FarmFresh Design Review Report

Ishita Kumar, Ishita Sinha, Kushagra Sharma
Electrical and Computer Engineering, Carnegie Mellon University
{ikumar, isinha, kushagrs} @andrew.cmu.edu

Abstract- **It is important to separate rotten fruits early in the food distribution process to prevent healthy fruits from getting spoiled and before more monetary value is added through packaging and transportation. Thus, FarmFresh, a fruit quality evaluator and sorter, can help save energy and resources as well as help prevent food wastage. Our solution uses computer vision to autonomously sort and separate healthy and rotten fruits, thus preventing human error. It can save hours of manual labour for farmers so they can utilise their time to work on other value-adding tasks that can't be automated.**

Index Terms- Computer Vision, Fruit classification, Machine Learning, NVIDIA Jetson Nano, OpenCV

I. Introduction

The saying, one rotten apple spoils the barrel is quite literally true. If not separated, rotten fruits can spoil fresh fruits and wreak havoc in the food distribution industry. It is also important to separate rotten fruits as early as possible before more monetary value is added through packaging and transportation. Thus, to save energy and resources and prevent food wastage, it is necessary for farmers to accurately separate rotten fruits before further distribution. This task must be automated to prevent human error and save farmers' time. Although fruit sorting machines exist, they are very expensive. As 91% of US farmers are small farmers, there is a vital need for a more affordable yet autonomous solution[1]. This is why we introduce FarmFresh: a low-cost fruit quality evaluator and sorter. Our solution uses computer vision to quickly and autonomously sort and separate healthy and rotten fruits. This prevents human error and can save hours of manual labor for farmers. This saved time can instead be spent on other more value-adding tasks that cannot be automated.

FarmFresh aims to be highly accurate in sorting rotten and fresh fruits. Our project will work with bananas, apples, and oranges. We plan to achieve rotten fruit detection with false negatives less than 5% and false positives less than 15%. Our solution is meant to be integrated into existing conveyor belt systems, so FarmFresh will be built to accommodate the typical conveyor belt speed of 5 cm/s and specific fruit spacings by taking pictures of fruit every 4 s. Our project will take pictures in uniform bright lighting to mimic factory settings and achieve consistency. FarmFresh will be powered to work for 8 hours at a time so it can last a typical work day.

II. Design Requirements

To ensure that FarmFresh is useful to farmers, we considered numerous requirements from the farmer's perspective. These requirements also ensure that our product is competitive in the market space, and offers something new that the alternatives lack. A list is presented below:

- Support multiple fruits. We want to be able to support bananas, apples, and oranges.
- Needs to be compatible with a majority of conveyor belt architectures.
- Needs to be easy to install and portable. To validate this requirement, we will be building our own conveyor belt system, and making sure FarmFresh is fully compatible with it.
- FarmFresh needs to quickly sort fruits, and needs to be able to work on existing conveyor belt speeds of 5 cm/s. No fruit should be missed.
- Needs to have a false negative rate of less than 5%. This is to ensure that we rarely categorize rotten fruits as fresh since that can potentially be catastrophic. To get a better understanding of accuracy metrics, we found several papers online. In "A Survey on Computer Vision Technology for Food Quality Evaluation", the author was able to achieve 97% accuracy classifying cracked eggs using edge detection[2]. In a more recent paper, the authors were able to achieve 89% accuracy correctly predicting rotten vegetables such as carrots and bell peppers[3]. This is why we chose a high classification accuracy.
- Needs to have a false positive rate of less than 15%. We define a false positive as follows: a fruit is predicted as being rotten, when it is not actually rotten. This is a less severe case. We do lose value from the misclassified fruit, but at least it doesn't impact the rest of the fruits. This is why we want our product to have stringent requirements when it comes to detecting rotten fruits, at the cost of potentially increased false positives.
- Needs to be able to support a typical 9-5 day, so the battery should last at least 8 hours.
- Needs to cost less than $600 so that it is cost effective.

[1] https://www.nass.usda.gov/Publications/AgCensus/2007/Online_Highlights/Fact_Sheets/Farm_Numbers/small_farm.pdf

[2] https://www.researchgate.net/publication/312494475_A_Survey_on_Computer_Vision_Technology_for_Food_Quality_Evaluation

[3] https://www-sciencedirect-com.proxy.library.cmu.edu/science/article/pii/S0924224403002711

To simplify some of the design decisions later on, we assume that fruits are evenly separated from each other at intervals of around 20 cm. This assumption is based on the fact that the fruits are collected/pre-sorted at even intervals, which means that most farm systems would not need to change their existing infrastructure much to accommodate this. This lets us establish how often we need to take pictures. With the conveyor belt speed operating at 5 cm/s, and fruit spacing at 20 cm, if we take a picture every 4 s, we can avoid unnecessary computation (thus saving battery life), while not missing any fruits.

The main considerations when designing the mechanical aspects (conveyor belt) were the following:

- The width of the conveyor belt needs to be large enough to support placing fruits.
- The length of the conveyor belt needs to be large enough to support placing multiple fruits (at least 3) so that we can perform stress tests and simulate the process better.

These two considerations allow us to set the width of the conveyor belt at 4 inches (the average width of a banana is 2 inches), and the length at 26 inches, so that we can place 3 fruits at any given time at intervals of 20 cm (7.8 inches).

III. Architecture and/or Principle of Operation

The architecture of FarmFresh consists of two major components - the conveyor belt design, which is the mechanical component, and the rest of the system, which constitutes the hardware and software parts of the product. This can be seen in our block diagram in Figure 6 in the Appendix on page 8.

As part of this split, we must note that the conveyor belt setup itself does not depend on any part of the product, and vice versa. Our product is designed to be integrated into existing conveyor belt systems that meet our speed specifications. As a result, the product is independent of the conveyor belt, so the two components can be built separately, and they operate separately as well. The functioning of our product would not impact the conveyor belt, and vice versa.

Figure 1 provides an overview of how our product is designed to work with the conveyor belt system. As a fruit comes along, the camera modules capture an image of the fruit. Once this image has been captured, it is sent through the camera multiplexer to the NVIDIA Jetson Nano which saves it on disk. The board runs our classification algorithm on the images of the fruit and determines whether it should be classified as rotten or good. It passes a signal to the gate controller, and accordingly the gate controller rotates the gate so that the fruit is pushed into the appropriate basket.

The entire system has the base computer as the NVIDIA Jetson Nano. To the Jetson Nano, we have connected two Raspberry Pi camera modules (connected via the arducam module), which correspond to the cameras shown in Figure 1. As fruits come along the conveyor belt, the cameras will be programmed to automatically capture a picture every 4 s. The cameras capture the picture of the fruit and save it to disk.

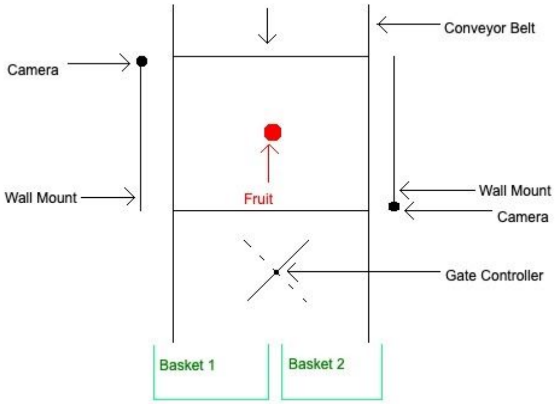


Fig. 1. System picture

Once the image of the fruit has been captured by both cameras, it is sent to the NVIDIA Jetson Nano. On the NVIDIA Jetson Nano board, we would have our classification algorithm. This algorithm begins by converting the image to hsv space. Post that, it performs image segmentation to isolate the fruit from the background. We plan on having a white background for the fruits on the conveyor belt so that the images can be captured with ease. We also plan on building a shed using the wall mounts so that it could hold the cameras and the NVIDIA Jetson Nano in a fixed position. Further, we plan on having controlled lighting in this shed so that all of the images can be classified more easily. Once the fruit has been segmented from its background, and appropriate noise filtering is applied, this image would be passed through an edge detection algorithm. Using this algorithm, we would be able to determine the boundary of the fruit in the image. Once this has been achieved, we can pass the image into our classification algorithm in order to determine whether the fruit is rotten or not. We plan on building 2 classifiers in order to see which one works best. The first one would involve performing a pixel by pixel analysis of the fruit in order to obtain the percentage of pixels corresponding to rotten parts of the fruit. We could obtain a threshold that would indicate above what percentage of the area would be classified as rotten, and using this, the classifier could classify the fruit based on whether it is rotten or good. An alternative classifier would involve using neural networks (Alexnet) in order to predict the rottenness of the fruit based on a classifier that we'll develop and train on several images. To perform all of this computation, we're using the OpenCV and Scikit-image libraries provided by Python.

Once the classification has been performed, the result is passed along to the gate controller. The gate controller facilitates communication between the Adafruit servo controller and the servo motor itself. Once the Adafruit shield receives the signal, it sends along the signal to the servo motor controller so that the gate can be rotated the appropriate amount in the appropriate direction in a specified amount of time. Based on the result of the classification, the gate will rotate so as to put the fruit in one of two baskets - the good one, or the rotten one, depending on the result of classification.

This entire product is designed to be integrated into an existing conveyor belt system. Our conveyor belt setup consists of electrical as well as mechanical components. The electrical components include the RC motor speed controller board and a 12 V DC Gear Motor for moving the conveyor belt at the appropriate speed. The mechanical components that are involved in building the conveyor belt are discussed in Section V.

This entire system is powered by a 5 V power supply. We need this entire system to last a regular 8-hour working day for the farmer, so based on the use case requirements, we have decided to use a 25000 mAh power bank as our power source since we believe it should last for the requisite amount of time.

IV. Design Trade Studies

Hardware

A. Fruit Sorting Mechanism

For the final design of our fruit sorting system, we decided to use a servo motor to control a gate which can turn and direct the fruit into the appropriate basket. This is different from our initial design which involved pistons pushing fruit off a platform and into the right basket. We decided against the latter design because when we researched pistons, they were too slow to meet our required speeds. We looked into solenoid pistons as well, which were faster than the other types of pistons, but these were expensive. In addition, pistons were hard to configure since our team lacked the mechanical knowledge required. Furthermore, none of the pistons we researched had the necessary reach to push the fruit far enough to make it fall over the side of the conveyor belt. On the other hand, we all have experience working with servo motors and the gate system seemed to be a more simple design that met our requirements better. Thus, we chose the servo controlled gate system as our fruit sorting mechanism due to it being faster and more easily configurable.

B. Fruit Platform

It was important for us to decide a good system for placing our fruit as our image segmentation algorithm for fruits would be strongly affected by this choice. Our initial design involved a white rotating plate on which the fruit would be placed. We would then take pictures periodically as the fruit rotated. This would be good for image segmentation and we would only require one camera for this design. However, this design was not easily integratable with a conveyor belt system, which is our intended end goal. We also needed to test if our design would work with a conveyor belt, which was not accurately possible without moving the fruit in some way. Thus, we decided to build a conveyor belt and place fruits on it, as our end-user would, rather than using a rotating platform. This would provide a more realistic environment for our project. We plan to cover the base on which the fruit is placed with a white material to still get the benefits for image segmentation. We still wanted to be able to access all sides of the fruit thought, which naturally led us to incorporate multiple cameras in the new design.

C. Hardware Platform Placement

Our design requires a platform to place the camera(s) for taking fruit pictures and the Jetson Nano for computation. We decided to use wall mounts on either side of our conveyor belt system to place our cameras. Our design involves placing the cameras diagonally across from each other so as to get different angles of the fruit, including front and back. We decided a 2 camera system placed on mounts worked better than a single camera as it is integratable into a conveyor belt system. The single camera system would have had the fruit rotating and so we would have had to stop the movement of the fruits which is not ideal. The Jetson Nano will be placed on the top of a shed, directly above the conveyor belt. This shed will also have a uniform and bright lighting attached to it for consistency in our pictures. We decided the shed system would be best for placing the Nano as we would also need it for the lighting.

D. Computer

We decided to use an NVIDIA Jetson Nano for our computer vision and machine learning algorithms. This is because the Jetson Nano has a quad core A57 @ 1.43 GHz processor and a dedicated 128-core Maxwell GPU. This works well for our design as it has the computation power required for our algorithm to run quickly as well as take live pictures quickly. We would also prefer a dedicated GPU for processing fruit images and computer vision. We also looked into using the Raspberry Pi but we did not think it would provide good enough computational power for our project needs as it does not have a dedicated GPU. We chose Jetson Nano A02 over the newer B01 model (which has 2 CSI camera ports) because we wanted our design to be scalable to 4 cameras if we needed to introduce more angles of the fruit to meet our accuracy metrics. This meant using the arducam camera module regardless of whichever model we went for. Thus, we chose the cheaper version.

E. *Camera*

The camera is an important part of our design since we use it to take live pictures of fruit as it moves on the conveyor belt. These images are then analyzed to detect whether a fruit is rotten or not. The camera would also be connected to our computer. Thus, we need a camera that takes good quality pictures and can be connected to the Jetson Nano. We decided on using two Raspberry Pi cameras (configured via the camera multiplexer) for the needs of our project since the Jetson Nano actually has dedicated slots for these cameras and they are connected directly to the GPU. The Raspberry Pi Camera Module v2 contains a 8-megapixel sensor, which is high enough resolution for our need to capture detailed fruit images along with any rottenness present. In addition this camera is easy to program as there is significant documentation on how to configure it with the Jetson Nano. We also looked into using USB cameras but decided against it because they would connect to the CPU instead and the costs were similar enough that the USB cameras did not save us much money either. On the other hand, there were certain USB cameras with much better camera quality, but those were much more expensive. Since the Raspberry Pi cameras fit our needs already, we decided to use them for our project rather than the USB cameras.

Software

A. *Image Segmentation*

We are using OpenCV's HSV color thresholding for image segmentation. This works a lot better than RGB thresholding as it is difficult to capture the exact RGB colors a fruit has especially since the exact RGB color value can vary slightly from fruit to fruit and also within the fruit. On the other hand, HSV separates the color information (hue) into its own channel which can help us capture a range of colors associated with the fruit independent of saturation or lightness values.

A. *Fruit Quality Evaluation*

The fruit quality evaluation system is the backbone of our project as this is the algorithm that will decide on the classification of fruit as fresh or rotten. After segmenting the fruit from the image, we use edge detection to get the boundary. Next, we do a pixel by pixel analysis within the boundary of the fruit to detect the percentage of rotten parts based on a fixed HSV color threshold for the same. We also plan to have a neural network based algorithm (e.g AlexNet) for rotten fruit classification. We want to compare both methods and pick the best one rather than making a tradeoff decision without results for both methods.

V. System Description

In this section, we present the block diagram discussed earlier in greater detail. We will go over all the subsystems individually. Our overall design is composed of 3 main subsystems: 1) software for classifying fruits 2) hardware to take pictures, and coordinate the control mechanism for sorting 3) mechanical conveyor belt system which integrates the software and hardware parts

A. *Software subsystem*

This subsystem is the green box named classification algorithm and the gate controller module inside the block diagram. The purpose of this subsystem is to take an image as an input and produce a single output that classifies the fruit in the image as either good or rotten, and to communicate that decision to the Adafruit servo controller.

1. *HSV conversion*

The first step in processing the images is to convert the image from RGB space to HSV space. We tried to process the images using the rgb space, but found it difficult to accurately differentiate between pixels based on color alone. The exact color threshold was hard to capture even after trying to adjust the bounds for a while. After researching, we found HSV space to fit our needs much better since it separated the hue into a separate channel independent of saturation or lightness. Hence, we converted the image to HSV space. This resulted in a significant increase in accuracy during segmentation.

2. *Image segmentation*

Once the image was converted to HSV space, we were able to accurately segment the image and isolate the fruits based on hue (e.g yellow for banana). To accomplish this, we kept adjusting the hue, saturation and brightness bounds. Then we realized a better method would be to graph the HSV colors prominent in a fruit. With that data, we were able to create a mask that segmented the fruit from the background. One challenge we anticipate is that lighting levels will fluctuate depending on the time of day but we try accounting for this by hanging a fixed and uniform light source above our fruit. We can also add an extra step to correct for the ambient brightness. This will most likely be in the form of taking a few pictures before starting the classification process and calibrating the brightness.

3. *Noise filtering*

We realized we needed this step when we were trying to classify the fruit images as rotten or good. The black area on the outside of the fruit (left over as a result of image segmentation) was being taken into account as rotten areas within the fruit. This is important to remove as it will lead to fruits being mistakenly classified as rotten when they are good, driving up the false positive rates. To fix this, we plan to add noise filtering as an intermediary step and use an edge detection algorithm so as to only analyze pixels within the fruit. To accomplish noise filtering, we will apply a Gaussian blur before the edge detection. However, we can't be too aggressive with the blur either as that will blur areas within the banana too that shouldn't be removed. So, it's a balancing act (removing extraneous parts outside the fruit while not removing the 'good' parts inside the fruit) that we will need to get right.

4. *Image classification*

Finally, we classify the fruit in the image as being good or rotten. We have not implemented this step yet, but our plan is to analyze localized groups of pixels that are a different hue and brightness than their neighbours based on a fixed HSV threshold that signifies rottenness. We will analyze darker areas inside the fruit since those symbolize decay. Depending on the size and frequency of these darker areas, we will categorize the fruit as good or rotten. Since this is significantly more difficult than the other steps, we have prepared a contingency plan in case we are unable to achieve our designed accuracy metrics: we will use a neural network based solution. Our current plan B consists of using AlexNet for its success at classifying images. This approach has its own challenges, namely requiring a large amount of input data. Having said that, there is a lot of support available for neural networks on the Jetson Nano, so we are confident we can be successful with this approach. We will test this on our MVP fruit, banana, and then pick the approach with the best results for our other fruits.

5. *Gate controller*

Once the classification algorithm makes a decision about the quality of the fruit, a signal is sent to the gate controller system. The purpose of this system is to communicate the classification decision with the Adafruit servo driver, so that the driver can move the servo motor appropriately.

B. Hardware subsystem

This subsystem is responsible for coordinating the different moving parts of the design, and controlling the flow of information. It consists of the NVIDIA Jetson Nano 2 GB, camera subsystem, the Adafruit servo controller, the 5 V DC servo motor, and the 25000 mAh Krisdonia power bank.

1. *NVIDIA Jetson Nano 2 GB*

We will be using the NVIDIA Jetson Nano 2 GB as the central hub for all hardware related tasks. It will be used to coordinate all the other pieces. For a discussion of why we picked this particular model, refer to the earlier tradeoffs section.

2. *Camera subsystem*

The camera subsystem consists of 2 Raspberry Pi camera modules V2, and the arducam multi camera adapter module V2.2. The multi camera module has support for upto 4 cameras, and accesses them in a time multiplexed way (access is not simultaneous). The Nano will be issuing commands to the camera multiplexer which will take pictures using each of the attached cameras and save them on the disk. We have been able to install one camera directly (without the camera multiplexer) successfully on the Nano and use it to take pictures; the next step is to incorporate the full camera subsystem. As mentioned in the design tradeoffs, we chose this design primarily to mitigate the risk of not having enough angles (and thus enough data) of the fruit. We want our design to be scalable to 4 cameras in case we need more angles of the fruit to meet the accuracy metrics. The installation of the camera multiplexer (showcasing 4 cameras) is shown below.

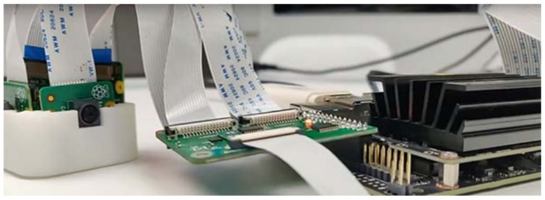


Fig 2. Camera Subsystem & Jetson Nano Interface[4]

The multiplexer is connected to both the GPIO pins of the Nano, and to the CSI port. Note that this is not our setup (we have not implemented this); this is a setup we found online.

3. *Adafruit 16-channel 12-bit PWM/Servo Driver*

The main purpose of this system is to facilitate communication between the Jetson Nano and the servo motor that will control the gate sorting mechanism once the classification algorithm section

[4] https://www.arducam.com/docs/camera-for-Jetson-Nano/multiple-cameras-on-the-Jetson-Nano/arducam-multi-camera-adapter-on-the-Nano/

produces an output. We need this module since it's not possible to control the servo motor directly from the Nano. A schematic for installation is shown below. This is what we will be building.

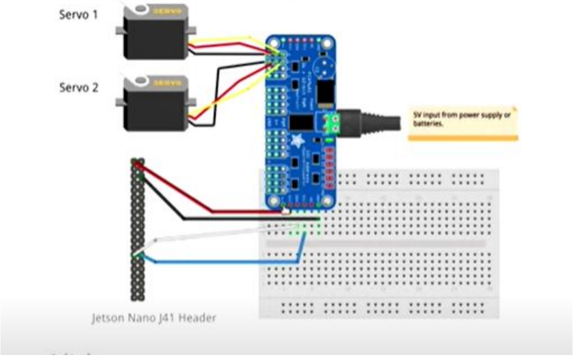


Fig 3. Adafruit Servo Controller & Jetson Nano Interface[5]

Note once again that this is not our current setup (we have not yet implemented this). We found this online. The servo driver also uses GPIO pins. It is possible to have contention between the servo driver and the camera multiplexer since they both use GPIO pins, but there are enough pins on the Nano so that this shouldnt be a problem.

4. *Sorting mechanism*
The sorting mechanism consists of the servo motor and a piece of metal attached directly onto the shaft. The long piece of metal directs the fruit in the appropriate basket by physically blocking it. More detail can be found in Figure 1 of Section 3.

5. *Power*
We will be using the 25000 mAh krisdonia power bank to power the entire system. We chose this so that we can power the Jetson Nano for upwards of 8 hours, which is one of our requirements. We will also be using this power bank to power the arducam camera multiplexer and the Adafruit servo motor shield.

C. *Mechanical subsystem*

The mechanical subsystem consists of the conveyor belt. The system is divided into 2 parts, the mechanical side, which is the actual belt, and the electronics side which is the system that drives the belt. The 12 V DC gear motor is the point of contact between the 2 subsystems.

1. *Mechanical*
The conveyor belt system consists of 2 pieces of wood that act as side panels, and a laminated piece of wood that goes inbetween. The side panels are exactly one inch longer than the laminated wood, which lets us place the rollers (PVC pipes of 1 inch diameter) in between the side panels of wood, tangent to the central piece of laminated wood. An image of this partial setup is presented below.

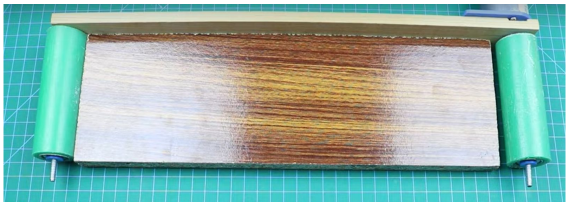


Fig 4. Partially Completed Conveyor Belt System[6]

The motor is connected to the 3D printed shaft coupler and attached to one of the rollers. The belt goes on top of the rollers. This completes the mechanical portion of the conveyor belt. A finished image is presented below:

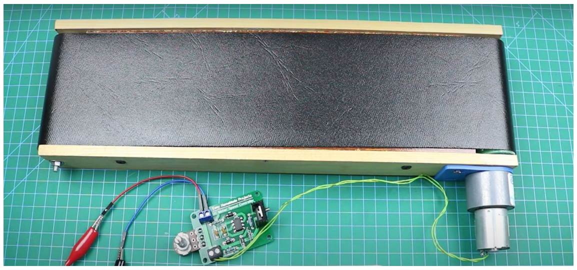


Fig 5. Top View of Completed Conveyor Belt System[7]

Note that we have not built either of the designs above ourselves; both of these images were found online. We will be building these designs in the coming weeks. A discussion of the dimensions of the conveyer belt and why we chose them can be found in Section 2. Other necessary miscellaneous parts (and their dimensions) required to complete the belt can be found in the budget and parts list later in the report.

2. *Electronic*
The electronics subsection consists of just the motor speed controller board (not depicted in the images above, but available in the budget section). This will let us control the speed of the motor and achieve the required 5 cm/s. Note that this board will be powered separately (not using the power bank), since we want to maintain independence between the mechanical system and the hardware/software systems. Since conveyor belts aren't connected to batteries in farms, any appropriate power source is sufficient. We do have a plan B in case the RC motor controller board doesn't work. We will use a motor controller PCB using the 555 speed controller chip. We found the appropriate schematic of the PCB (and the appropriate gerber files), and the components that need to be soldered on online.

[5] https://www.arducam.com/docs/camera-for-Jetson-Nano/multiple-cameras-on-the-Jetson-Nano/arducam-multi-camera-adapter-on-the-Nano/

[6] https://www.youtube.com/watch?v=o7VVmtX7SKs&t=216s

[7] https://www.youtube.com/watch?v=o7VVmtX7SKs&t=216s

VI. Project Management

A. *Schedule*

The first five weeks of the semester were spent in finalising the project idea, developing a well thought out design for it, and finalising our parts list. Post that, Ishita Kumar started looking into image segmentation methods to segment out fruits and separate them from their background. In the meantime, Kushagra worked on understanding how the NVIDIA Jetson Nano works, and integrating it with a Raspberry Pi camera module. After the image segmentation has been completed, Ishita Sinha plans on writing up code for edge detection. This would be required for us to be able to identify the rotten parts of the fruit v/s the good ones. This could be followed by developing the rottenness classifiers, while Kushagra would be working on the conveyor belt.

Post this initial setup phase with the primary algorithm being done, we'll all come together to work on a physical set up for the product. Next, we will work on automating the process and moving into testing and improving the MVP in the second week of April. Once the MVP has been finalised, we plan on updating our product to include apples and oranges as well, after which we plan on having all-inclusive tests. Once our product is working, we have kept around a week for recording the video and working on the final report and presentation. We have accounted for more than a week of slack, as can be seen in our schedule, since it ends on May 2nd. Our schedule is in Figure 7 in the Appendix on page 9.

B. *Team Member Responsibilities*

While each team member will have a role in every task to ensure that everyone has a holistic understanding of the working of the entire product, we have split responsibilities on the basis of who'll be leading the task. The primary and secondary responsibilities of each team member are shown in the chart below:

Team Member	**Primary Responsibility**	**Secondary Responsibility**
Ishita Kumar	Image segmentation	Automating cameras to click pictures every 4 seconds
Ishita Sinha	Edge detection	Automating gate rotation to ensure the gate rotates the appropriate amount in the right direction
Kushagra Sharma	Hardware component setup	Conveyor belt setup

C. *Budget*

Our budget and parts list has been included in Figure 8 in the Appendix on page 10. The main hardware tools we'll be using include an NVIDIA Jetson Nano, 2 Raspberry Pi camera modules, arducam camera multiplexer, adafruit servo controller, and the hardware needed to construct the gate. The mechanical component we'll be building is the conveyor belt, so we have also included costs corresponding to parts needed to build it.

D. *Risk Management*

As part of the project, one of our biggest risks with respect to meeting the use case requirements would be ensuring we meet the accuracy targets we have set. Our current accuracy targets are to achieve a false negative rate less than 5% and a false positive rate less than 15%. We need to be more contrite about the false negatives versus the false positives. This is because having a rotten fruit classified as a good one (false negative) would lead to all of the good fruits potentially getting spoilt, while classifying a good fruit as rotten (false positive) would lead to a loss incurred only due to that single fruit. In order to meet our accuracy targets, we plan on designing multiple classification algorithms so that we aren't dependent on a single software approach, and are keeping the option open for installing more cameras if we need more angles of the fruit.

Next, another major risk would be if we cannot meet the conveyor belt speed or if we miss fruits. Our product is designed to be integrated into existing conveyor belt systems, assuming they work at a speed of 5 cm/s. We're assuming 4 seconds to process a given fruit, so the distance between 2 fruits would be around 20 cm. However, we want to ensure we don't miss a fruit if the speed is too high, or classify the same fruit twice if the speed is too low, or if there are spacing issues. For the same, we plan on experimenting with different speeds to see which speed gives us the most optimal output, while ensuring we don't miss a single fruit.

Another great risk we're facing is building a conveyor belt. None of us have ever built anything mechanical, so it's possible we won't be able to build it, or it may not meet our system requirements. Since the actual product needs to be integrated into existing conveyor belt systems, a conveyor belt isn't really part of the product itself, so even if we can't build a conveyor belt well, it shouldn't affect the performance of our product. To mitigate this risk, we plan on using a treadmill as a conveyor belt, and have also included building the conveyor belt in an earlier stage in our schedule, so we have time to build it and know if we need to use a treadmill or not.

Lastly, we run the risk of not being able to integrate all of the different parts of the project in a smooth fashion since it has several components and we haven't built a product from scratch earlier. Moreover, we do run the risk of falling short of time. To account for this, we have provided for more than a week of slack time. We're all going to work on all parts so that we understand how everything works. We're updating the schedule regularly and holding each other accountable.

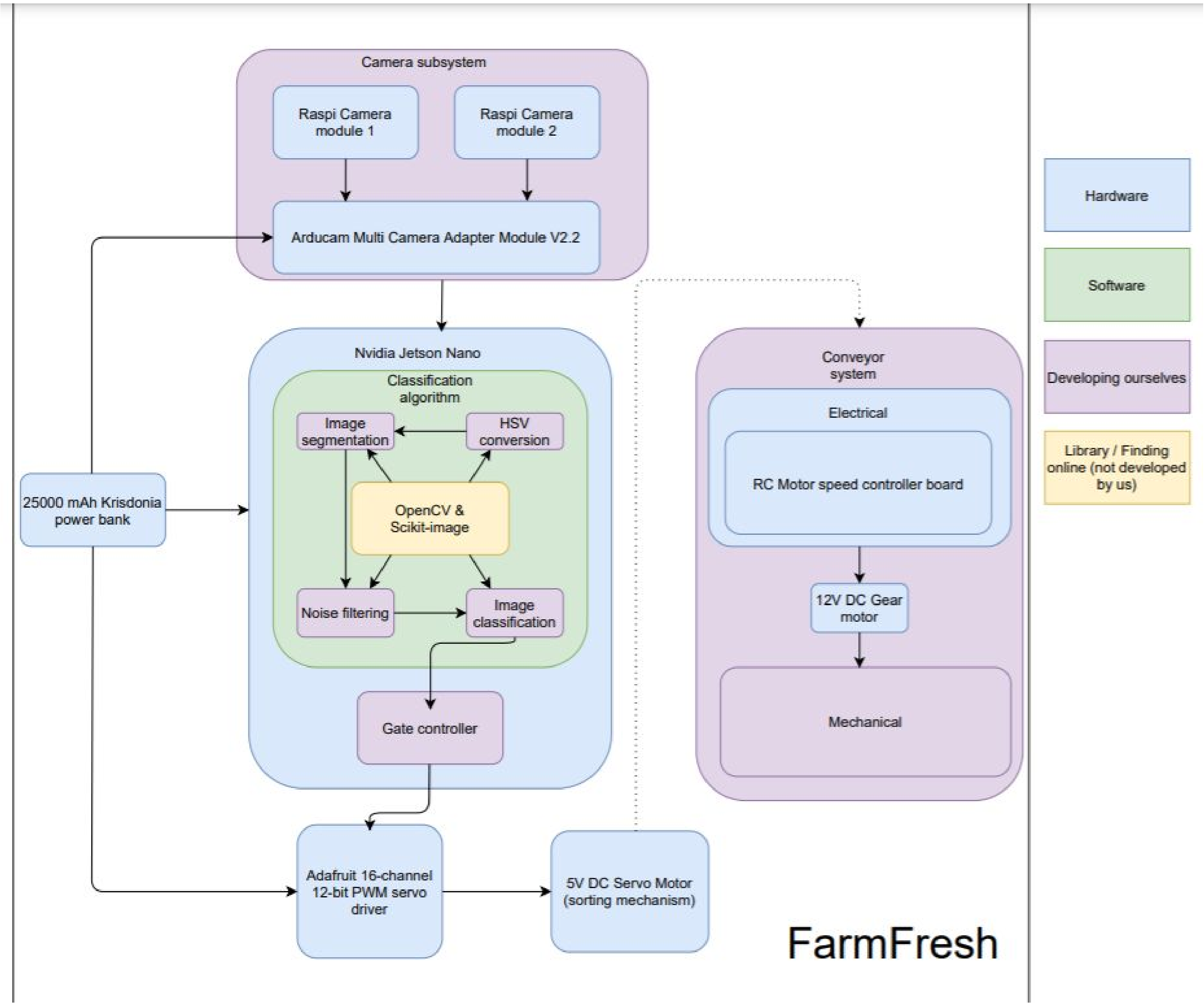
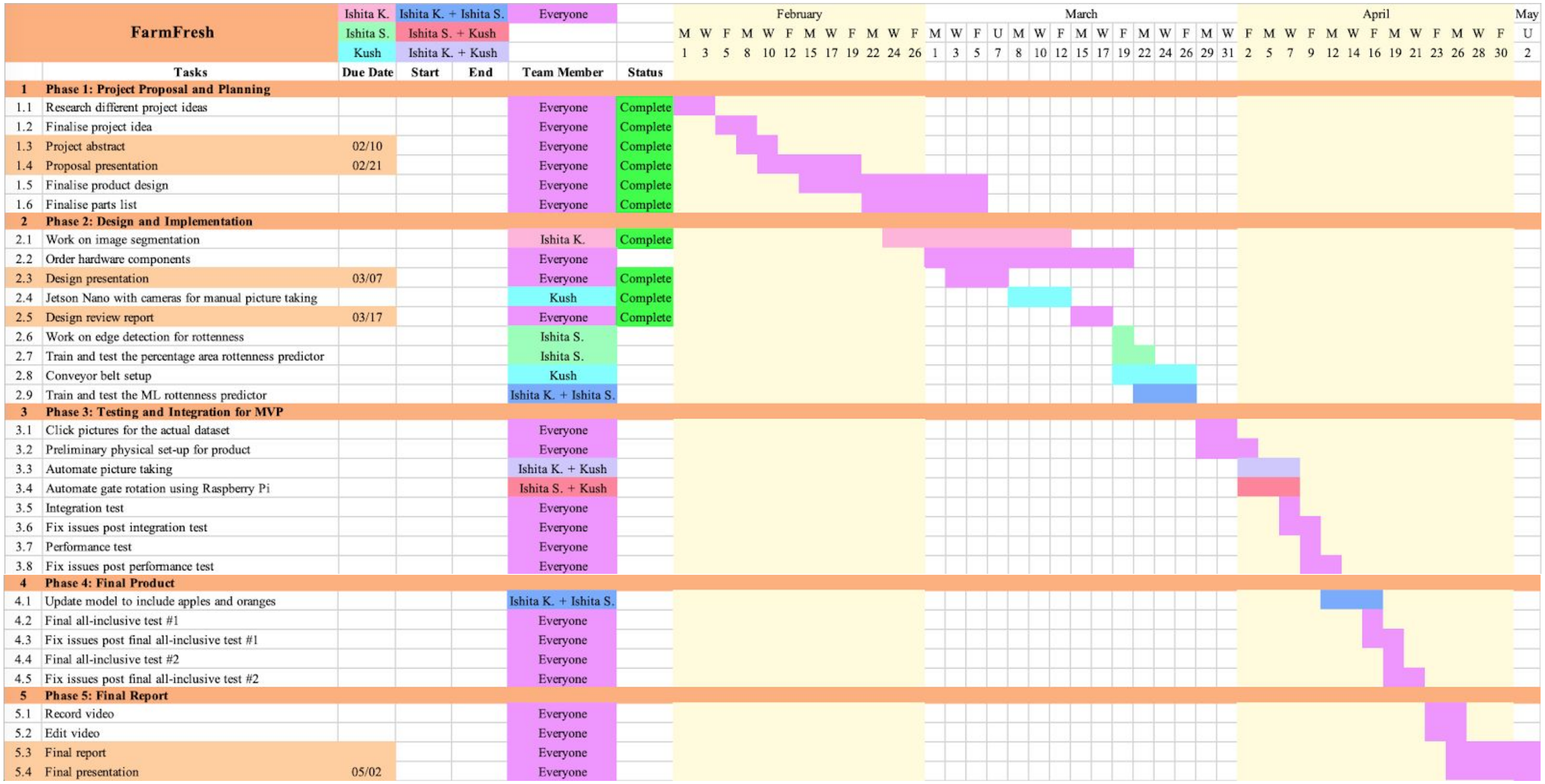
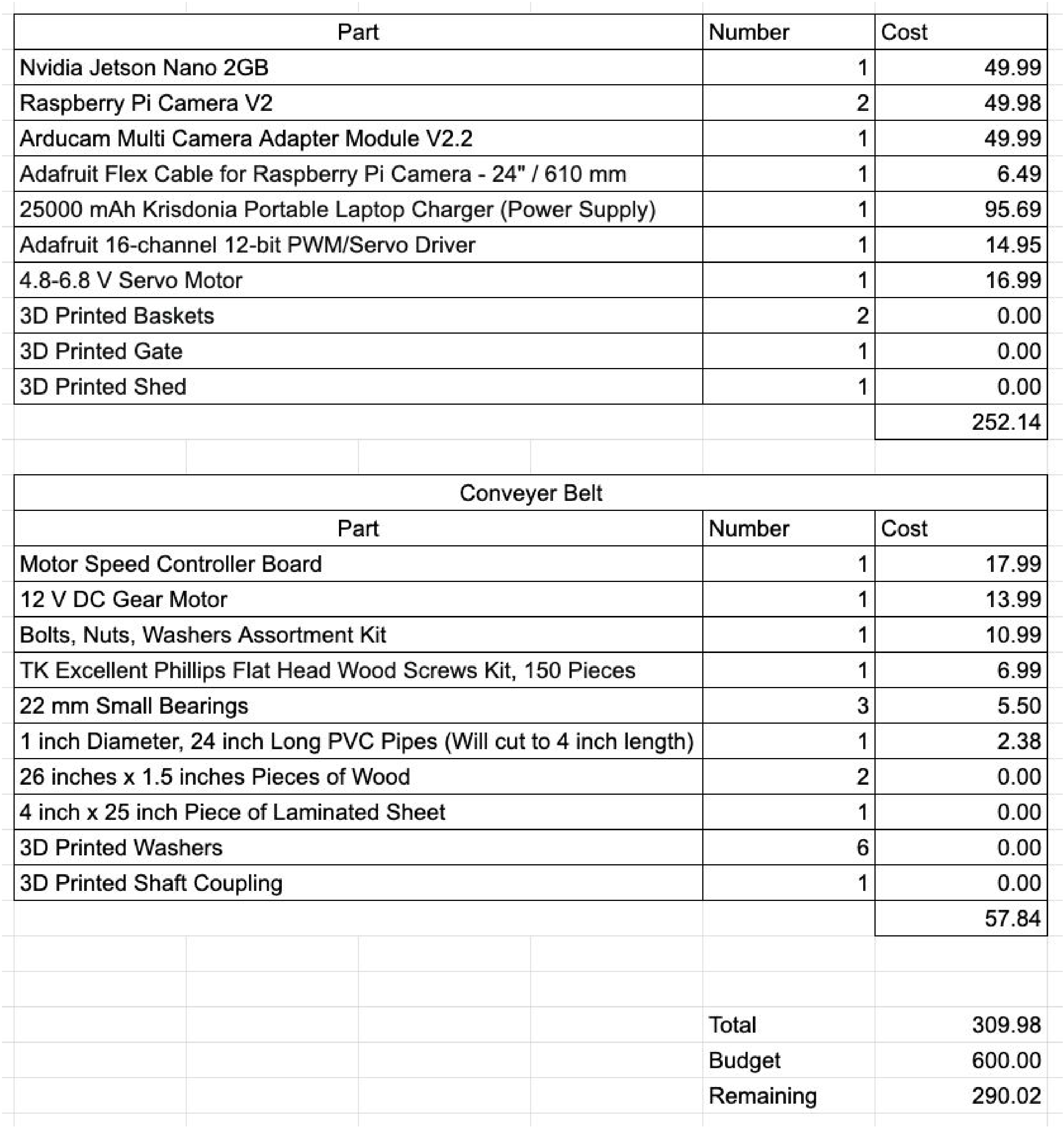


Fig. 6. Block Diagram



	Tasks	Due Date	Start	End	Team Member	Status
1	**Phase 1: Project Proposal and Planning**					
1.1	Research different project ideas				Everyone	Complete
1.2	Finalise project idea				Everyone	Complete
1.3	Project abstract	02/10			Everyone	Complete
1.4	Proposal presentation	02/21			Everyone	Complete
1.5	Finalise product design				Everyone	Complete
1.6	Finalise parts list				Everyone	Complete
2	**Phase 2: Design and Implementation**					
2.1	Work on image segmentation				Ishita K.	Complete
2.2	Order hardware components				Everyone	
2.3	Design presentation	03/07			Everyone	Complete
2.4	Jetson Nano with cameras for manual picture taking				Kush	Complete
2.5	Design review report	03/17			Everyone	Complete
2.6	Work on edge detection for rottenness				Ishita S.	
2.7	Train and test the percentage area rottenness predictor				Ishita S.	
2.8	Conveyor belt setup				Kush	
2.9	Train and test the ML rottenness predictor				Ishita K. + Ishita S.	
3	**Phase 3: Testing and Integration for MVP**					
3.1	Click pictures for the actual dataset				Everyone	
3.2	Preliminary physical set-up for product				Everyone	
3.3	Automate picture taking				Ishita K. + Kush	
3.4	Automate gate rotation using Raspberry Pi				Ishita S. + Kush	
3.5	Integration test				Everyone	
3.6	Fix issues post integration test				Everyone	
3.7	Performance test				Everyone	
3.8	Fix issues post performance test				Everyone	
4	**Phase 4: Final Product**					
4.1	Update model to include apples and oranges				Ishita K. + Ishita S.	
4.2	Final all-inclusive test #1				Everyone	
4.3	Fix issues post final all-inclusive test #1				Everyone	
4.4	Final all-inclusive test #2				Everyone	
4.5	Fix issues post final all-inclusive test #2				Everyone	
5	**Phase 5: Final Report**					
5.1	Record video				Everyone	
5.2	Edit video				Everyone	
5.3	Final report				Everyone	
5.4	Final presentation	05/02			Everyone	

Fig. 7. Schedule Gantt Chart



Part	Number	Cost
Nvidia Jetson Nano 2GB	1	49.99
Raspberry Pi Camera V2	2	49.98
Arducam Multi Camera Adapter Module V2.2	1	49.99
Adafruit Flex Cable for Raspberry Pi Camera - 24" / 610 mm	1	6.49
25000 mAh Krisdonia Portable Laptop Charger (Power Supply)	1	95.69
Adafruit 16-channel 12-bit PWM/Servo Driver	1	14.95
4.8-6.8 V Servo Motor	1	16.99
3D Printed Baskets	2	0.00
3D Printed Gate	1	0.00
3D Printed Shed	1	0.00
		252.14

Conveyer Belt		
Part	Number	Cost
Motor Speed Controller Board	1	17.99
12 V DC Gear Motor	1	13.99
Bolts, Nuts, Washers Assortment Kit	1	10.99
TK Excellent Phillips Flat Head Wood Screws Kit, 150 Pieces	1	6.99
22 mm Small Bearings	3	5.50
1 inch Diameter, 24 inch Long PVC Pipes (Will cut to 4 inch length)	1	2.38
26 inches x 1.5 inches Pieces of Wood	2	0.00
4 inch x 25 inch Piece of Laminated Sheet	1	0.00
3D Printed Washers	6	0.00
3D Printed Shaft Coupling	1	0.00
		57.84

Total	309.98
Budget	600.00
Remaining	290.02

Fig. 8. Budget and Parts List