Interim Report

Shaheen Amin, 01866464 Dr. Edward Stott, Supervisor

December 2023

Abstract & Plagirism Declaration

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. Utilizing Computer Vision, the system will classify components like resistors, capacitors, ICs, and MOSFETs, and sort them into designated bins. The project will be executed in three stages: initial development of a Computer Vision system for component identification, integration of a semi-automated sorting mechanism, and potentially, full automation of the sorting process or additional features like IC testing. Supervised by Dr. Edward Stott, this project aims to practically solve the problems faced by the Level 1 Labs.

I affirm that I have provided explicit references for all the material in my Interim Report that is not authored by me, but is represented as my own work. I have used GPTv4, as an aid in the preparation of my report. It was used for LaTeX formatting, however all technical content is original.

Contents

1 Project Specification

This project aims to solve the problem of electronic waste and cluttered workspaces in the Level 1 Labs by developing an automated system for identifying and sorting electronic components.

It aims to do so 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:

- A Computer Vision system for component identification
- An interface from which to observe and control the system's state
- A semi-automated sorting mechanism
- Potentially, full automation of the sorting process or additional features like IC testing

After initial consultation with my supervisor Dr. Stott and a visit to the Level 1 Labs to discuss the project, I received a sample of the different electrical components that were commonly used, which are the following:

- Resistors
- Capacitors
- Ceramic Capacitors
- Inductors
- Diodes
- MOSFETs
- Transistors
- LEDs
- Wires
- ICs

For this initial stage in the project, the focus is on developing the foundation for the system to be

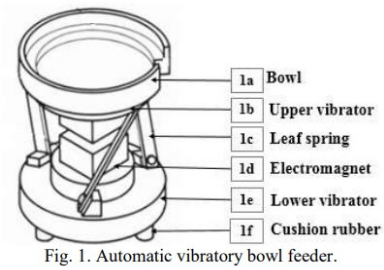

Figure 1: VBF[21]

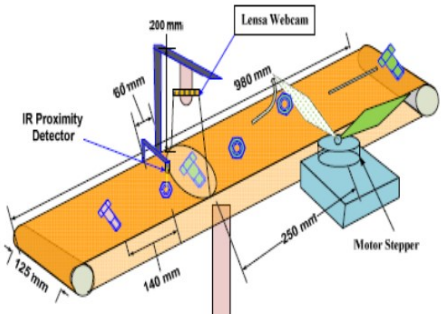

Figure 2: Conveyer Belt[6]

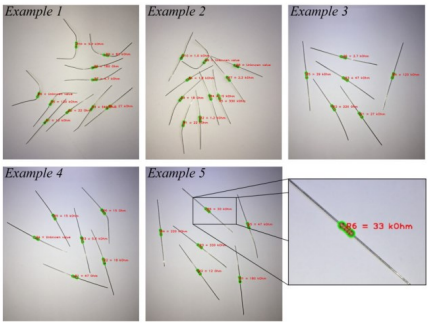

Figure 3: Resistor Test Set[19]

built upon so that the development of the various systems can be done independently, in parallel, and in such a way that each system is modular and easily extensible. For this reason, the project is split into three stages:

- Initial development of a Computer Vision system for component identification
- Integration of a semi-automated sorting mechanism
- Potentially, full automation of the sorting process or additional features like IC testing

This approach was taken to allow me to set defined goals to ensure that the project is always working towards a deliverable product. Each stage extends the previous stage, and so the project can be considered as a series of smaller projects.

2 Background

For the background of this project, it is necessary to consider literature related to the following:

- Mechanical design of existing sorting machines
- Existing Computer Vision systems for component identification
- Real-time Computer Vision architectures

2.1 Mechanical Design

Bowl Feeders

In my research, I have discovered that most industrial sorting machines use a Vibratory Bowl Feeder (VBF) to feed components into the system; as shown in Figure 1, the VBF consists of a bowl that vibrates coupled with a spring and electromagnet to feed components into the system. The paper by Nam et al.[21] 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 comprehsive overview of the design of a VBF, and provides a good starting point for the design of the VBF for in the future when the project is extended to include a VBF 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[29] delves into the mathematical modelling of a VBF, focusing more on the overall performance of the VBF rather than efficiency, and Silversides et al.[31] provides a good overview of the forces involved in the operation of a VBF, stengthening the basis for its design and viability.

The paper by Zhang and of Technology. Department of Mechanical Engineering[37] 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 is not applicable to this project, however it does provide a good insight into the design of a sorting system.

It seems conclusive that a VBF is the most viable option for the feeding mechanism of the sorting system, and so the design of the VBF will be based on the research outlined above. **Transport Mechanisms**

The paper by Dhenge et al.[6] depicts a conveyer 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 2, 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 seperate 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.[27] to sort Philippine table eggs also features a similiar conveyer belt system, down facing camera and an arm to sort the eggs into different categories.

The approaches taken here are similiar to industrial sorting systems that I have come across in videos during my research; and this approach seems to answer two major design questions for the project, namely how to transport the components to the computer vision system, and how to transport the components from the computer vision system to the sorting system.

The use of a converyer belt system is obvious, but the concept of having the camera facing down at the components is not one that I had considered initially, as for the first stage I wanted the system to be semi-autonomous; having the camera face-up allows the user to simply place the component on a plate that was directly above the camera, and the component would be immediately identified. Having the camera face down would block the user's view of the component, which would be a major inconvenience. However, it seems that this would be the most viable approach for the future of the project.

2.2 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.[6] to more modern Computer Vision techniques Chand and Lal[1], Xu et al.[35]. 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 Network) as used by Dhenge et al.[6] is a relatively simple but outdated statistical technique that was successful in identifying nuts and bolts, which are very distinct components. In this paper, PCA is used as a feature extractor and then fed directly into, though not explicitly mentioned, an FCL (Fully Connected Layer), as the classifier. Although a valid approach, CNNs (Convolutional Neural Networks) have been shown 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.[35], that PCA may suffer from, and so this approach is not viable for the project.

The paper by Xu et al.[35] use the SqueezeNet CNN architecture to identify 22 different subcategories of electronic components, specifically resistors, capacitors, and inductors, and acheives a True Positive Rate (TPR) of 99.999% with only a 2.67ms average inference time on a GTX 1050 2GB GPU, released in 2016, an impressive result. This informs the design of the Computer Vision system, as it shows that a CNN is a viable approach to the problem of component identification.

The paper by Muminovic and Sokic[19] dicuss the use of SVM 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 is very distinct from other components, given the 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 3, 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 very promising for the project, and will inform the design of the Computer Vision system.

2.3 Real-time 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 conveyer 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 single frame, and not require multiple frames to identify a component. As I envision the system to be self-contained, it must also be able to run on the Raspberry Pi 4, which has a 1.5GHz quad-core

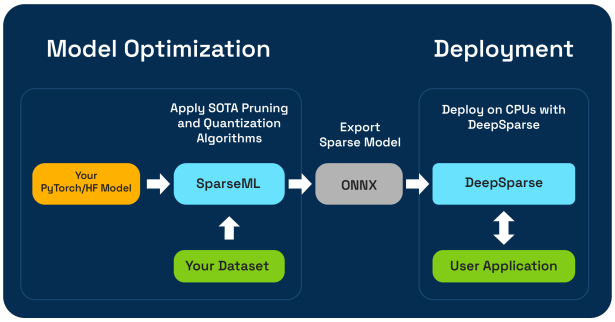

Figure 4: SparseML Pipeline[18]

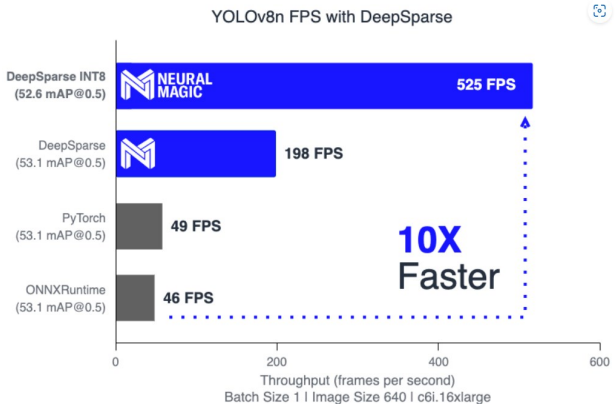

Figure 5: DeepSparse Performance[17]

64-bit ARM Cortex-A72 CPU, and 2GB of RAM. The system will also display a camera feed to the user, and so the Computer Vision system must be able to run in real-time alongside the camera feed.

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[1] 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 has been shown to be very effective at identifying objects in real-time Terven and Cordova-Esparza[33]. 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.[12] 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, acheiving 67 FPS, albeit on a powerful NVIDIA TITAN Xp GPU. For the Raspberry Pi 3B, the paper by Sismananda et al.[32] has shown to run YOLOv3 at a very low 1 FPS, with an IoU (Intersection over Union) accuracy of 86.7%, which is very low.

However, efforts made by Neural Magic[17] in the optimisation of YOLOv5 and YOLOv8 show performance improvements of up to 10x on CPUs, through their open-source optimisation toolkit SparseML[18] and CPU inferencing runtime, DeepSparse[16]; a very promising result.

2.4 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 that are well labelled with bounding boxes and class labels. This is a time consuming process, and it is important to have a large dataset to ensure that the model is robust and generalises well.

To solve this problem, a paper by Yang et al.[36] proposes many different training techniques, the most promising of which is semi-supervised learning. In this approach, the model is trained on a small labelled 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 labelled 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 labelling the entire dataset.

2.5 Key Takeaways

After reviewing the literature, I have determined the following;

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

A down-facing camera is the most viable approach for the future of the project, as it allows for a conveyer belt system solving the problem of component transportation, and so the design of the Computer Vision system will be based on the research outlined above.

The most viable approach to the problem of component identification is to use a CNN, specifically the YOLOv8 architecture, as they have been shown to be very effective at identifying electronic components, and with the ability to to optimise the model using SparseML and DeepSparse adds much promise, as it may allow me to optimise the model for the Raspberry Pi 4, which will be used in the project. Additionally, the YOLO architecture is one that I will likely come across in my modules Deep Learning COMP70010 and Machine Learning COMP70014, which will serve to deepen my understanding of the architecture and allow me to make more informed design decisions. As such, for the purposes of this project, I will be using the YOLOv8 architecture, making use of the SparseML and DeepSparse optimisation tools and employing semi-supervised learning to train the model.

3 Implementation

I have identified three systems that compromise the project:

- The Hardware, Mechanics and Electronics
- The software that coordinates the various components
- The Computer Vision system

3.1 Hardware, Mechanics and Electronics

The hardware, mechanics and electronics (HME) of the system are the physical components that make up the system. It is necassary to first consider the HME of the system as they are the foundation upon which the rest of the system is built; for example, when designing the Computer Vision system, it should utilise training data that was collected using the HME that the system will be deployed on, ensuring that the system is robust to the conditions it will be used in. As such, the main focus of this stage has been the HME.

3.1.1 Hardware

For the initial stage of the project, the main hardware components of the HME are:

- Raspberry Pi 4 Model B 2GB
- 7" Touchscreen Display
- 24V dc, 6.25A, 150W Power Supply
- Okdo Adjustable Focus OV5647 Camera
- LED Light Strip

Raspberry Pi 4 Model B 2GB
The Raspberry Pi 4 Model B is 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 and is widely used in the industry. It has a large amount of software and driver support, so I can be confident that I will be able to find a solution to any issues that may arise. Additionally, it has GPIO pins that can be used to control other components which will be necassary in future stages. It also has a dedicated camera port which allows for a camera to be connected directly to the SoC, which is necassary for the Computer Vision system.

It also has WiFi support for SSH and remote development, as well as HDMI port for the display. Originally, the 4GB model was ordered however an issue with the EE Stores resulted in receiving the 2GB model instead. This was not an issue as the 2GB model is sufficient for the initial stage of the project, however if it is found to be a bottleneck in future stages, a more powerful model may be used. The decision to not return the 2GB model is because the EEStores are not available over the Christmas break.

7" Touchscreen Display
The DFRobot 7" Touchscreen Display was chosen as it is a relatively cheap display that is compatible with the Raspberry Pi. It has touchscreen support and has a Raspberry Pi 4 mount on its back, as well as HDMI adapter boards to connect to the Pi. This means I do not have to design a physical mount for the Pi, and only need to design a mount for the display.

24V dc, 6.25A, 150W Power Supply
There are two parameters that were taken into consideration when choosing a power supply: the power output and the voltage.

- **Power Rating**
  The power supply must be able to drive all

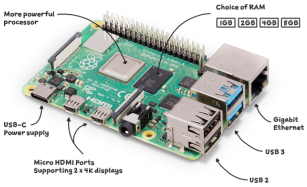

Figure 6: Raspberry Pi 4 Model B 2GB[7]

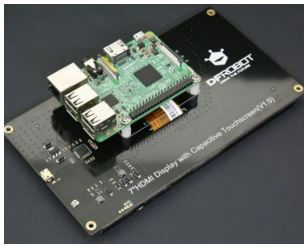

Figure 7: DFRobot 7" Touchscreen Display[5]

Figure 8: Power consumption of the system

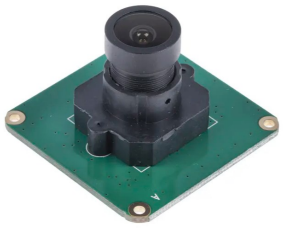

Figure 9: Okdo Adjustable Focus OV5647 Camera[22]

components in the system with overhead in case of spikes in power consumption. In this first stage, the system only consists of the Raspberry Pi, the display, LED lights and camera, making for a power consumption of under 20W, recorded using a smart plug with a power meter. However, in the future, the system will likely need to drive additional motors and sensors, so the total power consumption will be higher. Even with accounting for this, the power consumption will be under 100W, so a 150W power supply will be sufficient, allowing a 50W overhead. In hindsight, this may be considered excessive.

- **Voltage Rating**
  The voltage of the power supply must be greater or equal than the highest voltage of any component in the system; this is necessary as I have only step-down converters available on hand, so choosing a higher voltage allows me to step-down the voltage to the required voltage for each component. 24V was chosen as it is a common voltage for stepper motors, which may be used in future stages of the project. It is also used for some LED strips, which are used in this stage. The Pi 4 uses 5V and the LED strip uses 12V, which the 24V power supply can be stepped down to.

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 camera to be specifically tuned to the design of the system, allowing for the best possible image quality. It also has a 5MP sensor[23], which is sufficient for the Computer Vision system, as high resolution images would be preprocessed and reduced.

LED Light Strip
An LED strip is required to illuminate the components so that the camera can capture images of them. Initially, a 12V LED Ring was chosen as it would allow for a uniform light source and can be mounted directly below the camera; however for reasons explained in the next section, it was replaced with a 5V WS2812B LED strip as it solves the issues faced with the LED Ring.

3.1.2 Mechanics

To house the components, a 3D printed housing was designed. It features a modular design, allowing for an iterative design process where I am able to work on different components of the system independently. It also allows for easy replacement of components should my design prove to be inadequate. The housing was designed in FreeCAD[8], a free and open source parametric CAD software that I have personal experience with.
The parametric design allows for easy modification of the design should I need to make changes - with good design practices, I can ensure that changes can be done as I please by simply modifying a few numerical parameters. The design was then printed using my own personal 3D printer, a heavily modified Voxelab Aquila C2. For this stage on the project, I must design 3 major components for the housing:
PSU Housing
The PSU housing contains the power supply and ensures that all high voltage components are safely enclosed. It also houses the power switch, the power socket and the terminals. As I will require step down converters for the Pi and LED strip, the PSU housing also contains mounting points for the step down converters, as well as

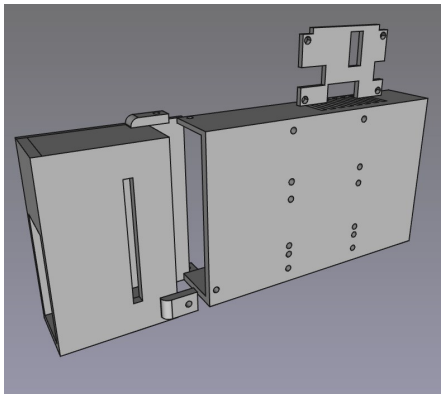

Figure 10: PSU Housing

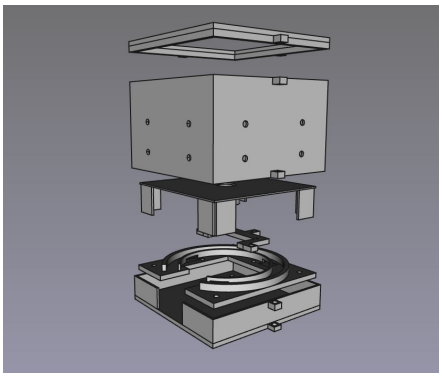

Figure 11: Camera Housing

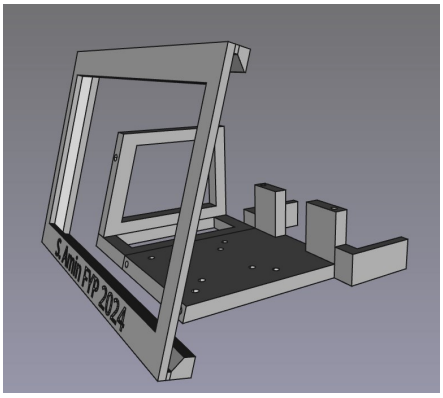

Figure 12: LCD Display Housing

a mounting plate to ensure that the step down converters are mounted flat to the PSU housing.

Camera Housing

The camera housing contains the camera and the LED strip. It also contains the mounting points for the camera and LED strip. The camera housing will also mount an acrylic plate above the camera, so components can be placed on the plate and be imaged by the camera.
From bottom to top in Figure 11, is the bottom casing, LED Ring mount, camera mount, light diffuser, middle casing and top casing.

LCD Display Housing

The LCD display housing contains the LCD display and the Raspberry Pi. As the 7″ DFRobot LCD display has a Raspberry Pi 4 mount on its back, an explicit mount for the Pi is not required. The mount will position the LCD display at an angle, so it can be viewed from above. The LCD display simply slides into the mount, allowing for easy removal and also contains mounting points to secure the LCD display to the camera housing.
The LCD housing is split into two parts; the LCD cover, where the LCD display slides into, and a base plate, which mounts to the camera housing. The LCD cover is then mounted onto the base plate, securing the LCD display in place.

The components are secured using brass M3 heat-set inserts and M3 screws. The heat-set inserts are inserted into the 3D printed parts using a soldering iron, and the components are then screwed into the inserts. This allows for easy removal of components should I need to make changes to the design, following my modular design principle.

3.1.3 Electronics and Safety

As my project is a physical system, there are several electronic components that are required to make the system work.

Power Supply

As mentioned in Sections 3.1.1 and 3.1.2, the system is powered by a 24V, 6.25A, 150W power supply, and must be stepped down to 5V for the Pi and 12V for the LED strip. This is achieved using a XL4015 Step Down Converter[15], a variable step down converter that can output up to 75W. The high power rating of the step down converter allows for a large overhead, which is useful for future stages of the project where additional components may be added to the system.

For powering the system, I have chosen a standard IEC C14 power connector and a RS Pro Snap-In Fused Rocker Switch[4]. As the power supply is a 6.25A power supply, I have chosen a 6A fuse for the switch, which is the closest fuse rating to the power supply's current rating. This ensures that the fuse will blow before the power supply is damaged, should there be a short circuit in the system.

For wiring the power supply and connecting the step down converters, I have chosen 18AWG wire, which is rated for 17A[3], more than enough for the system's power consumption. The wire is securely crimped to the power supply using a crimping tool.

Raspberry Pi 4

To power the Raspberry Pi, I have chosen a standard USB-A to USB-C cable and connected it to a female USB-A breakout board connected to the 5V step down converter. This allows for easy removal of the Pi should I need to make changes to the system, and prevents the unnecessary cutting of cables.

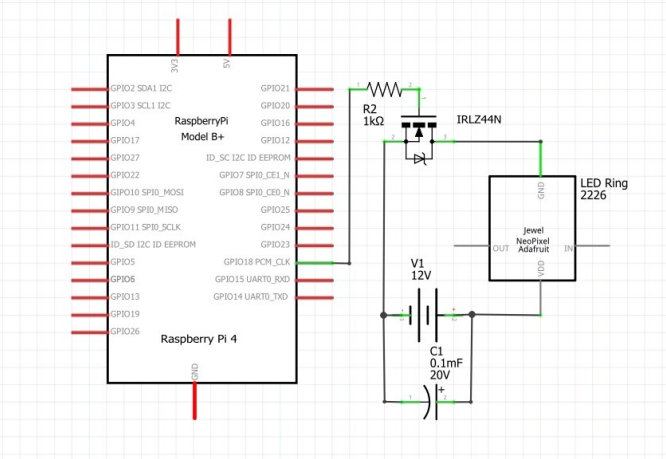

Figure 13: Wiring Schematic for MOSFET, made with Fritzing[9]

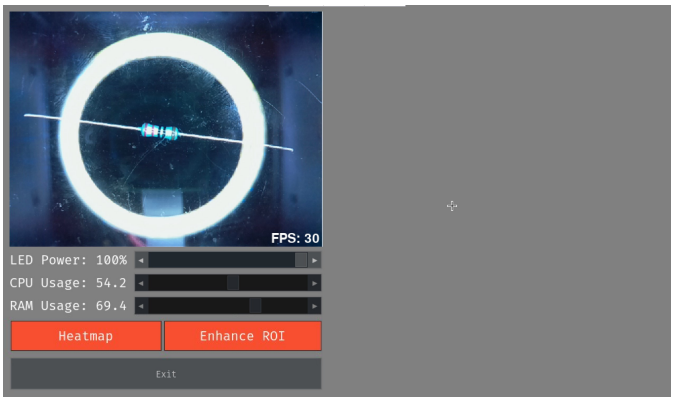

Figure 14: Main UI, captured using RealVNC[28]

As the Pi is mounted to the LCD display, a USB-A to Micro USB cable is used to connect the Pi to the display, which is then connected to the Pi's HDMI port, using a supplied HDMI adapter board.

LED Lighting

As mentioned in Section 3.1.1, initially, a 12V LED Light Ring was used and was changed to be a WS2812B 5V LED strip, for issues described later. As shown in Figure 13, the LED Ring was connected to a 12V supply (achieved using the 12V step down converter) and parallel to a 100μF capacitor, which is used to smooth out the voltage spikes.

The LED Ring was then connected to an N-channel MOSFET IRLZ44N[13], which is controlled by a PWM-enabled GPIO pin on the Pi. This GPIO pin is connected to a 1kΩ resistor, which prevents potential overcurrent from damaging the Pi. The MOSFET is then connected to ground, completing the circuit.

The IRFZ44N is a logic level MOSFET, meaning it can be fully turned on with a 3.3V gate voltage, which is the voltage of the Pi's GPIO pins, and the high switching speed (up to 1mHz) of the MOSFET allows for the LED Ring to be PWM controlled, allowing for the brightness of the LED Ring to be controlled.

3.1.4 Design Problems and Solutions

3D printing is typically an iterative process, where the design is printed, tested, and then modified based on the results. This further reinforces the need for a modular design, as it allows for easy modification of each component. After a few iterations with errors due to tolerances and slightly incorrect measurements, (not relevant to the technical content of this report) I was able to produce a design that physically fit together and mounted the components as intended, however when combined with the software and electronics, I encountered a few problems that I had to overcome by making some design decisions.

Mechanical Design Problems

The first design problem I encountered was the initial design of the LCD display mount. As mentioned in Section 3.1.2, the LCD housing is split into two parts; the LCD cover, where the LCD display slides into, and a base plate, which mounts to the camera housing. The initial design of the LCD cover only mounted to the base plate using two M3 screws at the bottom Figure 17, which made it unstable and prone to wobbling. Overtime, this would put stress on the plastic, causing it to break. To solve this, I added a third mounting point to the LCD cover in the form of a bracket that reaches halfway up the LCD cover, as shown in Figure 18, increasing the stability of the LCD display preventing it from wobbling.

The second design problem I encountered was the design of the camera housing. As mentioned in Section 3.1.2, the camera housing contains the camera and the LED strip. The camera itself is not a macro lens, but rather a standard lens with an adjustable focus. The workaround is to set the focus as close as possible, and then determine the optimal distance between the camera and the object to be imaged. This is done by placing an object on the acrylic plate. Initially, the height of the camera housing was too short Figure 19, resulting in a blurry image, which was remedied by simply changing the parametric parameters

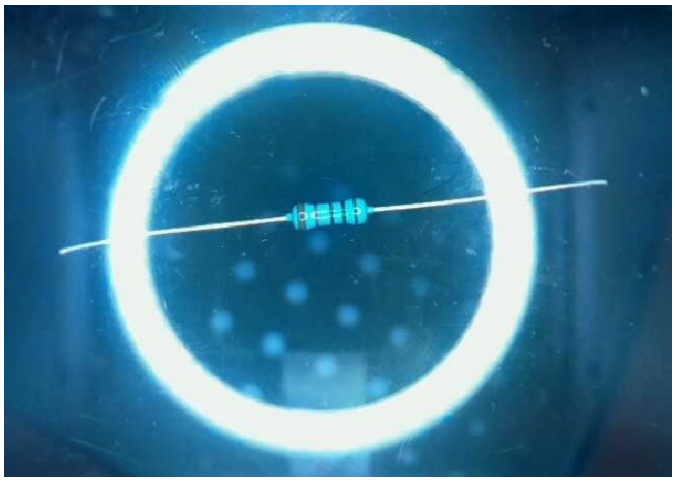

Figure 15: Glare from LED Ring

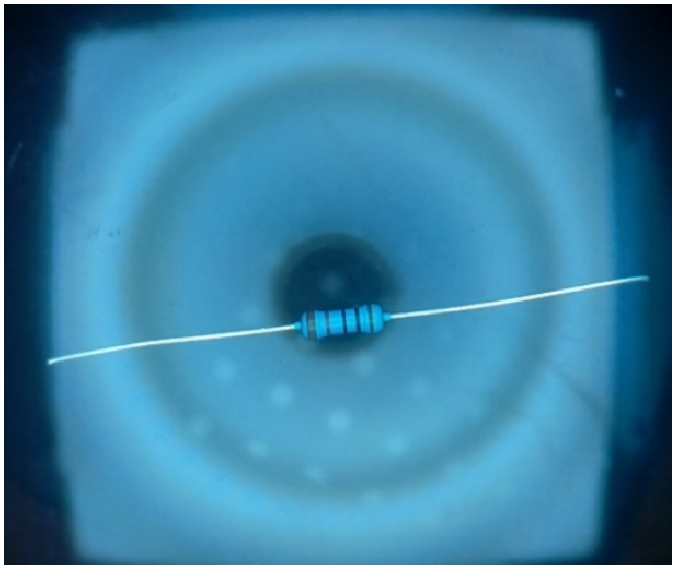

Figure 16: Diffused Light

that define the height of the camera housing Figure 20. The above design problems were solved by simply modifying the design, however the third design problem was more complex. As mentioned in Section 3.1.2, the approach to imaging components is to place them on the acrylic plate, and then image them using the camera. However, when doing so, the light from the LED Ring would reflect off the acrylic plate Figure 15, resulting in a incredible glare that would obscure the image of the component. There was an attempt to 3D print a light diffuser to reduce the glare Figure 16, however this was not effective, and instead causes the entire image to be affected by the reflection of the LED Ring. This was a major design problem as, in the non-diffused image, any components in the glare of the ring are obscured, and in the diffused image, the entire image has incorrect colours and is partially obscured. To solve this, three solutions were considered:

- **Polarising Filter**
  Polarising filters can be used to reduce glare by filtering out light that is polarised in a certain direction.
- **Different Light Source**
  The LED Ring was chosen as it provides a uniform light source as it would surround the camera, however it is possible to use a different light source that does not cause glare, for example an LED strip that is mounted above the camera.
- **Down-facing Camera**
  Instead of imaging the components from above, the camera can be mounted below the acrylic plate, and the components can be imaged from below. The issue is that the light of the LED Ring bounces off the bottom layer of the acrylic plate, so any components on top are obscured.

However, in the end none of these could be considered because of the electronics design problem discussed in the following section.

Electronics Design Problems

Attempting to dim the LED Ring resulted in very spoaradic flickering, and it did not react with changes to the frequency of the PWM signal despite the MOSFET control circuit shown in Figure 13. Originally, I considered that the issue was with the MOSFET as I had initially used a MOSFET with a higher gate threshold voltage, and so I replaced it with a MOSFET with a lower gate threshold voltage. However, this did not solve the issue, and I also attempted to add a capacitor in parallel to the LED Ring to buffer the current, but to no avail. The LED Ring seemed to be a simple ring of LEDs, with no additional circuitry, so it was not clear it was not dimmable. My supervisor Dr. Stott was also unsure as to why it was not dimmable, and proposed a different circuit with an inductor and diode but this requires a much higher switching frequency than the Pi's PWM signal can provide, and so it was not viable.

For these reasons, and after reviewing literature as dicussed in Section 2, I decided to redesign the camera mount to be down-facing (as I may require a conveyer belt in the future), and replace the LED Ring with a WS2812B LED strip, as these are known to be dimmable (as I have personal

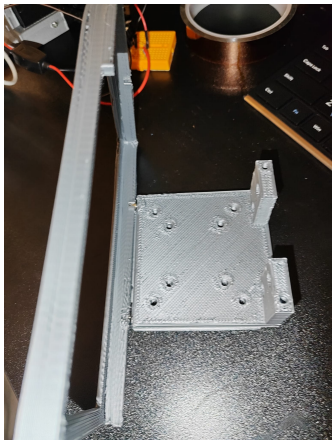

Figure 17: Unbraced Screen

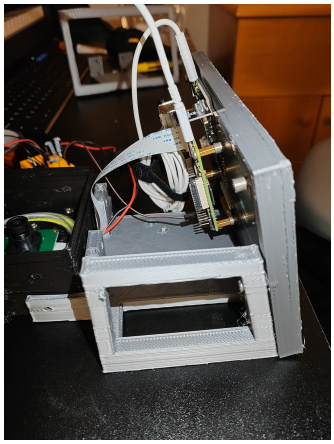

Figure 18: Braced Screen

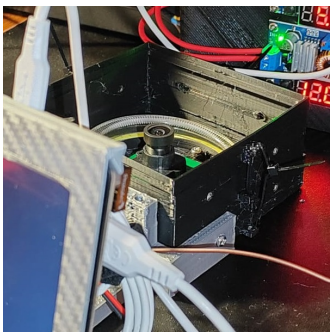

Figure 19: Short Camera Housing

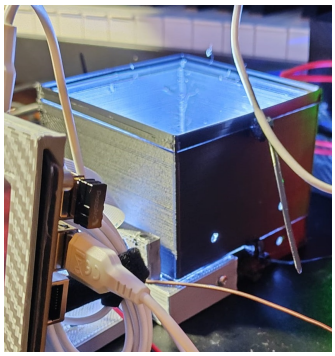

Figure 20: Optimal Camera Housing

experience with them). They are also addressable, however this is not a feature that I will be using in this stage of the project.

3.2 Software

The entire system is controlled by a Raspberry Pi 4B running a very light weight Linux distribution called DietPi. I aim for the entire system to run locally on the Pi, and so the Pi will be responsible for running the Computer Vision system, controlling the hardware, and running the user interface. In order to achieve this, I have identified the following software components:

- The User Interface
- The Computer Vision system
- The Hardware Control system

The system's software is written in Python[26], as it is a language I am very competent with and is well suited for the project, given the availability of libraries for Computer Vision and hardware control. Again, the concept of modularity underpins the design of the software, and so each component is designed to be independent of the others, and can be developed in parallel.

All written Python code adheres to a style defined in a .pylintrc file, which is a configuration file for the pylint[25] linter. This ensures that the code is consistent and readable, and also helps to identify potential bugs and errors. The code is also documented using docstrings for readability and maintainability. Each software component is designed to be self-contained so they can be run independently for testing purposes.

All software constants are defined in a constants.py file, which is imported by all other Python files, allowing for easy modification of constants that modify the behaviour of the system which is useful for testing and debugging.

Additionally, a lot of thought went into streamlining the development process to ensure that the system is easy to develop and maintain. For example, I use Visual Studio Code's[2] Remote SSH extension to develop the system remotely, as it allows me to develop the system on my laptop, while using the familiar VSCode environment with any extensions that I am accustomed to.

The Pi uses Git[10] for version control, and the repository is hosted on GitHub[11], with its own dedicated branch for the Pi that is regularly updated with the main branch. The vision system and my laptop also maintain their own Git branches which are regularly updated with the main branch, ensuring that all systems are running the same code. This is crucial as it allows me to develop the system on my laptop, and then push the changes to the Pi, without having to manually copy the files over.

During development, the Pi connects to my laptop's WiFi hotspot, and is configured to be discoverable with the Pi's hostname, allowing me to connect using SSH and a VNC Viewer without needing the Pi's IP address, which is dynamic and may change. The Pi also makes use of SSH keys, allowing me to connect to the Pi without needing to enter a password, ensuring quick and easy access.

User Interface

As the system is intended to be self-contained, it is important to have a user interface that displays all relevant information to the user and allows them to interact with the system. The UI is displayed on the 7" touchscreen display, and is written using Pygame[24];

Figure 21: Dataset Labeller Tool

This design decision was made after experimentation with other UI libraries like Tkinter[34], however it was found that Pygame was the most viable for several reasons; it contains a module for camera streaming, which is used to display the camera feed on the UI, and can also be used to feed the camera frames to the Computer Vision system; there is a very large amount of control provided by Pygame over the UI, especially in terms of performance. This is crucial as the UI is only there to provide information to the user, and so it is important that it does not interfere with the Computer Vision system or the hardware control system. With an additional library named pygame_gui[20], it is possible to create a UI with GUI elements like buttons, text boxes, and drop down menus, making for a fully comprehensive UI library.

In Figure 14, the current version of the UI is shown. On the left, the camera feed and camera frame rate is displayed, with system CPU and RAM usage displayed, for monitoring purposes. There is a slider to adjust the power of the connected LED Ring light, as well as two buttons related to the Computer Vision system.

The right side is blank for now, but will be used to display information about the Computer Vision system and the hardware control system.

3.3 Computer Vision

As the Computer Vision system should be trained on data that is representative of the conditions it will be used in, and because of the previously mentioned issues with the mehcanical design of the system, the Computer Vision system was not developed in this stage. The data collection process is manual and time consuming, and so it is far more efficient to ensure that the mechanical design is finalised, and the foundation of the project is robust and reliable. As will be discussed in the Section 4 (Project Plan), the mechanical design will be overhauled in the next stage, so if I had collected data for the Computer Vision system in this stage, it may have been rendered useless in the next stage.

However, this does not mean that no work was done on the Computer Vision system. As per Section 2 Background, I have determined that YOLOv8 is the most suitable model for the Computer Vision system. As such, I have developed a number of tools to aid in the development of the Computer Vision system.

Additionally, careful consideration of the Computer Vision system pipeline was done in this stage.

3.3.1 Data Collection

For the task of component identification, the Computer Vision system must be trained on data that is representative of the conditions it will be used in. As such, the data collection process is manual and time consuming. For this reason, I have developed a customtkinter[30] script to aid in the data collection process. The script allows me to label images with bounding boxes and class labels, and save the labels in a format that is compatible with the YOLOv8 model. The script also allows me to quickly label images using shortcut keys, and sorts each image into a folder based on the class label, making it easy to manage the data. Given that this is my own script, I can easily add or remove features as needed - it is likely that I

will need to add more features such as component orientation, so having my own script is very useful.

This script works in conjuction with the UI, seen in Figure 14, the Raspberry Pi (made possible using RealVNC[28]) and allows me to label images with bounding boxes and class labels.

The script has support for multiple components, including:

- Resistors
- Capacitors
- Ceramic Capacitors
- Inductors
- Diodes
- MOSFETs
- Transistors
- LEDs
- Wires
- ICs

A screenshot of the script can be seen in Figure 21.

A Jupyter Notebook[14] was also developed to aid in developing the Computer Vision system. This is common practice as it allows code to be run and the data associated with it to be inspected, which is very helpful for debugging purposes. It has the capability of training every model that will be used in the Computer Vision system, and contains features like data augmentation, checkpointing, and model evaluation.

The data augmentation feature is very useful as it allows me to retain the original data should I need to add more labels or modify the data in any way. The checkpointing feature allows me to save the model at any point during training, which serves as a backup but also allows me to resume training and helps to protect against overfitting. The model evaluation feature allows me to evaluate the model on a test set, and provides useful metrics like precision, recall, and mAP. This is useful for evaluating the model's performance, and also for comparing the performance of different models.

3.3.2 Vision System Pipeline

The Computer Vision system pipeline is the sequence of steps that the Computer Vision system takes to identify components.

The system pipeline is as follows:

- **Preprocessing**
  The image is preprocessed to improve the quality of the image, and to make it easier for the model to identify components, ensuring uniformity and consistency in the data.
- **Component Detection**
  The image is fed to the model, which outputs a class, confidence score, bounding box, and orientation for each component.
- **Component Value Identification**
  The value of the component is identified from the subimage within the bounding box, and is used to further classify the component.

While a single model can be used to identify components, it is necassary to use two models to identify the value of the component. This is because the value of the component may differ depending on the type of component. For example, the value of a resistor is read from the colour bands, whereas the value of a capacitor is read from the text printed on the component. As such, each component may require a different model to identify its value.

This is not an issue as it conforms to the modular design of the system, and allows for easy extensibility. For example, if a new component is added to the system, it is easy to add a new/reuse an existing model to identify the value of the component; however the Component Detection model will need to be retrained as the dimensionality of the output layer will change due to the addition of a new class. The Jupyter Notebook developed for the Computer Vision system makes this process very easy, as it would simply require me to add the new component to the list of components, and the Notebook will handle the rest.

4 Project Plan

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.

5 Evaluation Plan

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.

6 Safety Plan

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.

7 References

[1] 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.

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

[3] RS Components. Rs pro 18 awg hook up wire, pvc insulation. https://docs.rs-online.com/770d/A700000007035534.pdf, n.d. Accessed: 22-01-2024.

[4] RS Components. Rs pro c14 snap-in iec connector. https://docs.rs-online.com/c6ad/0900766b815867b2.pdf, n.d. Accessed: 22-01-2024.

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

[6] 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.

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

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

[9] Fritzing. Fritzing. https://fritzing.org/, n.d. Accessed: 22-01-2024.

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

[11] GitHub. Github. https://github.com/, n.d. Accessed: 22-01-2024.

[12] 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.

[13] Infineon. Irlz44n datasheet. https://www.infineon.com/dgdl/Infineon-IRLZ44N-DataSheet-v01_01-EN.pdf?fileId=5546d462533600a40153567217c32725, n.d. Accessed: 22-01-2024.

[14] Jupyter. Jupyter. https://jupyter.org/, n.d. Accessed: 25-01-2024.

[15] Katranji. Xl4015 datasheet. https://www.katranji.com/tocimages/files/457120-274359.pdf, n.d. Accessed: 22-01-2024.

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

[17] 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.

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] 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.

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

[29] 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.

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

[31] 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.

[32] 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.

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

[34] Tkinter. Tkinter documentation. https://docs.python.org/3/library/tkinter.html, n.d. Accessed: 22-01-2024.

[35] 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.

[36] 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.

[37] Z. Zhang and Massachusetts Institute of Technology. Department of Mechanical Engineering. *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.