eggs. *African Journal of Agricultural Research*, 13: 918–926, 04 2018. doi: 10.5897/AJAR2018.13113.

[38] RealVNC. Realvnc. https://www.realvnc.com/en/, n.d. Accessed: 22-01-2024.

[39] Gunther Reinhart and Michael Loy. Design of a modular feeder for optimal operating performance. *CIRP Journal of Manufacturing Science and Technology*, 3(3):191–195, 2010. ISSN 1755-5817. doi: https://doi.org/10.1016/j.cirpj.2010.09.003. URL https://www.sciencedirect.com/science/article/pii/S1755581710000908.

[40] Roboflow. Roboflow. https://roboflow.com/, n.d. Accessed: 01-06-2024.

[41] Tom Schimansky. Custom tkinter widgets. https://github.com/TomSchimansky/CustomTkinter, n.d. Accessed: 25-01-2024.

[42] Richard Silversides, Jian S Dai, and Lakmal Seneviratne. Force Analysis of a Vibratory Bowl Feeder for Automatic Assembly. *Journal of Mechanical Design*, 127(4):637–645, 08 2004. ISSN 1050-0472. doi: 10.1115/1.1897407. URL https://doi.org/10.1115/1.1897407.

[43] Pertiwang Sismananda, Maman Abdurohman, and Aji Gautama Putrada. Performance comparison of yolo-lite and yolov3 using raspberry pi and motioneyeos. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–7, 2020. doi: 10.1109/ICoICT49345.2020.9166199.

[44] Juan Terven and Diana-Margarita Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *n.j*, 04 2023.

[45] Ultralytics. Yolov8. https://github.com/ultralytics/ultralytics, n.d. Accessed: 06-02-2024.

[46] Feeder University. Vibratory feeder basics. https://www.youtube.com/watch?v=m27oD1wfQ0Y, n.d. Accessed: 06-02-2024.

[47] Worldsemi. Ws2812b datasheet. https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf, n.d. Accessed: 24-05-2024.

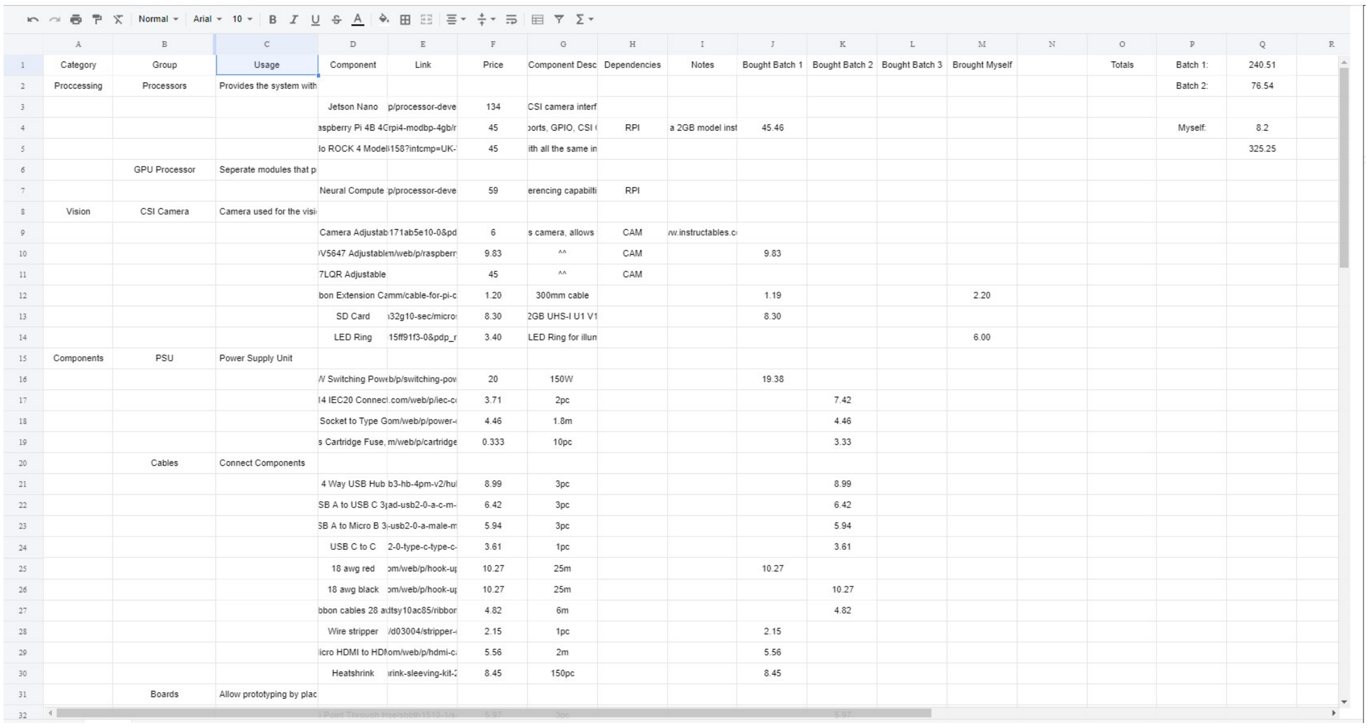
[48] Yuanyuan Xu, Genke Yang, Jiliang Luo, Jianan He, and Chenxi Huang. An electronic component recognition algorithm based on deep learning with a faster squeezenet. *Mathematical Problems in Engineering*, 2020:2940286, 2020. doi: 10.1155/2020/2940286. URL https://doi.org/10.1155/2020/2940286.

[49] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):8934–8954, September 2023. ISSN 2326-3865. doi: 10.1109/tkde.2022.3220219. URL http://dx.doi.org/10.1109/TKDE.2022.3220219.

[50] Department of Mechanical Engineering Zhengyang Zhang, MIT. *Design and Development of an Automated Sorting and Orienting Machine for Vials*. Massachusetts Institute of Technology, Department of Mechanical Engineering, 2019. URL https://books.google.co.uk/books?id=Mi5WzQEACAAJ.

6 Appendix

Bill of Materials



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Category	Group	Usage	Component	Link	Price	Component Desc	Dependencies	Notes	Bought Batch 1	Bought Batch 2	Bought Batch 3	Brought Myself		Totals	Batch 1:	240.51	
2	Proccessing	Processors	Provides the system with													Batch 2:	76.54	
3				Jetson Nano	p/processor-deve	134	CSI camera interf											
4				aspberry Pi 4B 4G	rpi4-modbp-4gb/r	45	ports, GPIO, CSI (	RPI	a 2GB model inst	45.46						Myself:	8.2	
5				lo ROCK 4 Model	158?intcmp=UK-	45	ith all the same in										325.25	
6		GPU Processor	Seperate modules that p															
7				Neural Compute	p/processor-deve	59	erencing capabilti	RPI										
8	Vision	CSI Camera	Camera used for the visi															
9				Camera Adjustab	171ab5e10-0&pd	6	s camera, allows	CAM	w.instructables.c									
10				V5647 Adjustable	m/web/p/raspberr	9.83	^^	CAM		9.83								
11				7LQR Adjustable		45	^^	CAM										
12				bon Extension Ca	mm/cable-for-pi-c	1.20	300mm cable			1.19			2.20					
13				SD Card	32g10-sec/micro	8.30	2GB UHS-I U1 V1			8.30								
14				LED Ring	15ff91f3-0&pdp_r	3.40	LED Ring for illun						6.00					
15	Components	PSU	Power Supply Unit															
16				V Switching Pow	b/p/switching-pow	20	150W			19.38								
17				14 IEC20 Connec	.com/web/p/iec-c	3.71	2pc				7.42							
18				Socket to Type G	om/web/p/power-	4.46	1.8m				4.46							
19				s Cartridge Fuse,	m/web/p/cartridge	0.333	10pc				3.33							
20		Cables	Connect Components															
21				4 Way USB Hub	b3-hb-4pm-v2/hul	8.99	3pc				8.99							
22				SB A to USB C 3	ad-usb2-0-a-c-m-	6.42	3pc				6.42							
23				SB A to Micro B 3	-usb2-0-a-male-m	5.94	3pc				5.94							
24				USB C to C	2-0-type-c-type-c-	3.61	1pc				3.61							
25				18 awg red	om/web/p/hook-u	10.27	25m			10.27								
26				18 awg black	om/web/p/hook-u	10.27	25m				10.27							
27				bbon cables 28 a	dtsy10ac85/ribbon	4.82	6m				4.82							
28				Wire stripper	/d03004/stripper-	2.15	1pc			2.15								
29				icro HDMI to HD	om/web/p/hdmi-c	5.56	2m			5.56								
30				Heatshrink	rink-sleeving-kit-2	8.45	150pc			8.45								
31		Boards	Allow prototyping by plac															
32				Point Through Ho	e/sbbth1510-1/s-	5.97	3pc				5.97							

Figure 19: Bill of Materials

FreeCAD System Design

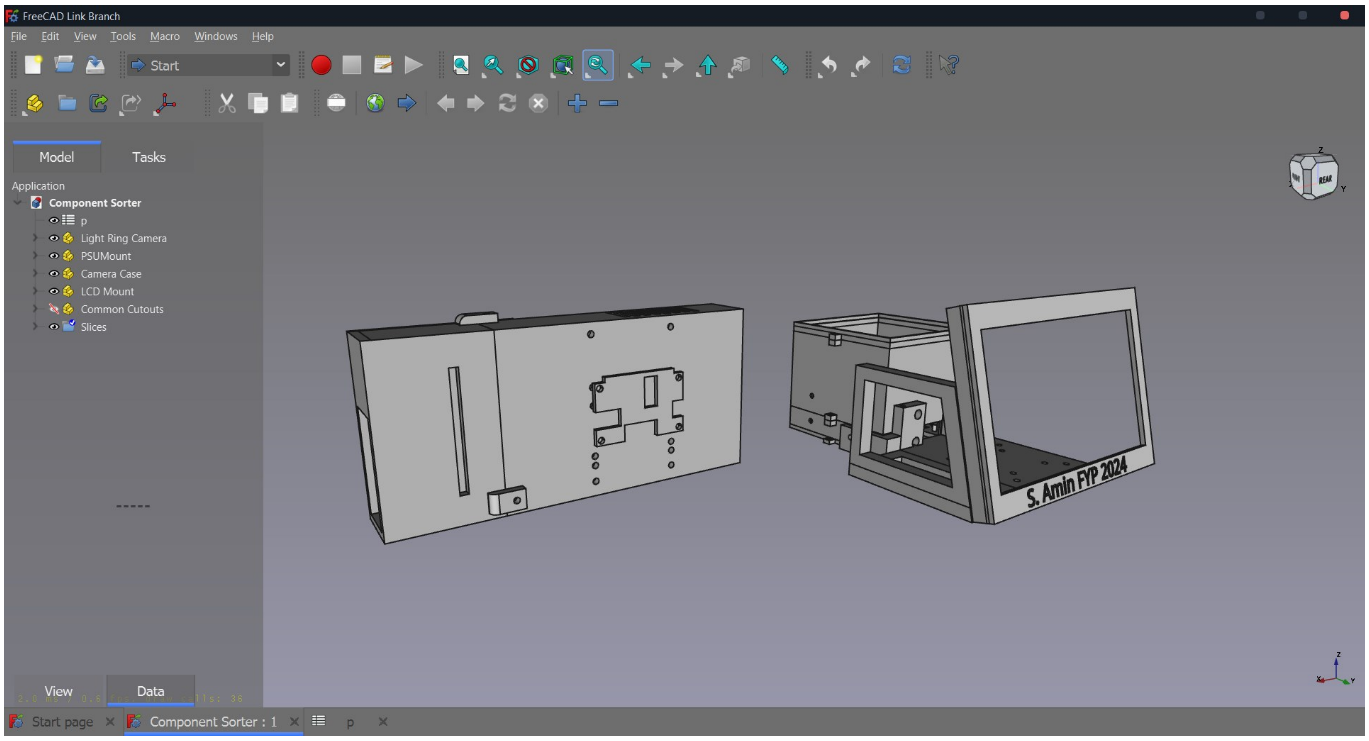


Figure 20: Stage 1 FreeCAD System Design

Raspberry Pi Benchmarking

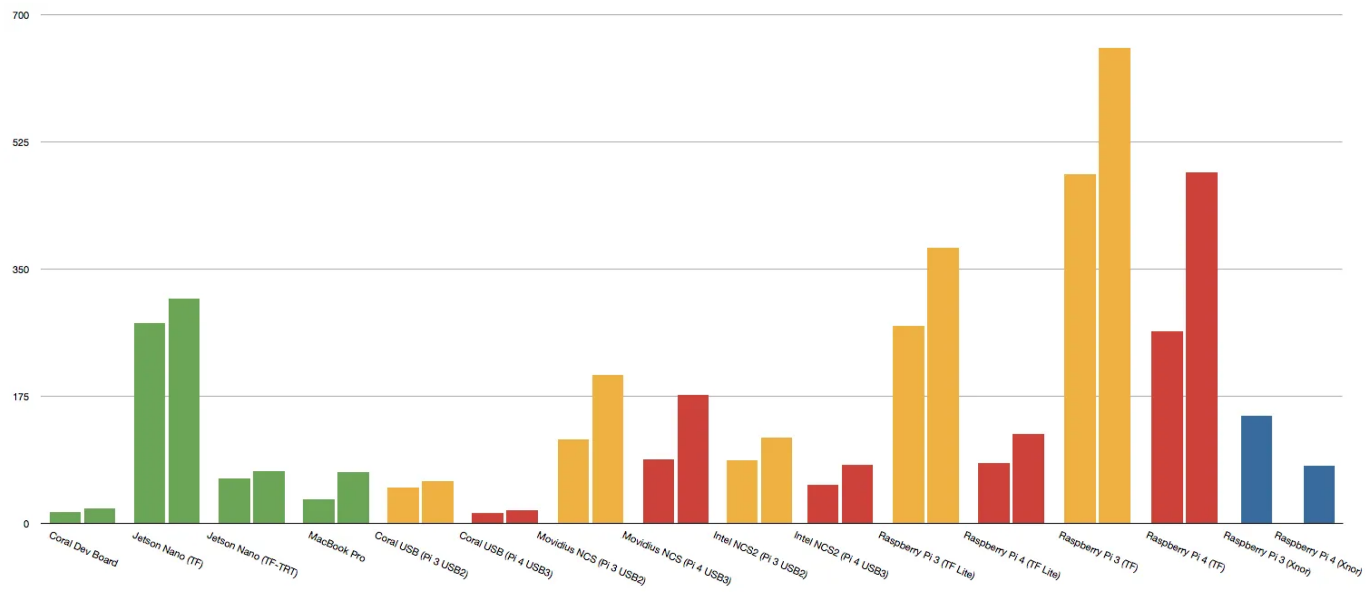


Figure 21: Raspberry Pi Benchmarking