EYE BILL: Assistive Bill Recognition Device for Visually Impaired and Blind People

By

Ravin Baldwin C. Angeles
Jessa C. Aquino
Albert Angelo C. Genova
Rafael Jonathan O. Lim
Jerimiah R. Riesgo
Angelica May A. Silao

B.S Computer Engineering

Technological Institute of the Philippines

Manila

A Final Documentation Submitted to the Department of Computer Engineering, in Partial Fulfillment of the Requirements of **Project Design** Course

January 2021

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES

1338 Arlegui St., Quiapo, Manila

DEPARTMENT OF COMPUTER ENGINEERING

APPROVAL SHEET

The proposed system/project/design entitled **EYE BILL: Assistive Bill Recognition Device for Visually Impaired and Blind People** which was presented on January 6, 2021, by the proponents:

Jessa C. Aquino

Ravin Baldwin C. Angeles

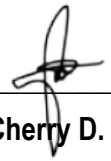
Albert Angelo C. Genova

Rafael Jonathan O. Lim

Jerimiah R. Riesgo

Angelica May A. Silao

is hereby APPROVED by the following members of the committee:



Dr. Cherry D. Casuat
Panel Member

Engr. Mon Arjay F. Malbog
Panel Member

Dr. Rufo I. Marasigan Jr.
Head Panel Member

Dr. Jennifer B. Enriquez
Chairperson, Computer Engineering

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES

1338 Arlegui St., Quiapo, Manila

DEPARTMENT OF COMPUTER ENGINEERING

ACCEPTANCE SHEET

The project design entitled **EYE BILL: Assistive Bill Recognition Device for Visually Impaired and Blind People** has been prepared and submitted by the proponents:

Jessa C. Aquino

Ravin Baldwin C. Angeles

Albert Angelo C. Genova

Rafael Jonathan O. Lim

Jerimiah R. Riesgo

Angelica May A. Silao

for approval to the committee for the Project Design

After a thorough review and evaluation of the proposed project design, the committee has accepted the presented proposed design based on the required criteria.

The acceptance is valid to the information being presented. January 6, 2021, / 1st semester of the school year 2020 - 2021

Dr. Cherry D. Casuat
Panel Member

Engr. Mon Arjay F. Malbog
Panel Member

Dr. Rufo I. Marasigan Jr.
Head Panel Member

Dr. Jennifer B. Enriquez
Chairperson, Computer Engineering

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

Despite the pandemic, our team successfully completed our project design and with that we would like to praise and thank God the Almighty for His showers of blessings. We would like to express our deep and sincere gratitude to our project adviser, Dr. Alvin Alon, Professor at Technological Institute of the Philippines Manila, to Dr. Jennifer Enriquez, our Design Project (COE 506) class adviser and the Computer Engineering Program Chair, to the panels and committee of our defense Dr. Cherry Casuat, Engr. Mon Arjay F. Malbog and Dr. Rufo I. Marasigan Jr. for giving us the opportunity to do this project and providing us invaluable guidance. Their dynamism, vision and sincerity and motivation have deeply inspired us. They taught us the methodology to carry out this project and to present the project works as clearly as possible. It was a great privilege and honor to work and study under their guidance. We are extremely grateful for what they have offered us.

We are also extending our thanks to the Resources for the Blind (inc.) for allowing us to continue and giving us enough resources needed for the development of this project. We also thank all the staff and teachers of Philippine National School For The Blind (PNSB) for allowing us to conduct an interview and allowing us to observe the blind and visually impaired inside their campus to gather data needed to complete this project.

Most importantly, we would like to say thanks to our friends and family, this project wouldn't be possible without their genuine guidance and support. Finally, our thanks go to all the people who have supported us to complete this project design directly or indirectly.

- **TEAM HAPTECH**

CHAPTER I: PROJECT BACKGROUND

This chapter includes the project's background which is about the struggles of a blind or a visually impaired person regarding the use of money in their everyday lives and the problems with the existing solutions. The objectives, the client, the scope and limitation of the project, and the process flow in conducting this project are also tackled in this chapter.

The Project

The importance of money can be easily realized from the fact that almost all the economic, social, and other activities are carried and completed through money. The importance of money is increasing day by day with the rapid economic development changes and other overall requirements of humans. Money is said to be the master key for the solution to all financial difficulties. Using money is simple; you just have to give a quick look at your money, take out the right amount of cash, and that's it. But for people who cannot see, this can become a difficult task. The word "blind" is a broad term. If you're legally blind, you may be able to see reasonably well with a pair of corrective lenses. What a blind person can see depends a great deal on how much vision they have. A person with total blindness won't be able to see anything. But a person with low vision may be able to see not only light but also colors and shapes. However, they may have trouble reading street signs, recognizing faces, or matching colors to each other. If you have low vision, your vision may be unclear or hazy. Some visual deficits cause part of your field of vision to be compromised. You might have a blind spot or a blurry spot in the middle of your field of vision, or your peripheral vision may be impaired on one or both sides. These issues can involve one or both eyes. If you have permanently reduced vision but retain some amount of your sight, you have low vision. Total blindness describes people with eye disorders who have no light perception (NLP). That is, a person who's blind doesn't see any light at all. Congenital blindness applies to people who are blind from birth. [1] Based on the 2017 data of the Department of Health, over two million people in the Philippines are blind or suffering from low vision. The DOH said an estimated 332,150 people in the country are bilaterally blind, while the current number of persons with low bilateral vision has already reached 2,179,733. [2]

In the Philippines, the currency is 'peso' (or officially *piso*), divided into 100 centavos (officially *centimo*). Its international abbreviation is PHP. The designers of the banknotes were Studio 5 Designs, who

took the designs for the P20, P50 & P1000, while Design Systemat took the remaining P100, P200, and P500 banknotes. Both were local design companies that were tapped by BSP to come up with six new designs for the six denominations [3]. To avoid counterfeiting, all Philippine banknotes include several security features, indicated on the front side:

1. embossed prints
2. serial number (in variable-sized figures)
3. security fibers
4. watermark
5. see-through mark ("Pilipino" spelled in Baybayin, letters, used before the arrival of the Spanish.)
6. concealed value
7. security thread
8. optically variable device (only on 500 and 1000 peso notes)

However, all denominations of the Philippine Peso Bill have identical sizes. This makes it impossible for blind people to distinguish one denomination from another. Manually, blind people fold a denomination of money in a particular way. For instance, a 20 peso bill can be left unfolded, a 50 peso bill can be folded widthwise, and a 100 peso bill can be folded lengthwise.

Folding money differently is a good solution to recognize a specific bill's denomination quickly; however, it requires someone the blind person trusts in identifying the currency for them to avoid being fooled or scammed by other people. There are instances that blind people want to live independently, and this includes taking up public transportation, going to work, and going to the market to buy their groceries; folding money differently is a good solution for them to pay, but it is a challenge to make sure that they received an exact amount for change, blind people get fooled as they often received an inadequate amount of money.

There are several apps which blind people can use on their phone to recognize currency. An example is 'Cash Reader' and 'TapTapsee'. Both applications are currently available in the PlayStore and Appstore, it uses the camera to check the currency and announces the value using speech or a series of haptic vibrations that help the blind and visually impaired to swiftly identify and count money even in noisy

environments. Governments also devised a way to help the blind tell apart different money denominations. In countries such as India, Australia, and Malaysia, each denomination of paper money has a distinct length. Along with this, blind people can use a small money identifier card to quickly measure and distinguish money. When the currency is lined up with the card, tactile marks on the card identify which bill is used based on length. Canada turned to a more direct approach. Wherein, money is being produced such that there are groups of Braille dots that indicate the value of the bill. Near one corner of the note, blind people can feel the dots. [4]

Design Problem

To test the accuracy of the existing applications, Taptapsee and Cash Reader, there were five (5) capturing devices used. This helped to identify what specification the application can produce the most accurate result. Both applications were simulated in a real-life scenario adapting various backgrounds (cement, leafy, concrete, tiles, wood, messy, and a bill placed on a person's lap).

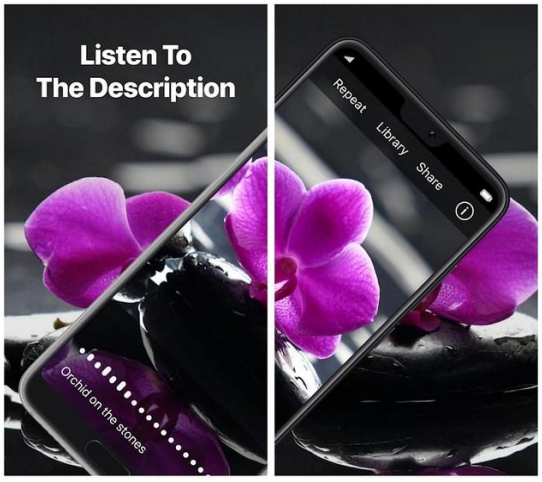


Figure 1.1 Taptapsee Application

Figure 1.1 shows the Taptapsee application that is designed to help the blind and visually impaired to identify objects they encounter in their daily lives. Simply double-tap the screen to take a photo of anything, at any angle, and hear the app speak the identification back to the user. The application can identify two or three-dimensional objects at any angle within seconds. After capturing the image/video as an input, the application starts performing its processes to identify the captured image/video and release a descriptive result using the device's VoiceOver. This application does picture recognition and can record up

to 10 seconds of video recognition; it needs an internet connection to produce results. The proponents were able to test TapTapsee and the following result materialized. [5]

Table 1.1 Taptapsee Testing Result

	PREDICTION RESULT		
DEVICE USED	**TRUE**	**FALSE**	**ACCURACY**
iPhone SE	40	20	66.67%
Honor 8x	45	15	75%
Huawei Nova 2i	43	17	71.67%
Honor Play	42	18	70%
Xiaomi Redmi	44	16	73.33%

Table 1.1 shows the result of tests using the TapTapsee application. Honor 8x got the highest accuracy with 75% and iPhone SE got the lowest with 66.67%. The TRUE prediction result signifies the correct prediction while the FALSE signifies the incorrect bill prediction. Accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Based on the result of the testing, the most-optimized among the capturing devices tested for the TapTapsee is the Honor 8x which got the highest accuracy of 75%.

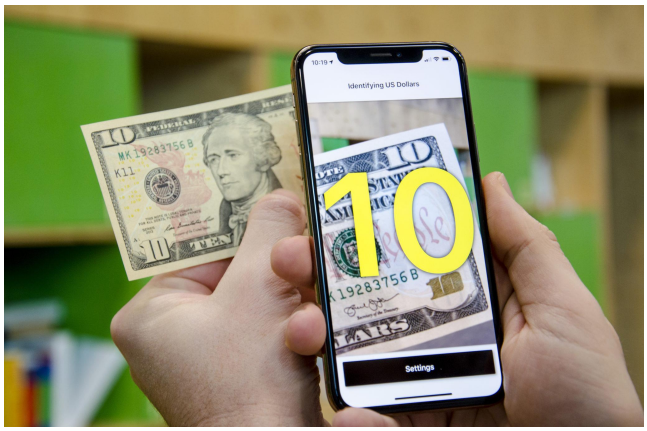


Figure 1.2 Cash Reader Application

Figure 1.2 shows the cash reader application, which identifies banknote denominations for the largest number of currencies. It works by opening the app and placing the banknote in front of the camera. The banknote denomination is instantly read aloud via your platform's speaker. At the same time, large, contrasting characters will appear on the screen. The reading/result can be processed without an internet connection and be presented through iOS VoiceOver and/or Android Talkback. In cases where the user is in a noisy place, or the user wants privacy, the application can be switched to silent mode, and the banknote denomination can be transformed into vibrations. [6] The proponents were able to test Cash Reader and the following result materialized:

Table 1.2 Cash Reader Testing Result

	PREDICTION RESULT		
DEVICE USED	**TRUE**	**FALSE**	**ACCURACY**
iPhone SE	43	17	71.67%
Honor 8x	42	18	70%
Huawei Nova 2i	42	18	70%
Honor Play	40	20	66.67%
Xiaomi Redmi	41	19	68.33

Table 1.2 shows the testing done using the Cash Reader application, iPhone SE got the highest accuracy with 71.67% and Honor Play got the lowest with 66.67%. The TRUE prediction result signifies the correct prediction while the FALSE signifies the incorrect bill prediction. Accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Based on the result of the testing, the most-optimized among the capturing devices tested for the TapTapsee is the iPhone SE which got the highest accuracy of 71.67%.

To further test the features of both applications, the proponents tested if it can differentiate a counterfeit or authentic Philippine Peso Bill. The results of tests are as follows:

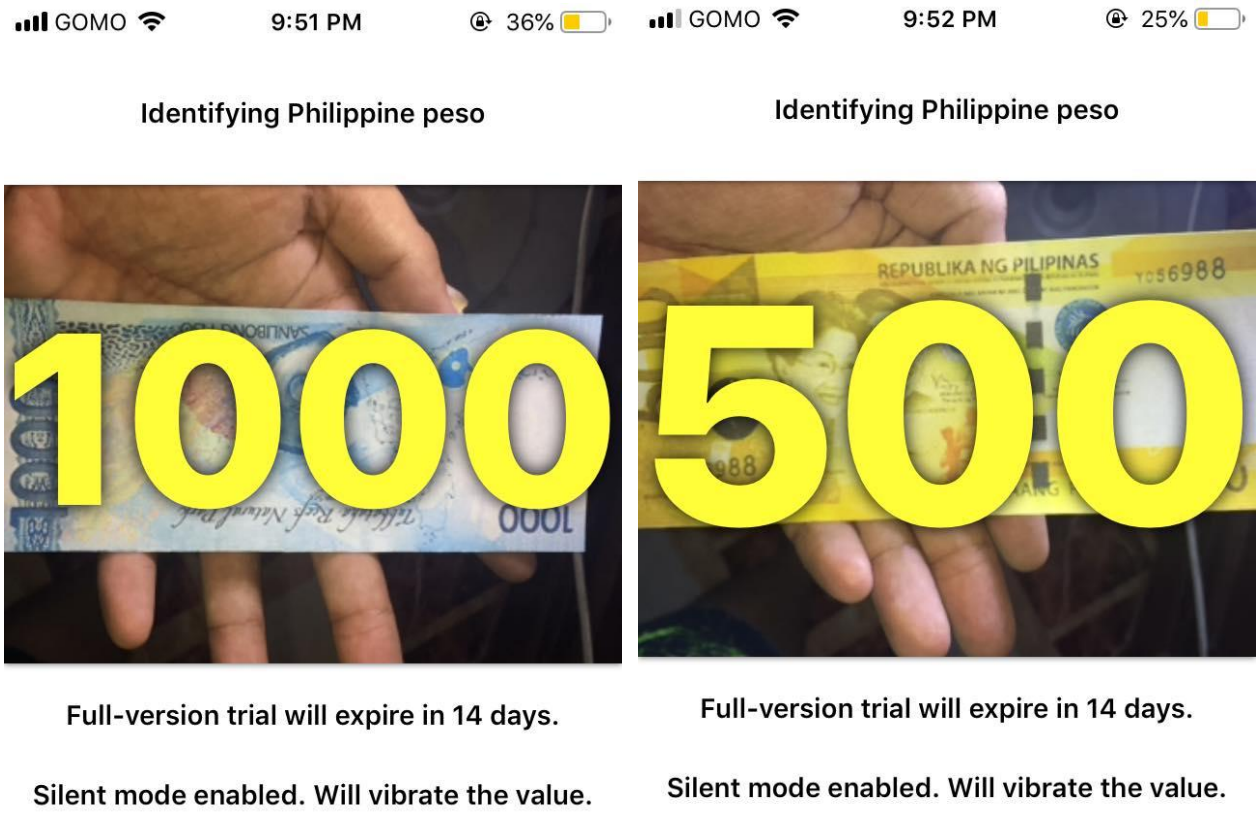


Figure 1.3 Counterfeit Testing Using Cash Reader

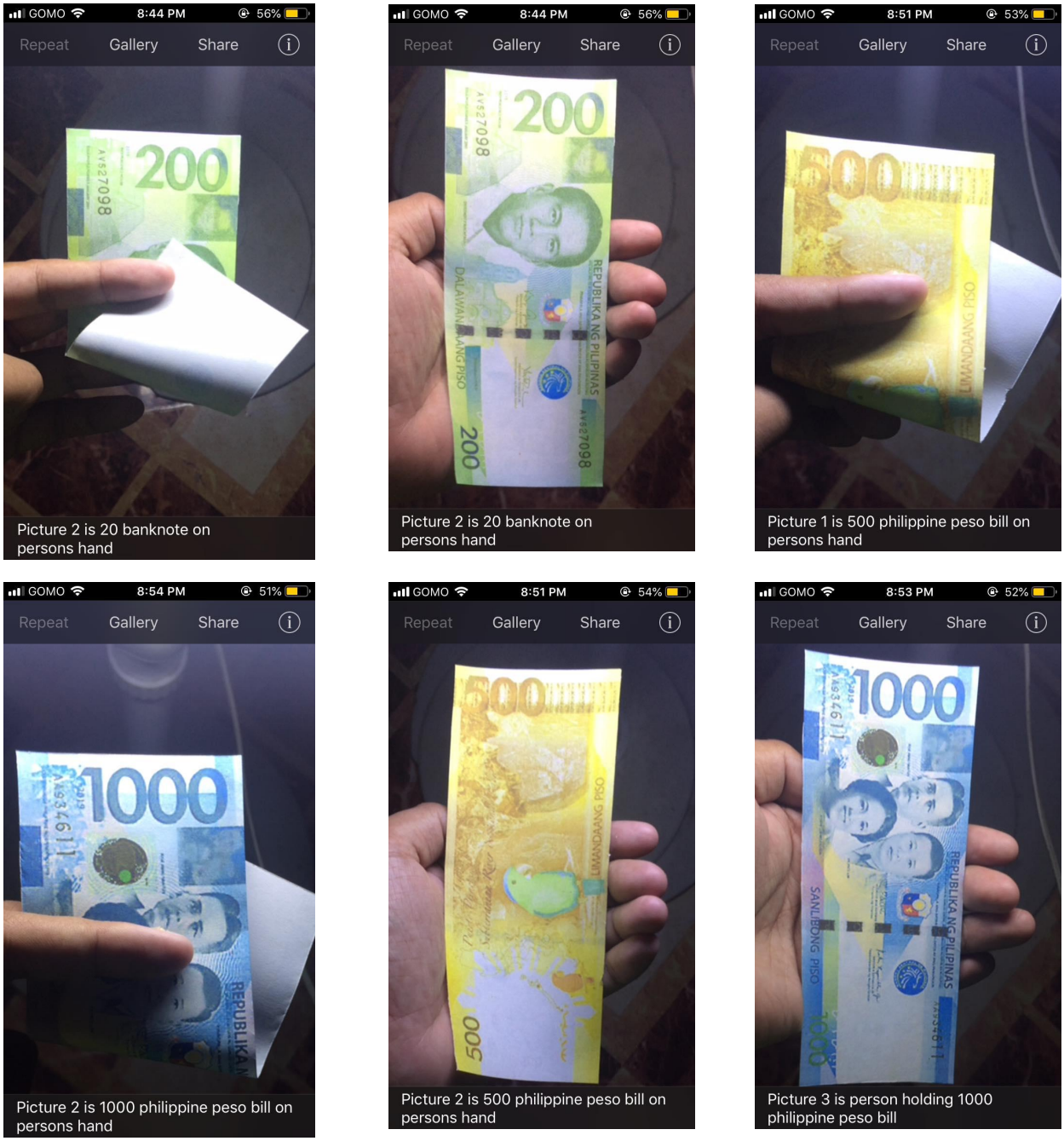


Figure 1.4 Counterfeit Testing Using TapTapsee

As shown in Figures 1.3 and 1.4, a copy of a Philippine Peso was printed and served as a fake bill. Both applications still produced an output indicating its denomination, however, it was not detected that the money was counterfeit. Therefore, with the result, the applications have no counterfeit money detection.

Through the testing, the following design problems were discovered:

1. The user's environment affects the accuracy in determining the denomination of the Philippine Peso Bill.
2. Accessibility is limited due to the need for an internet connection.
3. The existing solutions have no counterfeit money detection features.

Project Objectives

This describes the desired results of the project that the proponents are aiming to achieve. These must be completed for the betterment of the project.

1. Improve the accuracy in determining the denomination of the Philippine Peso Bill.
2. Simulate a platform that can:
    a. complete its function without the need of an internet connection.
    b. identify whether a Philippine Peso Bill is counterfeit or authentic.

The Client

The client is a non-profit organization named 'Resources for the Blind Inc..' They began as a personal project of Dr. Arthur Lown to produce part of the Filipino Bible in Braille. In 1993 he registered Resources for the Blind International (RBI) as a 501(c)(3) tax-exempt, no loan-profit organization. RBI supports the work in the Philippines in various ways and works to develop partnerships in other countries. Since that first project of producing Braille Filipino Bible portions, Resources for the Blind has expanded to three offices in the Philippines with over 60 staff and one office in the U.S.

Project Scope and Limitations

This part refers to the information included in the scope of the project including the restraints for the project. To ensure the attainment of the objectives, this project covers the following:

1. Development of three (3) algorithms to identify the denomination and the authenticity of the Philippine peso bill.
2. There are three (3) design options that were tested using five (5) different capturing devices which resulted in fifteen (15) models.
3. Smartphones were used for testing while taking into consideration the similarities of the megapixel between the capturing devices and phone cameras.

However, the following are the limitations:

1. The platform only identifies the 'New Generation Currency Series' banknotes released by the BSP issued between 2010 -2019.
2. The platform is only a simulation of a device made for the blind or visually impaired in identifying the denomination of the Philippine Peso bill.
3. The accuracy result was measured based on the data gathered by the proponents through algorithm testing using Google Colab.
4. The speed of the platforms' algorithm was measured based on the result of the execution time of Google Colab.
5. The platform was hindered by the current situation, thus the testing was done by the proponents, not by the blind.
6. The testing of the platform and the existing solutions was done by the proponents, not by the blind or visually impaired.

Project Development

The Waterfall model was created to explain the process layer of the project as shown in Figure 1.5.

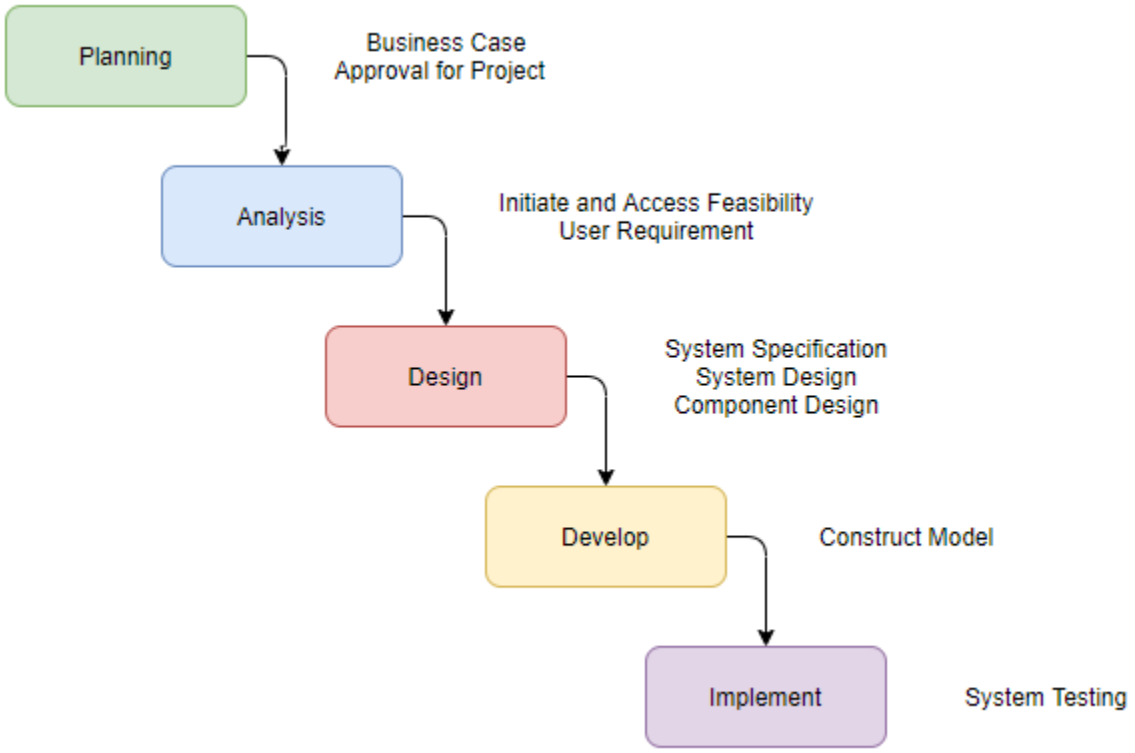


Figure 1.5 Eyebill Waterfall Model

In a software development plan, the waterfall model is the most common and most systematic way diagram in the software development plan. This type of technique is a phase-by-phase method technique, whereas the project cannot continue if the current phase is not yet completed. Below are the stages of the Waterfall Model:

Planning: Business Case & Approval for the Project

The planning phase involves the building of the business case and approval for the project. After validating the problem, the business case should be solidified as to why the product/project is necessary. Once the business case is formulated, the approval for the project shall be issued by the adviser before continuing to the next phase.

Analysis: Initiate & Access Feasibility & User Requirements

The second phase involves initiating the project, starting from the feasibility check. Once it is confirmed to be feasible, the group must determine the requirements of the product's users. Only then shall the group move to the next phase.

Design: System Specification, System Design, & Component Design

The third phase consists of 3 processes: System Specification, System Design, & Component Design. On the System Specification, the functional and non-functional requirements for the project shall be concluded. Once done with the System Specification, only then shall they continue to create the System Design where the architecture, modules, interfaces, and data for the product's system shall be defined.

Develop: Construct Model

In this phase, the model for the platform shall be constructed and is aligned to the third phase.

Implementation: System Testing

The product shall be tested to verify its reliability. The system testing must be done to determine if the objective, functional & non-functional requirements, and constraints are met.

CHAPTER II: DESIGN INPUTS

This chapter includes the system's requirement that lists all the features of Eyebill. It also includes related studies and articles that have significant relation and influence on the development of the platform. Details about the platforms and technologies involved in the platform's design were also stated.

Client's Requirements

Based on the client, the existing solution is not enough to cater to the needs of blind people. As a solution in the Philippines, there are multiple applications available for mobile devices that could help in distinguishing the value of the money; however, blind people are having a hard time navigating a smartphone due to the risk of it being stolen and the accessibility is often limited due to the need of internet connection. Based on the problems given by the client, the proponents were able to conclude the following requirements:

- Develop a platform that can be used specifically for the blind for them to accurately determine the amount of money.
- Make sure that the platform can work offline.
- Considering that the target users are blind people, the platform should be easy to access and easy to use.
- The platform should be able to produce an output fast and must avoid significant delays.
- The platform should be cost-effective.

Design Criteria and Design Constraints

To develop the platform, the design criteria based on the requirements given by the client was followed. However, some constraints may help in shaping the platform to fit the exact needs of the client. Both are presented in the table below:

Table 2.1 Design Criteria and Design Constraints

DESIGN CRITERIA	DESIGN CONSTRAINTS
The accuracy of the platform should improve compared to the existing solutions.	Accuracy
The value of the platform should be cost-effective.	Cost
The platform should be able to produce an output that avoids significant delays.	Speed

Accuracy

This project is concerned with the output accuracy of the overall system. The design to be implemented must be with high accuracy and an acceptable rate of failure. Bad equipment, poor data processing, or human error can lead to inaccurate results that are not very close to the truth. Accuracy can help increase users' satisfaction. The result of the accuracy depends on the type of camera and its specifications, such as the camera sensor, image resolution, and the camera pixel. To measure the accuracy, the proponents must test each algorithm and evaluate its result individually to determine the best algorithm that could provide the users with the best accuracy of the system. Accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Or for binary classifications, can be calculated in terms of positive and negatives:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negatives.

True Positive (TP) can be defined as an outcome where the algorithm correctly predicts the positive class. True Negative (TN) is an outcome where the algorithm correctly predicts the negative class. A False Positive (FP) is an outcome where the algorithm incorrectly predicts the positive class. And a False Negative (FN) is an outcome where the model incorrectly predicts the negative class.

To compute the values mentioned above, factors such as the camera sensor, camera pixel, camera resolution are important. Each factor affects the accuracy of the result.

- The camera sensor and camera pixel - A larger sensor allows the manufacturer to offer a wider ISO range, and the camera will be able to shoot at higher ISO speeds whilst keeping noise low. The more pixels you put on a sensor, then the smaller the pixel will be. For example, an 18-megapixel sensor will have smaller pixels than a 12-megapixel sensor, assuming both sensors were the same size. The size of each Pixel is measured and shown as a Micron with the symbol µm or simply µ, and is short for micrometer. [7]
- Camera resolution - The amount of detail that the camera can capture is called the resolution, and it is measured in pixels. The more pixels a camera has, the more detail it can capture, and the larger pictures can be without becoming blurry or "grainy." In essence, larger pixels can collect more light than smaller pixels, which translates to better image quality and handling of noise per pixel. Some typical resolutions include:
    - 256x256 - Found on very cheap cameras, this resolution is so low that the picture quality is almost always unacceptable. This is 65,000 total pixels.
    - 640x480 - This is the low end on most "real" cameras. This resolution is ideal for emailing pictures or posting pictures on a Web site.
    - 1216x912 - This is a "megapixel" image size -- 1,109,000 total pixels -- good for printing pictures.
    - 1600x1200 - With almost 2 million total pixels, this is "high resolution." You can print a 4x5 inch print taken at this resolution with the same quality that you would get from a photo lab.
    - 2240x1680 - Found on 4-megapixel cameras -- the current standard -- this allows even

larger printed photos, with good quality for prints up to 16x20 inches.

- 4064x2704 - A top-of-the-line digital camera with 11.1 mp takes pictures at this resolution. In this setting, you can create 13.5x9 inch prints with no loss of picture quality. [8]

Cost

The project should be able to produce a platform that is cost-effective and is worth the price. In a business dictionary [9], Cost is an amount that has to be paid or given up to get something. In business, the cost is usually a monetary valuation of effort, material, resources, time and utilities consumed, risks incurred and opportunity forgone in production and delivery of a good or service. For this project, the materials must be cost-efficient as possible, with the overall performance of the design justifying the cost. The total cost can be defined as:

$$Cost = \sum cost\ of\ components\ (PHP)$$

To compute the cost of the system, a Bill of Materials (BOM), which is a comprehensive inventory of the raw materials, assemblies, subassemblies, parts, and components, as well as the quantities of each needed to manufacture a product was created. In a nutshell, it is the complete list of all the items that are required to build a product. [9]

Speed

A platform that can produce an output without any significant delay must be created. This will avoid the hassle for the users and help them utilize their time in using the platform. Significant delays may cause errors and unnecessary waiting time for the users. The speed based on the algorithm used must be measured. Factors such as algorithm efficiency and CPU can affect the speed of the system.

- Algorithm efficiency - An algorithm must be analyzed to determine its resource usage, and the efficiency of an algorithm can be measured based on the usage of different resources. For maximum efficiency, we wish to minimize resource usage. However, different resources such as time and space complexity cannot be compared directly, so which of two algorithms is considered to be more efficient often depends on which measure of efficiency is considered most important. An algorithm is considered efficient if its resource consumption, also known as computational cost, is at or below some acceptable level.

Roughly speaking, 'acceptable' means: it will run in a reasonable amount of time or space on an available computer, typically as a function of the size of the input.

- CPU - The clock speed measures the number of cycles your CPU executes per second, measured in GHz (gigahertz). A “cycle” is technically a pulse synchronized by an internal oscillator, but for our purposes, they're a basic unit that helps understand a CPU's speed. Even though today's processors are tremendously fast, their performance can be affected by several factors such as clock speed, cache size, and the number of cores.
    - Clock speed - Clock speed is the number of pulses the central processing unit's (CPU) clock generates per second. It is measured in hertz. CPU clocks can sometimes be sped up slightly by the user. This process is known as overclocking. The more pulses per second, the more fetch-decode-execute cycles that can be performed, and the more instructions that are processed in a given space of time. Overclocking can cause long term damage to the CPU as it is working harder and producing more heat.
    - Cache size - Cache is a small amount of high-speed random access memory (RAM) built directly within the processor. It is used to temporarily hold data and instructions that the processor is likely to reuse. The bigger its cache, the less time a processor has to wait for instructions to be fetched.
    - The number of cores - A processing unit within a CPU is known as a core. Each core is capable of fetching, decoding, and executing its instructions. The more cores a CPU has, the greater the number of instructions it can process in a given space of time. Many modern CPUs are dual (two) or quad (four) core processors. This provides vastly superior processing power compared to CPUs with a single core. [10]

Storyboard

The storyboard was created to visualize the solution to the problem. Based on the information gathered, effective solutions can be designed to be able to attain the client's requirement.

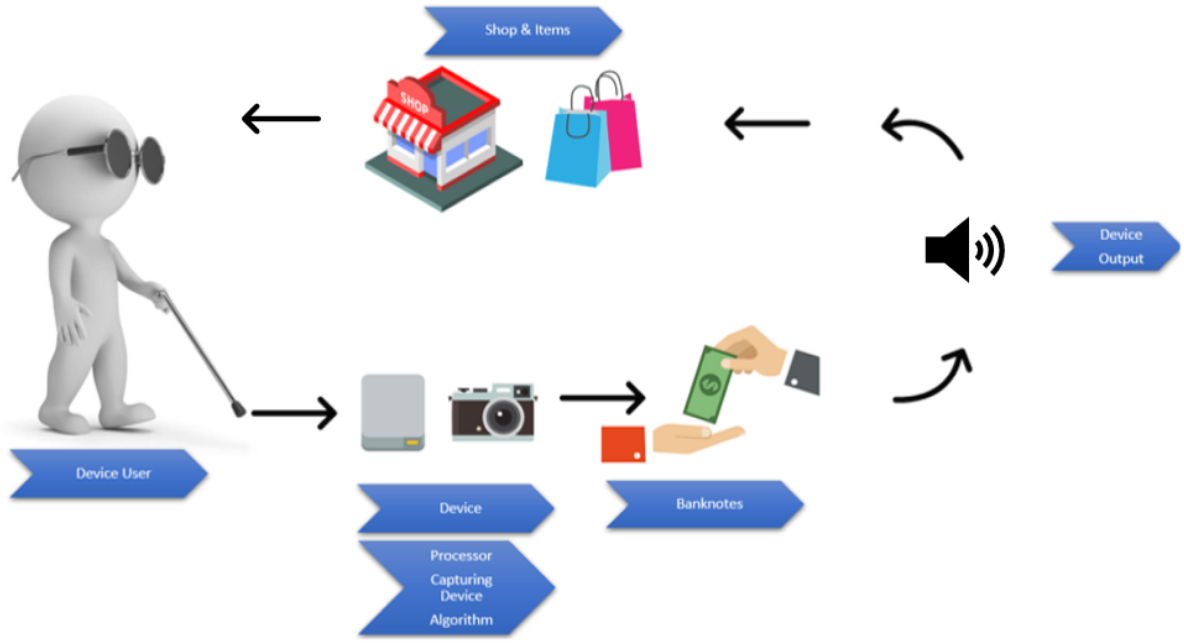


Figure 2.1 Current Scenario and Current Solution

Figure 2.1 shows the current scenario and current solution wherein the platform is brought by the user whenever they step outside to perform tasks that involve making payments. The platform scans the presented banknote. The platform then determines whether the banknote presented is counterfeit or forged. If the banknote is genuine, the platform determines the value of the banknote. The reading of the platform should be delivered to the user via audio, stating the value of the banknote that is presented. On an occasion where the banknote presented is counterfeit, the platform warns the user and won't produce a value reading.

Relevant Information

There are a lot of studies regarding bill identification specifically for the blind. The lack of identification devices motivated the need for a device that can be used as a bill identifier. Many of us are quite fortunate as we see and navigate the world with our own eyes. However, the recognition of banknotes presents a huge difficulty for blind / visually impaired people. The visually impaired community relies on many artificial aids that allow them to lead an average life in the complex modern society. Identification of currency notes is of utmost importance in this regard, and there are a lot of studies regarding this matter. However, to create such devices, there are technical requirements that must be presented and defined.

To capture an image, a capturing device is needed. A capture device (also referred to as a capture card) is a hardware device that you connect to your computer that converts the video signal from your camera's output into a digital format your computer can recognize. If you intend on using a camera that outputs HDMI, HD-SDI, component, etc., then you will need a capture device for Livestream Producer or Livestream Studio to recognize your camera(s).

Ala TechSource [11] published a journal about capturing devices that state the advances in capturing devices from flip phones to touch screens. Over the last decade, no media type has been more democratized than video. The cost of producing a video has gone from thousands of dollars for poor-quality pictures to less than $200 for a camera that will take high-definition video and can also double as a still camera. With the decline in the price of producing video has come the ability to publish your video at no cost online, through services like YouTube, Vimeo, Blip tv, and others. These two factors have led to the largest explosion of video production the world has ever seen, with YouTube alone having 20 hours of video uploaded to it every minute of every day. While video cameras have been steadily dropping in price for years, it wasn't until 2006, when the Flip company began selling its eponymous camera, that the landscape changed significantly. The Flip was the first all-in-one camera that was everything the consumer needed, and nothing else. Kodak was a latecomer to the inexpensive video camera game, but it nearly perfected the inexpensive camera with the introduction of the Kodak Zi8. The Zi8 does a full high-definition video (1080p) at $199. For higher resolution video at a low price, the Kodak is a very, very good camera. It's one of the best options under $200 for capturing video.

In the United States, inventor Reiner Krapf [12] created an image capturing device which includes an optical module configured to capture an image, at least one sensor module configured to generate image signals for the captured image, and a data processing module configured to generate image data for the captured image based on the generated image signals. The image capturing device further includes a display device configured to display the captured image based on the generated image data and an assigned memory unit configured to store the generated image data. The data processing module has a marking unit configured to optically mark at least one image position in the displayed image on the display device and to produce on the display device a display of at least one designating element to allow a designation of at least one image position. The designating element is configured to allow a clear identification of the image position designated.

It's hard to evaluate a new piece of technology without considering its processor. [13] Processors are the brains behind a computer. They control the logic that performs calculations and run programs on your computer. A processor is a piece of hardware that interprets the instructions that drive a computer. Processors are known as the brains of a computer for a good reason: without them, computers could not run programs. Processors are often referred to as CPUs. Technically, there is more than one processor in a computer, such as a Graphics Processing Unit (GPU), but the CPU is arguably the most important one. Processing units take instructions from a computer's Random Access Memory (RAM). When these instructions are received, the CPU decodes and processes the action and then delivers an output.

According to Peter Y.K, CheungGeorge, A.Constantinides, WayneLuk [14], as both the cost and performance of microprocessors continue to improve, it becomes increasingly attractive to build computer systems containing many processors. Performance improvement is one obvious benefit of a multiprocessor system. In the ideal case, a system with N processors can provide N times speedup of compute-bound tasks. Given that the cost of a microprocessor is a small fraction of the total system cost, the cost-effectiveness of such an approach is obvious. A different but equally important reason for using multiprocessor systems is for better reliability. The idea here is that even if a single processor fails, the system will continue to work, though at a slower pace. The workload of the failed processor would be automatically taken up by the remaining processors.

For a processor to work better, the algorithm should be efficient. In layman's language, an algorithm can be defined as a step-by-step procedure for accomplishing a task. In the world of programming, an algorithm is a well-structured computational procedure that takes some values as input and some values as output. Algorithms give us an ideal option for accomplishing a task. Algorithms are often quite different from one another. One algorithm may use many fewer resources than another. One algorithm might take ten times as long to return the result as the other, and it would be better to compare two solutions together. Based on lbackstrom, a TopCoder member [15], one of the most important aspects of an algorithm is how fast it is. It is often easy to come up with an algorithm to solve a problem, but if the algorithm is too slow, it's back to the drawing board. Since the exact speed of an algorithm depends on where the algorithm is run, as well as the exact details of its implementation, computer scientists typically talk about the runtime relative to the size of the input. For example, if the input consists of N integers, an algorithm might have a runtime proportional to N2, represented as O(N2). This means that if you were to run an implementation of the algorithm on your computer with an input of size N, it would take C*N2 seconds, where C is some constant that doesn't change with the size of the input.

Additionally, the execution time of many complex algorithms can vary due to factors other than the size of the input. For example, a sorting algorithm may run much faster when given a set of integers that are already sorted than it would when given the same set of integers in random order. As a result, you often hear people talk about the worst-case runtime or the average-case runtime. The worst-case runtime is how long it would take for the algorithm to run if it were given the most insidious of all possible inputs. The average-case runtime is the average of how long it would take the algorithm to run if it were given all possible. Of the two, the worst-case is often easier to reason about and therefore is more frequently used as a benchmark for a given algorithm. The process of determining the worst-case and average-case runtimes for a given algorithm can be tricky since it is usually impossible to run an algorithm on all possible inputs. Many good online resources can help you in estimating these values.

According to Ajeet RamPathaka, ManjushaPandeya, SiddharthRautaray who wrote a paper on the Application of Deep Learning for Object Detection, deep learning technology has become a buzzword nowadays due to the state-of-the-art results obtained in the domain of image classification, object detection, natural language processing. The reasons behind the popularity of deep learning are two folded, viz. large availability of datasets and powerful Graphics Processing Units. As deep learning requires large datasets and powerful resources to perform training, both requirements have already been satisfied in this current era. [16] Motivated by the results of image classification, deep learning models have been developed for object detection and deep learning based object detection has also achieved state-of-the-results [17]. They aim to assess deep learning techniques based on convolutional neural networks (CNN) for object detection. The beauty of convolutional neural networks is that they do not rely on manually created feature extractors or filters. Rather, they train per se from raw pixel level up to final object categories. Meanwhile, according to Deepanshu Tyagi on medium.com [18] feature matching,is a piece of information which is relevant for solving the computational task related to a certain application. Features may be specific structures in the image such as points, edges or objects. Features may also be the result of a general neighborhood operation or feature detection applied to the image. The features can be classified into two main categories: The features that are in specific locations of the images, such as mountain peaks, building corners, doorways, or interestingly shaped patches of snow. These kinds of localized features are often called keypoint features (or even corners) and are often described by the appearance of patches of pixels surrounding the point location and the features that can be matched based on their orientation and local appearance (edge profiles) are called edges and they can also be good indicators of object boundaries and occlusion events in the image sequence.

CHAPTER III: PROJECT DESIGN

This chapter focuses on the design of the Eyebill system, starting from the process flow of the system, dataset gathering, testing of the models developed by the proposed algorithms to the design options of the proponents. The different models from the proposed algorithms are shown through graphical representations expressed as an architecture.

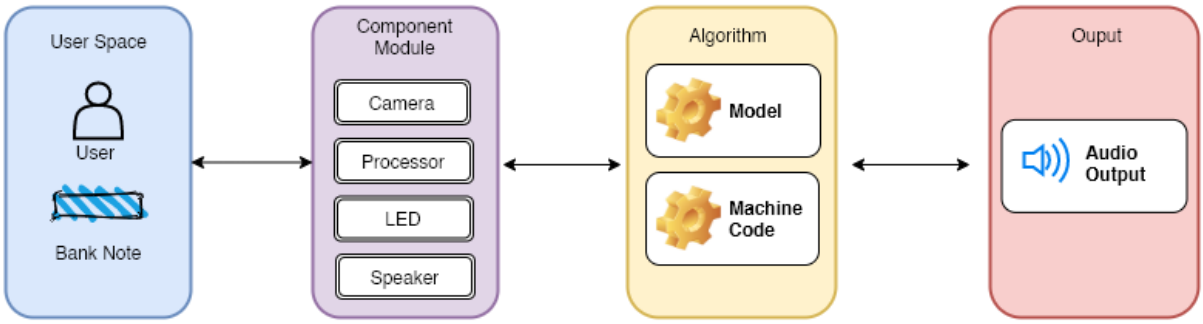


Figure 3.1 Eyebill System Architecture

Figure 3.1 signifies the Eyebill system architecture that is composed of a user-space where the user inserts the Philippine Peso bill into the system and that is captured by the camera, then the captured image is fed to the processor wherein it is processed by the Eyebill algorithm. The processed image is used by the model where it identifies the denomination through machine codes. The platform identifies whether the bill is counterfeit or not, then it produces the output indicating the denomination of the inserted Philippine Peso bill through audio.

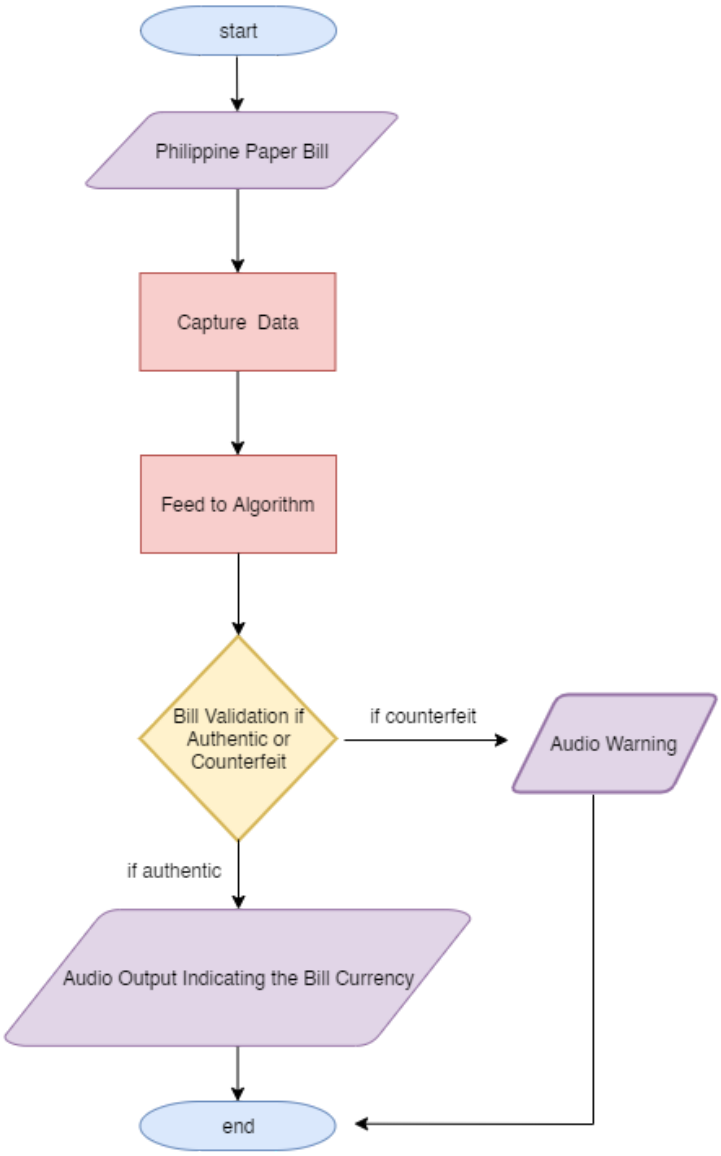


Figure 3.2 Eyebill: System Flowchart

As shown in figure 3.2, the Eyebill system starts when the user inserts the Philippine Peso bill into the system. The platform then captures the inserted bill and feeds it to the model to identify the denomination through machine codes. If the input is invalid, the platform produces an audio warning, else, the platform produces the output indicating the denomination of the inserted Philippine Peso bill through audiso.

I. Selection of Algorithms for Design Options

In this design project, the algorithm for the models was considered as a design option. There are three (3) design options and were tested using five (5) different capturing devices, which resulted in five (5) different models in every design option. In determining which algorithms to consider in the design options, the study on Object Detection results on COCO test_dev [19] produced the best algorithms as shown in the figure below. With this, it served as a reference to the proponents.

RANK	MODEL	BOX AP	AP50	AP75	APS	APM	APL	EXTRA TRAINING DATA	PAPER	CODE	RESULT	YEAR
1	Cascade Eff-B7 NAS-FPN (1280, self-training Copy Paste, single-scale)	57.3						✓	Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation			2020
2	YOLOv4-P7 (CSP-P7, multi-scale)	55.8	73.2	61.2	38.8	60.1	68.2	×	Scaled-YOLOv4: Scaling Cross Stage Partial Network			2020
3	DetectoRS (ResNeXt-101-64x4d, multi-scale)	55.7	74.2	61.1	37.7	58.4	68.1	×	DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution			2020
4	YOLOv4-P7 (CSP-P7, single-scale)	55.4	73.3	60.7	38.1	59.5	67.4	×	Scaled-YOLOv4: Scaling Cross Stage Partial Network			2020
5	EfficientDet-D7x (single-scale)	55.1	74.3	59.9	37.2	57.9	68.0	×	EfficientDet: Scalable and Efficient Object Detection			2019

Figure 3.3 Object Detection result on COCO test-dev

As seen in figure 3.3, the YOLOv4 and EfficientDet are among the top algorithms in the year 2020. The study was done by Papers with Code, a free and open resource with Machine Learning papers, code, and evaluation tables and they are supported by NLP and ML. Meanwhile, the YOLOv5 is the upgraded version of YOLOv4 which was released in 2020. It is state of the art and the newest version of the YOLO object detection series, and with the continuous effort and 58 open source contributors, YOLOv5 set the benchmark for object detection models very high [20]

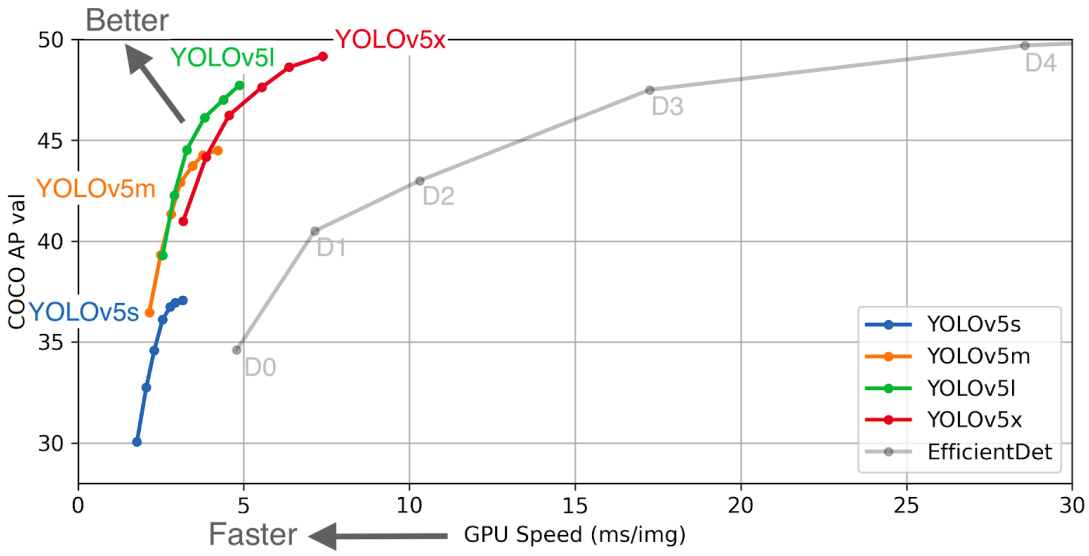


Figure 3.4 YOLO Version Comparison

As shown in figure 3.4, YOLOv5 already beats the EfficientDet and its other previous YOLOv5 versions. GPU Speed measures end-to-end time per image averaged over 5000 COCO val2017 images using a V100 GPU with batch size 32 and includes image preprocessing, PyTorch FP16 inference, post processing, and NMS [21].

II. Process of Training the Model

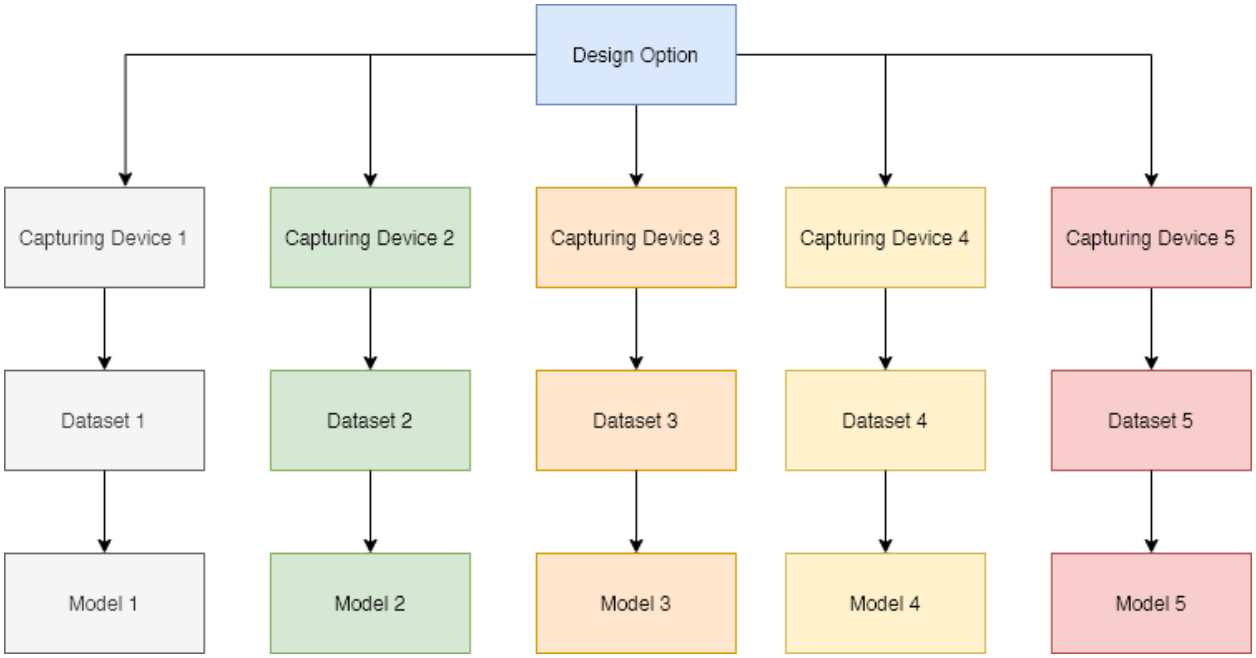


Figure 3.5 Process Flow of Training the Model

Figure 3.5 shows the process flow of training the model. There are three (3) design options that were tested using five (5) different capturing devices which resulted in five (5) models each. The capturing devices have different specifications, thus, the datasets collected have different qualities.

III. Capturing Devices

A series of tests to determine which capturing device specification as presented in table 3.1 can produce the best output for a specific algorithm were conducted. The testing resulted in five (5) models that were used for each design option. Also, these capturing devices were considered based on their compatibility with the processor. To lessen the cost, smartphones were used while taking into consideration the similarities of the megapixel between the capturing devices and phone cameras.

Table 3.1. Capturing devices with Specifications

Capturing Device	Equivalent Phone Model	Resolution	Sensor Pixel Size	Megapixel
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	1080	2.4 um	20
Arducam 13MP (IMX298)	Huawei Nova 2i	1080	1.12 um	13
MakerFocus RPI Camera Module	iPhone SE	1080	1.22µm	8
Arducam 16MP (IMX298)	Honor Play	1080	1.12 um	16
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	4096	0.8µm	64

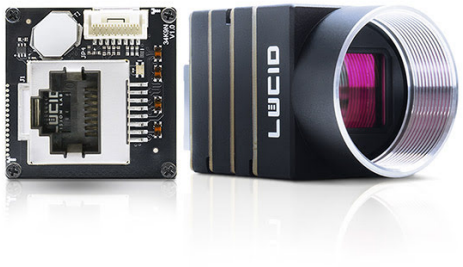


Figure 3.6 Lucid Vision Labs Phoenix 20MP Sony IMX 183 (IMX298)

Figure 3.7 Arducam 13MP

Figure 3.8 MakerFocus RPI Camera Module (IMX298)

Figure 3.9 Arducam 16MP

Figure 3.10 Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera

IV. Collection of Dataset

Figure 3.11 shows the process flow on how the dataset of the Eyebill system was developed. Five different datasets representing five (5) cameras, Lucid Vision Labs Phoenix 20MP, Arducam for Raspi, MakerFocus RPI Camera Module, Arducam 16MP, and Allied Vision Prosilica GT6400 APS-C to determine the best capturing device for the system were created.

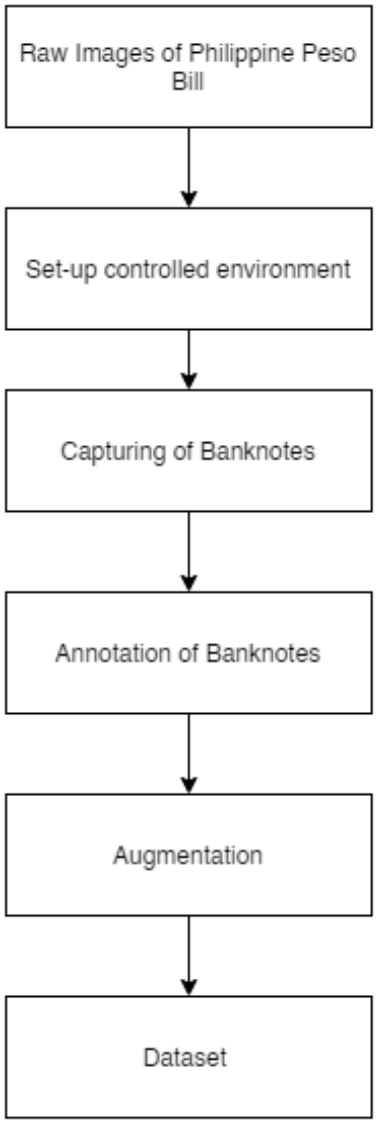
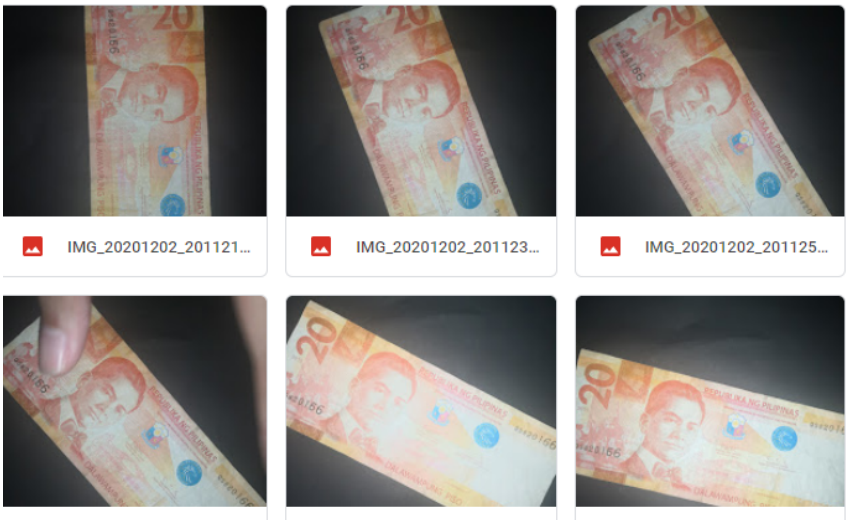


Figure 3.11 Process Flow for Data Gathering

1. Raw Images of Philippine Peso Bill - In preparation for data gathering, Philippine Paper bills (20, 50, 100, 200, 500, 1000) were collected. There are a total of 50 images for each banknote facing up and facing down which produces a total of 300 images.

2. Set-up Controlled Environment - a controlled environment is needed to replicate the environment inside the device. The set-up environment is as follows

- Distance from the bill: 6 inches
- Background color: black
- A total of 50 images for each banknote facing up and facing down
- Bill Orientation:



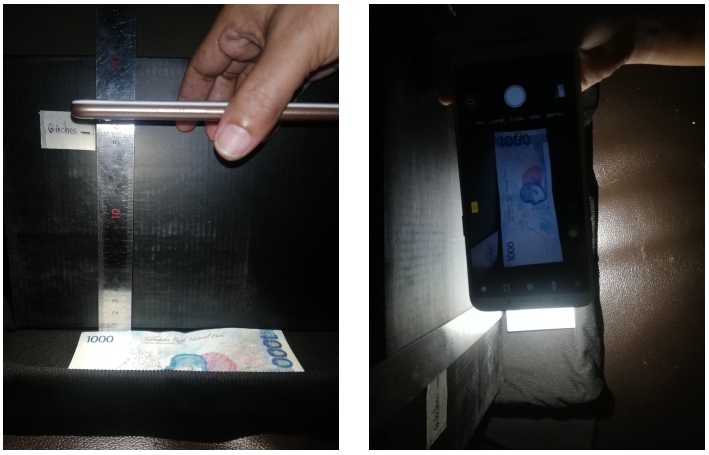


Figure 3.12 Controlled Environment

3. Capturing of Banknotes - After setting-up the environment, banknotes facing up and facing down were captured.

Figure 3.13 Annotation of Banknotes

4. Annotation of Banknotes - As shown in figure 3.13, the task of annotation or labeling data is needed. It is to associate information to the program elements in preparation for training a machine learning model.

5. Augmentation - This helps to significantly increase the diversity of data available for training models, without actually collecting new data. It can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

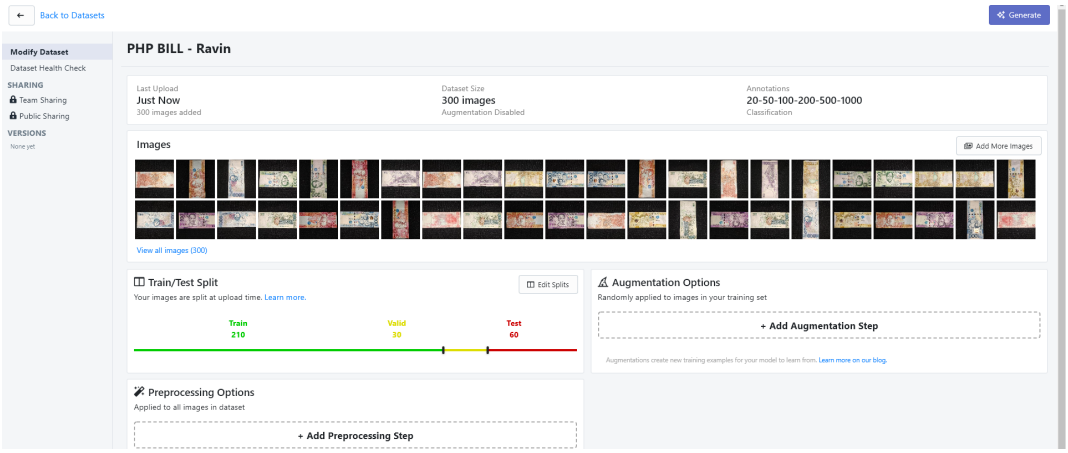


Figure 3.14 Number of Images Before and After Augmentation

Figure 3.14 above shows the number of images before the augmentation, which is a total of 300 images. After the augmentation, the dataset increases from 300 to 720 images. The techniques used include adding blur effects, changing brightness, and cropping each image.

F. Dataset - For each capturing device, a total of 300 images (50 images per denomination, front and back) were collected and underwent augmentation. The augmentation process produced 720 images from each capturing device, upon which are used for the proposed algorithms

V. Orb Algorithm for Counterfeit or Authentic Bill

There are two (2) algorithms which were considered for bill authentication, Kaze and Orb. ORB (Oriented FAST and rotated BRIEF) is a fast robust local feature matching first presented by Ethan Rublee et al. Feature matching is at the base of many computer vision problems, such as object recognition or structure from motion. The most important thing about the ORB is that it came from “OpenCV Labs”. An efficient alternative to SIFT or SURF in 2011. Kaze Features is a novel 2D feature detection and description method that operates completely in a nonlinear scale space [22]. However a study by Daniel R. Roosab1, Elcio H. Shiguemori b and Ana Carolina Lorena [23] states that ORB is faster to compute than AKAZE. ORB is also desirable in real-time applications. The study served as a basis to use the ORB algorithm to detect counterfeit or authentic detection.

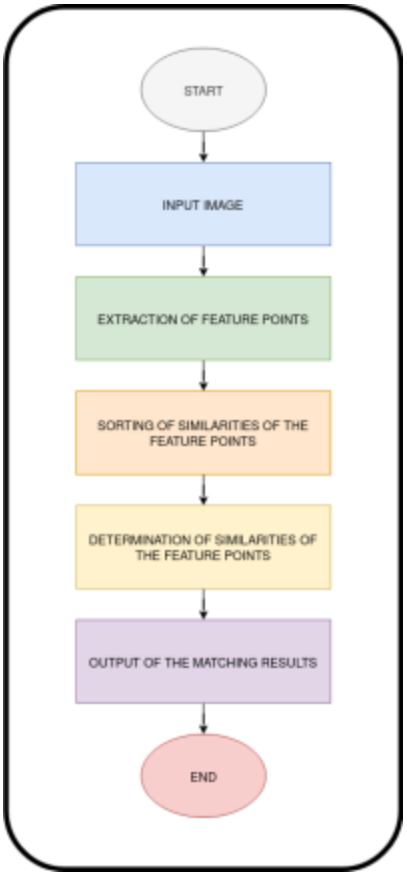


Figure 3.15 Orb Algorithm Flowchart

Figure 3.15 shows the flowchart of the ORB Algorithm which was used to identify authentic or counterfeit Philippine Peso Bill. The process starts upon inserting the Philippine Peso Bill into the platform, the ORB algorithm then starts the extraction of feature points wherein it recognizes the bills' unique key points. After the extraction, the ORB algorithm sorts the similarities of the former feature and points to a new inserted bill where it detects the similarities between the two. If the bill is fake, the platform warns the user, otherwise, it announces the bill denomination.

Figure 3.16 Authentic Philippine Peso Bill under UV Light

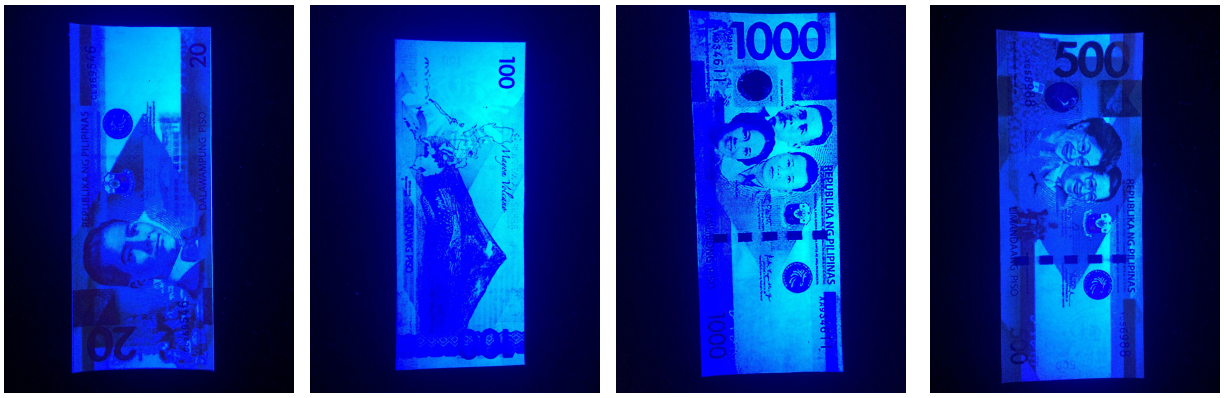


Figure 3.17 Counterfeit Philippine Peso Bill under UV Light

Figures 3.16 and 3.17 shows the comparison between a counterfeit Philippine Peso Bill and the authentic Philippine Peso bill under a UV Light. By doing this comparison, the following points are taken:

- Red and blue fibers embedded in the banknote paper glow under ultraviolet light on the authentic Philippine Peso Bill.
- Compared to a counterfeit bill, the authentic Philippine Peso Bill includes:
    - embossed prints
    - serial number (in variable-sized figures)
    - security fibers
    - Watermark
    - see-through mark ("Pilipino" spelled in Baybayin)
    - concealed value
    - security thread
    - optically variable device (only on 500 and 1000 peso notes)

Testing for Counterfeit or Authentic using ORB Algorithm

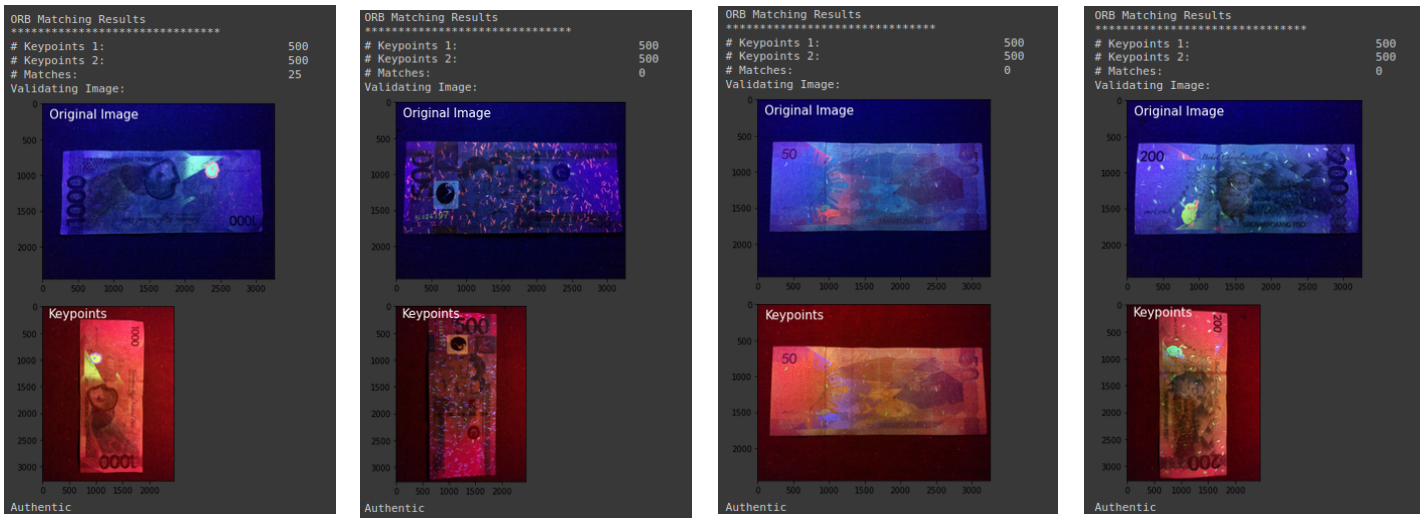


Figure 3.18 Baseline for ORB Algorithm

Figure 3.18 shows the images that serve as the basis of the ORB Algorithm to verify if the input Philippine Peso Bill is authentic or counterfeit.

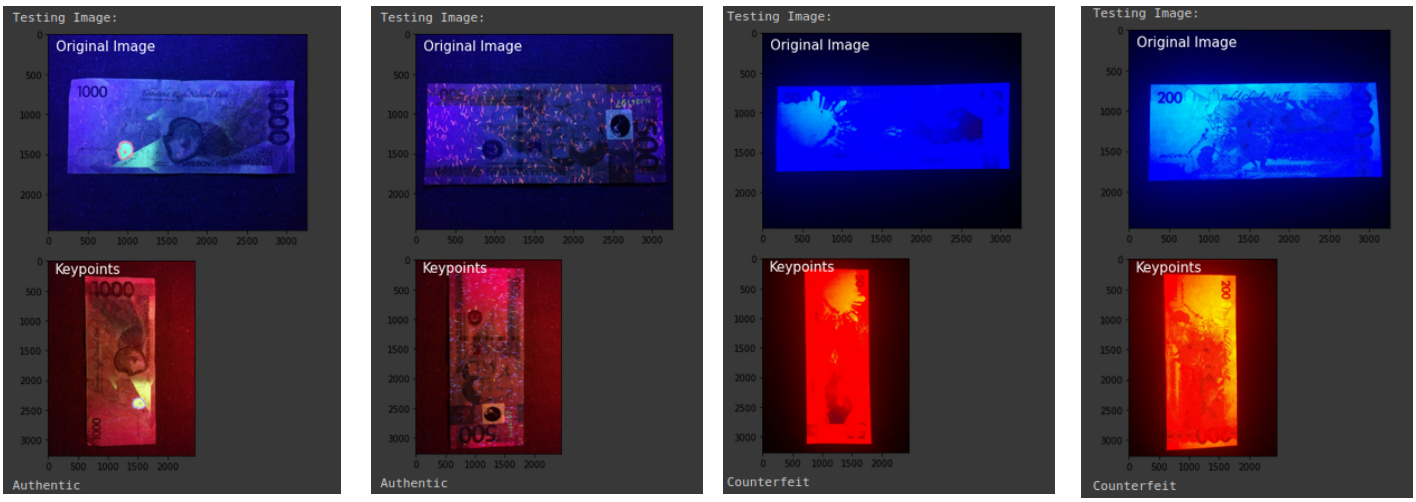


Figure 3.19 Output of Inserted PH Peso Bill

Figure 3.19 shows the output of the inserted Philippine Bill whether it is counterfeit or authentic.

Table 3.2 Testing Result for Counterfeit or Authentic

TP	445
TN	240
FP	35
FN	0
Accuracy	**95.14%**

As shown in Table 3.2, the accuracy of Counterfeit or Authentic testing is 95.14%. The accuracy was calculated in binary classifications in terms of positive and negatives, the formula is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where,

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

True Positive (TP) can be defined as an outcome where the algorithm correctly predicts the positive class. True Negative (TN) is an outcome where the algorithm correctly predicts the negative class. A False Positive (FP) is an outcome where the algorithm incorrectly predicts the positive class. And a False Negative (FN) is an outcome where the model incorrectly predicts the negative class.

VI. Bill of Material for Each Capturing Device

This section includes the Bill of Materials for all the capturing devices used in each design option. This includes all raw parts of the system, including the camera, processor, casing, speaker, battery, UV, and LED as well as its cost, type, and quantity. Each table varies depending on the capturing devices mentioned in table 3.3.

Table 3.3 BOM of Lucid Vision Labs Phoenix 20MP as its Capturing Device

Description	Qty	Type	Cost
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	1	Camera Module	21,300 PHP
Raspberry Pi 4 Model B	1	Processor	3,249.75 PHP
Plastic Casing	1	Casing	700 PHP
Button	1	Button	10 PHP
Mini Oval Speaker	1	Speaker	264.33 PHP
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	1	Amplifier	430 PHP
PiSugar2 Pro Battery	1	Battery	2402.49 PHP
Ultraviolet Light	1	UV	150 PHP
LED Light	1	LED	3.00 PHP
TOTAL:			**28,509.57 PHP**

The results in table 3.3 shown above signify the bill of materials with a camera cost of 21,300PHP, processor cost of 3,249.75PHP, casing cost of 700PHP, button cost of 10PHP, speaker cost of 264.33PHP, battery cost of 2402.49 PHP, UV cost of 150PHP and LED light cost of 3.00PHP with a total of 28,509.57PHP.

Table 3.4 BOM of Raspberry Pi 4 as its Capturing Device

Description	Qty	Type	Cost
Arducam for Raspi (IMX298)	1	Camera Module	2,400 PHP
Raspberry Pi 4 Model B	1	Processor	3,249.75 PHP
Plastic Casing	1	Casing	700 PHP
Button	1	Button	10 PHP
Mini Oval Speaker	1	Speaker	264.33 PHP
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	1	Amplifier	430 PHP
PiSugar2 Pro Battery	1	Battery	2402.49 PHP
Ultraviolet Light	1	UV	150 PHP
LED Light	1	LED	3.00 PHP
TOTAL:			**9,609.57 PHP**

The results in table 3.4 shown above signify the bill of materials with a camera cost of 2,400PHP, processor cost of 3,249.75PHP, casing cost of 700PHP, Button cost of 10PHP, speaker cost of 264.33PHP, battery cost of 2402.49PHP, UV cost of 150PHP and LED light cost of 3.00PHP with a total of 9,179.57PHP.

Table 3.5 BOM of MakerFocus RPI as its Capturing Device

Description	Qty	Type	Cost
MakerFocus RPI Camera Module	1	Camera Module	1,600 PHP
Raspberry Pi 4 Model B	1	Processor	3,249.75 PHP
Plastic Casing	1	Casing	700 PHP
Button	1	Button	10 PHP
Mini Oval Speaker	1	Speaker	264.33 PHP
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	1	Amplifier	430 PHP
PiSugar2 Pro Battery	1	Battery	2402.49 PHP
Ultraviolet Light	1	UV	150 PHP
LED Light	1	LED	3.00 PHP
TOTAL:			**8,809.57 PHP**

The results in table 3.5 shown above signify the bill of materials with a camera cost of 1,600PHP, processor cost of 3,249.75PHP, casing cost of 700PHP, Button cost of 10PHP, speaker cost of 264.33PHP, battery cost of 2402.49PHP, UV cost of 150PHP and LED light cost of 3.00PHP with a total of 8,379.57PHP.

Table 3.6 BOM of Arducam 16MP as its Capturing Device

Description	Qty	Type	Cost
Arducam 16MP (IMX298)	1	Camera Module	2,900 PHP
Raspberry Pi 4 Model B	1	Processor	3,249.75 PHP
Plastic Casing	1	Casing	700 PHP
Button	1	Button	10 PHP
Mini Oval Speaker	1	Speaker	264.33 PHP
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	1	Amplifier	430 PHP
PiSugar2 Pro Battery	1	Battery	2402.49 PHP
Ultraviolet Light	1	UV	150 PHP
LED Light	1	LED	3.00 PHP
TOTAL:			**10,109.57 PHP**

The results in table 3.6 shown above signify the bill of materials with a camera cost of 2,900PHP, processor cost of 3,249.75PHP, casing cost of 700PHP, Button cost of 10PHP, speaker cost of 264.33PHP, battery cost of 2402.49PHP, UV cost of 150PHP and LED light cost of 3.00PHP with a total of 9,679.57PHP.

Table 3.7 BOM of Allied Vision CMOS Camera as its Capturing Device

Description	Qty	Type	Cost
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	1	Camera Module	35,000 PHP
Raspberry Pi 4 Model B	1	Processor	3,249.75 PHP
Plastic Casing	1	Casing	700 PHP
Button	1	Button	10 PHP
Mini Oval Speaker	1	Speaker	264.33 PHP
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	1	Amplifier	430 PHP
PiSugar2 Pro Battery	1	Battery	2402.49 PHP
Ultraviolet Light	1	UV	150 PHP
LED Light	1	LED	3.00 PHP
TOTAL:			**42,209.57 PHP**

The results in table 3.7 shown above signify the bill of materials with a camera cost of 35,000PHP, processor cost of 3,249.75PHP, casing cost of 700PHP, Button cost of 10PHP, speaker cost of 264.33PHP, battery cost of 2402.49PHP, UV cost of 150PHP and LED light cost of 3.00PHP with a total of 42,209.57PHP.

VII. Prototype Layout

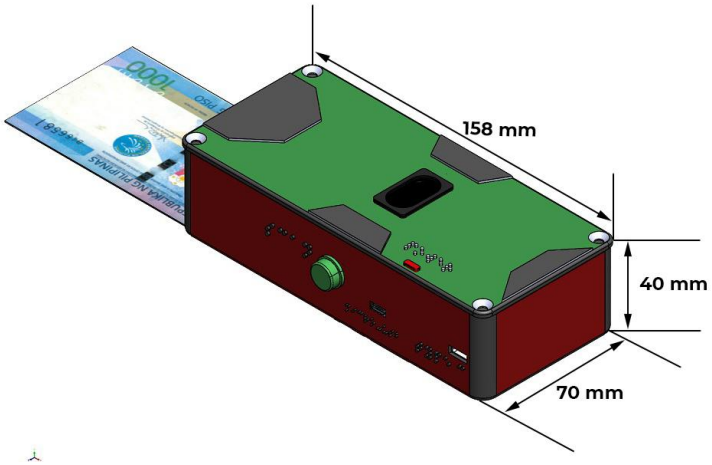


Figure 3.20 Prototype Layout using Solidworks

Figure 3.20 shows the prototype layout of the platform using Solidworks. The width of the platform is 70mm with a height of 40mm and a length of 158mm. The prototype is only an ideal design considering the objectives of the project, however the environment of the prototype was replicated for the testing of the design options.

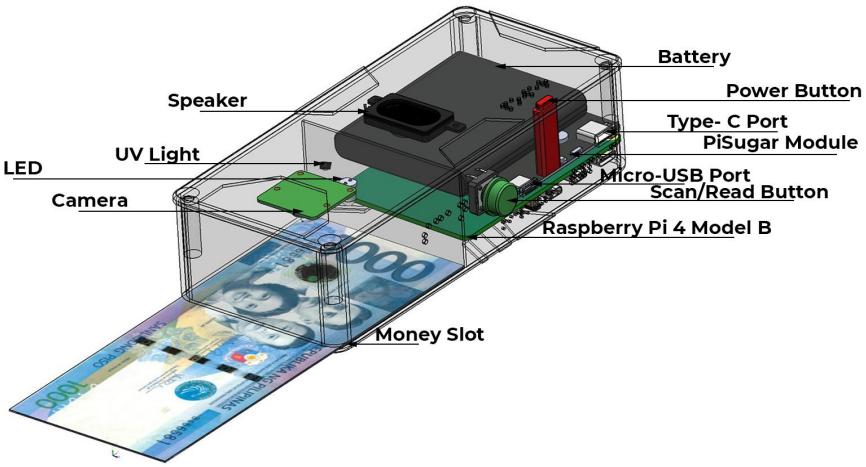


Figure 3.21 Prototype Layout using Solidworks

Figure 3.21 shows the prototype layout of the platform using Solidworks; the system consists of a battery, speaker, UV, arducam, power button, charging port, PiSugar2 Pro module, and the Raspberry Pi;

below are the other parts to complete the implementation and assembly of the project. The prototype is only an ideal design considering the objectives of the project.

- LED - emits light inside the device when an electric current is passed through it.
- Camera - an optical instrument used to capture images inside the bill.
- UV Light - fluorescent tubes with a dark blue or purple filter to remove other parts of the spectrum and leave just UVA light used in bill authentication.
- Speaker - output hardware device that connects to the Raspi to generate the output currency.
- Money Slot - a narrow opening or groove used to insert the bill inside the device.
- Raspi 4 Model B - used as a processor for the Eyebill system.
- Micro-USB Port - used as a charging port for the Eyebill platform.
- Scan/Read Button - used to scan/read the currency inserted by the user inside the Eyebill platform.
- PiSugar Module - a power supply module for raspberry pi zero
- Type-C Port - a 24-pin USB connector system with a rotationally symmetrical connector.
- Power Button - powers the Eyebill platform, it powers on when the button is pressed and powers off when it's pressed again.
- Battery - converts chemical energy into electrical energy by a chemical reaction, used to power the Eyebill platform.

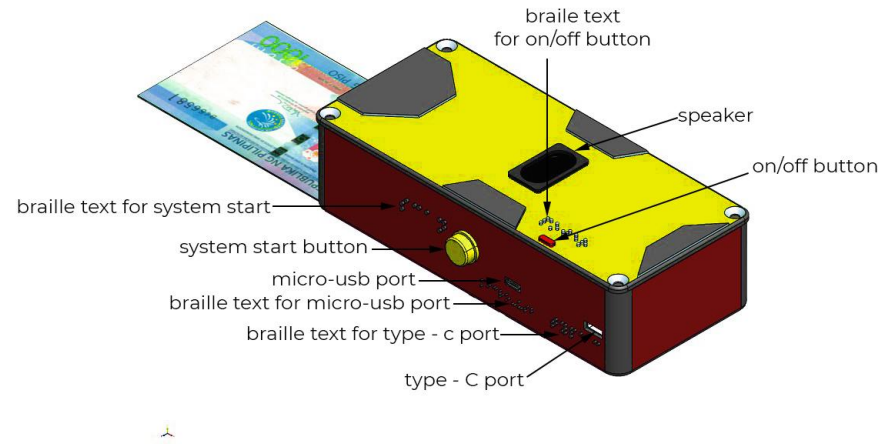


Figure 3.22 Eyebill Prototype Opaque

Figure 3.22 shows the prototype layout of the platform using Solidworks; the system consists of a battery, speaker, UV, arducam, power button, charging port, PiSugar2 Pro module, and the Raspberry Pi; below are the other parts to complete the implementation and assembly of the project.

- Micro-USB port - used as a charging port for the Eyebill platform.
- Type-C port - a 24-pin USB connector system with a rotationally symmetrical connector.
- Off/on button - powers the Eyebill platform, it powers on when the button is pressed and powers off when it's pressed again.
- Speaker - converts chemical energy into electrical energy by a chemical reaction, used to power the Eyebill platform.
- Braille text - used by the blind or visually impaired in reading a text, they are constructed from six dots. Braille dots are positioned like the figure six on a die, in a grid of two parallel vertical lines of three dots each.

Design Option 1: YOLOv4-darknet Algorithm

YOLO is short for You Only Look Once. It is a real-time object recognition system that can recognize multiple objects in a single frame. The real-time recognition system recognizes multiple objects from an image and also makes a boundary box around the object. It can be easily trained and deployed in a production system. [24]

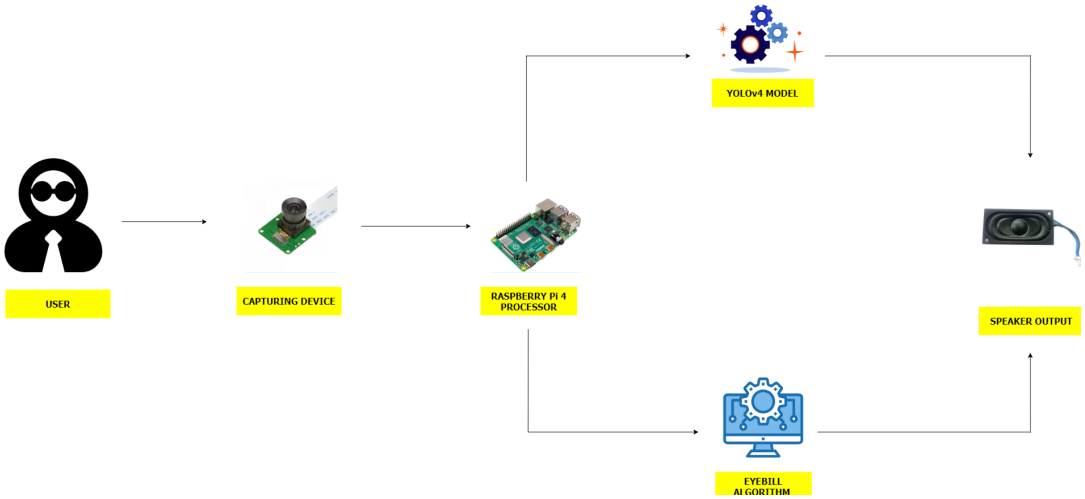


Figure 3.23 System Architecture using YOLOv4 Algorithm

Figure 3.23 shows the Eyebill system using the YOLOv4 algorithm, upon inserting the Philippine Peso bill, the platform uses its capturing device to photograph an image of the bill. The captured image is fed into the processor, which is the central part of the system. The processed image is used by the model where it identifies the denomination through the YOLOv4 algorithm. After the identification, the platform produces its findings and produces a sound using a speaker indicating the currency of the bill.

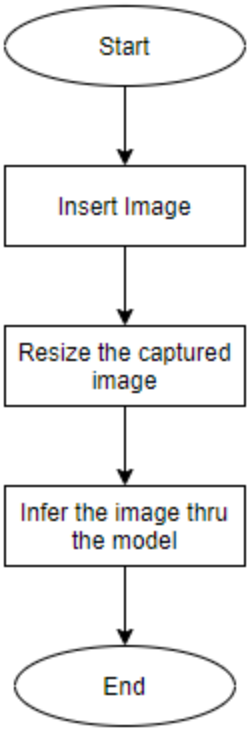


Figure 3.24 YOLOv4 flowchart

Figure 3.24 represents the Eyebill YOLOv4 flowchart; it is a step-by-step process in sequential order. Flowcharts use special shapes to represent different types of actions or steps in a process. The oval represents the start or endpoint; the rectangles represent the process; the diamond indicates the decision. The explanation of each process is as follows:

a. **Insert bill and capture images -** To start the system, the user inserts the Philippine Peso Bill into the platform. During this process, the system captures the bill.
b. **Resize the images -** After capturing, the system then resizes the bill to 416 x 416 to match the model trained; this helps to yield better results.
c. **Infer images into the model -** In this process, the system starts to predict a confidence score to give users an accurate output.

Testing of Models using Yolov4

Five (5) capturing devices for Design Option 1 were used, thus creating five (5) datasets. Each dataset was then split into two (2), 80% for the training and 20% for testing. The training dataset was used to create five (5) models using the Yolov4 algorithm. Meanwhile, the testing dataset will be used later on to check the quality and performance of the algorithm. It has an equal distribution of the Philippine Peso bill (having 10 images for each denomination) with a total of 50 images per capturing device. Each model was tested using all the testing dataset to avoid bias.

The prediction result signifies the testing, the 'true' signifies the correct prediction while the 'false' signifies the incorrect bill prediction. The training and testing was conducted by using the same number of input data for each Philippine Peso Bill, the same environment, and the same procedures to guarantee fairness in getting the results and accuracy. Accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

To compute the cost of the system, a Bill of Materials (BOM) (*see section IV. Bill of Material for Capturing Device*) was created, which is a comprehensive inventory of the raw materials of each capturing device based on the raw parts of the system, including the camera, processor, casing, speaker, battery, UV, and LED. The total cost can be defined as:

$$Cost = \sum cost\ of\ components\ (PHP)$$

The speed of each model was computed based on starting time - output time as shown in the code below:

```
PATH1
*/content/drive/MyDrive/PD2/TEST/albert test/
test img= [f for f in os. listdir (PATH1) if f.endewith('.jpg') ]
for x in test_img:
start = time. time ()
predict1 (PATH1 + x)
end = time. time 0)
print ("Time predicted = " +("30.7€" & (((end-start)/10) *1000)) + "ms")
```

a. Model Used: iPhone SE

The model used for this testing was trained using the iPhone SE training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.8 YOLOv4 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	60	0
Xiaomi Redmi k30 Pro	60	60	0
Huawei Nova 2i	60	59	1
Honor Play	60	38	22
Honor 8X	60	60	0

Table 3.8 signifies the result of the testing conducted to verify if the model works as it is designed. There are a total of 300 tests conducted for this model. The iPhone SE, Xiaomi Redmi k30 Pro, and Honor 8x got 60 correct predictions, while Huawei Nova 2i and Honor Play got 59 and 38 correct predictions, respectively.

Table 3.9 YOLOv4 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED (ms)	COST
MakerFocus RPI Camera Module	iPhone SE	100%	47.56	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	100%	47.20	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	98.33%	47.75	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	63.33%	47.62	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	47.82	28,509.57 PHP

Table 3.9 shows the result of the testing conducted, including the constraints of the system. iPhone SE, Honor 8X, and Xiaomi Redmi k30 Pro got an accuracy of 100% while Huawei Nova 2i and Honor Play got 98.33% and 63.33% consecutively. In terms of speed, the devices used are almost identical at 47 ms, while the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

b. Model Used: Xiaomi Redmi K30 Pro

The model used for this testing was trained using the Xiaomi Redmi K30 Pro training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.10 YOLOv4 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	60	0
Xiaomi Redmi k30 Pro	60	60	0
Huawei Nova 2i	60	60	0
Honor Play	60	43	17
Honor 8X	60	60	0

The results in table 3.10 signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used. The iPhone SE, Xiaomi Redmi k30 Pro, Honor 8x, and Huawei Nova 2i got 60 correct predictions while Honor Play got 43 correct predictions.

Table 3.11 YOLOv4 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	100%	35.15 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	100%	35.78 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	100%	36.87 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	71.67%	35.12 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100,%	36.16 ms	28,509.57 PHP

Table 3.10 shows the result of the testing conducted, including the constraints of the system. iPhone SE, Xiaomi Redmi k30 Pro, Huawei Nova 2i, and Honor 8x got an accuracy of 100%, while Honor Play got 71.67%. In terms of speed, the Honor Play got the fastest result with 35.12ms, and Huawei Nova 2i got the slowest with 36.87ms. The MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

c. Model Used: Huawei Nova 2i

The model used for this testing was trained using the Huawei Nova 2i training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.12 YOLOv4 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	60	0
Xiaomi Redmi k30 Pro	60	60	0
Huawei Nova 2i	60	59	1
Honor Play	60	46	16
Honor 8X	60	60	0

The results in table 3.12 signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and the iPhone SE, Xiaomi Redmi k30 Pro, and Honor 8x got 60 correct predictions while Huawei Nova 2i and Honor Play got 59 and 46 correct predictions, respectively.

Table 3.13 YOLOv4 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	100%	47.52 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	100%	47.15 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	98.33%	47.84 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	76.67%	47.20 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	47.64 ms	28,509.57 PHP

Table 3.13 shows the result of the testing conducted including the constraints of the system. iPhone SE, Xiaomi Redmi k30 Pro, and Honor 8X got an accuracy of 100% while Huawei Nova 2i and Honor Play got 98.33% and 76.67% consecutively. In terms of speed, the devices used are almost identical at 47 ms, while the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

d. Model Used: Honor Play

The model used for this testing was trained using the Honor Play training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.14 YOLOv4 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	54	6
Xiaomi Redmi k30 Pro	60	59	1
Huawei Nova 2i	60	60	0
Honor Play	60	60	0
Honor 8X	60	60	0

The results in table 3.14 signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and Huawei Nova 2i, Honor Play, and Honor 8X got 60 correct predictions while iPhone SE and Xiaomi Redmi k30 Pro got 54 and 59 correct predictions, respectively.

Table 3.15 YOLOv4 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	90%	44.87 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	98.33%	44.25 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	100%	44.87 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	100%	44.64 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	44.55 ms	28,509.57 PHP

Table 3.15 shows the result of the testing conducted, including the constraints of the system. Huawei Nova 2i, Honor Play, and Honor 8X got an accuracy of 100%, while iPhone SE and Xiaomi Redmi k30 Pro got 90% and 98.33% consecutively. In terms of speed, the devices used are almost identical with 44 ms, while the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

e. Model Used: Honor 8X

The model used for this testing was trained using the Honor 8X training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.16 YOLOv4 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	28	32
Xiaomi Redmi k30 Pro	60	33	27
Huawei Nova 2i	60	33	27
Honor Play	60	30	30
Honor 8X	60	60	0

The results in table 3.16 signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used and only Honor 8x got 60 correct predictions while iPhone SE got 28, Honor Play got 30, and both Xiaomi Redmi k30 Pro and Huawei Nova 2i got 33 correct predictions.

Table 3.17 YOLOv4 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	46.67%	47.94 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	55%	47.24 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	55%	47.89 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	50%	47.77 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	47.35 ms	28,509.57 PHP

Table 3.17 shows the result of the testing conducted, including the constraints of the system. Only Honor 8X got an accuracy of 100% while Honor Play got 50%; meanwhile, both Xiaomi Redmi k30 Pro and Huawei Nova 2i got 55% accuracy. In terms of speed, the devices used are almost identical at 47 ms, while the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

YOLOv4 Overall Test Result:

Table 3.18 YOLOv4 Average Result

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	92.33%	47.59 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	94.33%	35.82 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	95.20%	47.47 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	97.67%	44.64 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	61.33%	47.64 ms	28,509.57 PHP

Table 3.18 shows the overall result of the testing conducted, including the constraints of the system. Honor Play got the highest accuracy of 97.67%, while the Honor 8X got the lowest with 61.33%. In terms of speed, the Xiaomi Redmi k30 Pro is the fastest with 35.82 ms, and the Honor 8X got the slowest with 47.64ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

YOLOv4 Testing

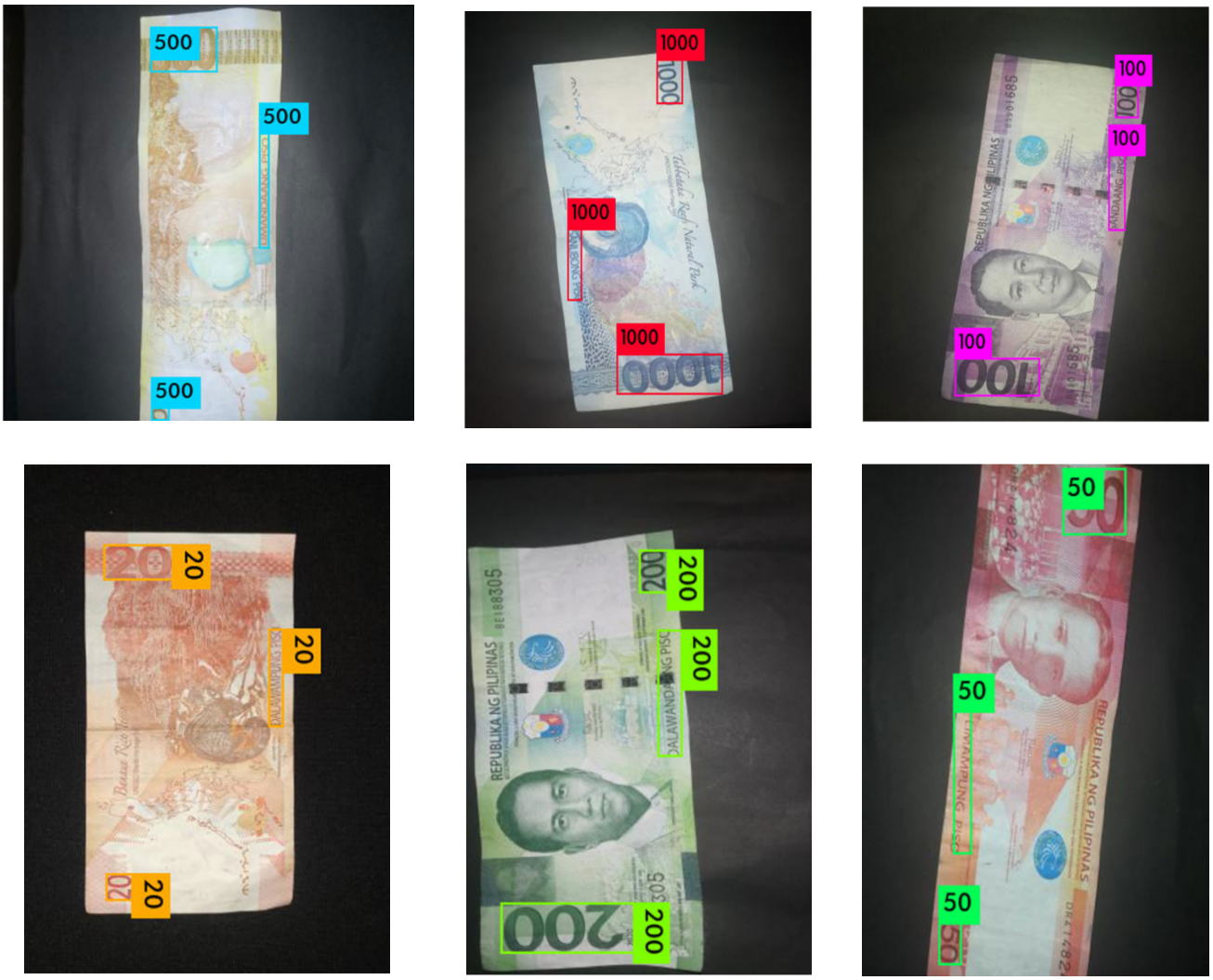


Figure 3.25 Eyebill System using YOLOv4

Figure 3.25 shows the actual testing using the YOLOv4, there are a total of 60 tests conducted per capturing device and as shown in table 3.12, Honor Play got the highest accuracy of 97.67% and Xiaomi Redmi k30 Pro is the fastest with 35.82 ms. The creation of the dataset for this algorithm is done through a controlled environment which has the standard as seen in Figure 3.14.

Design Option 2: YOLOv5 - PyTorch Algorithm

YOLOv5 - PyTorch is a recent release of the YOLO family of models. YOLO was initially introduced as the first object detection model that combined bounding box prediction and object classification into a single end to end differentiable network. It was written and is maintained in a framework called Darknet. YOLOv5 is the first of the YOLO models to be written in the PyTorch framework and it is much more lightweight and easy to use. That said, YOLOv5 did not make major architectural changes to the network in YOLOv4 and does not outperform YOLOv4 on a common benchmark, the COCO dataset [25]

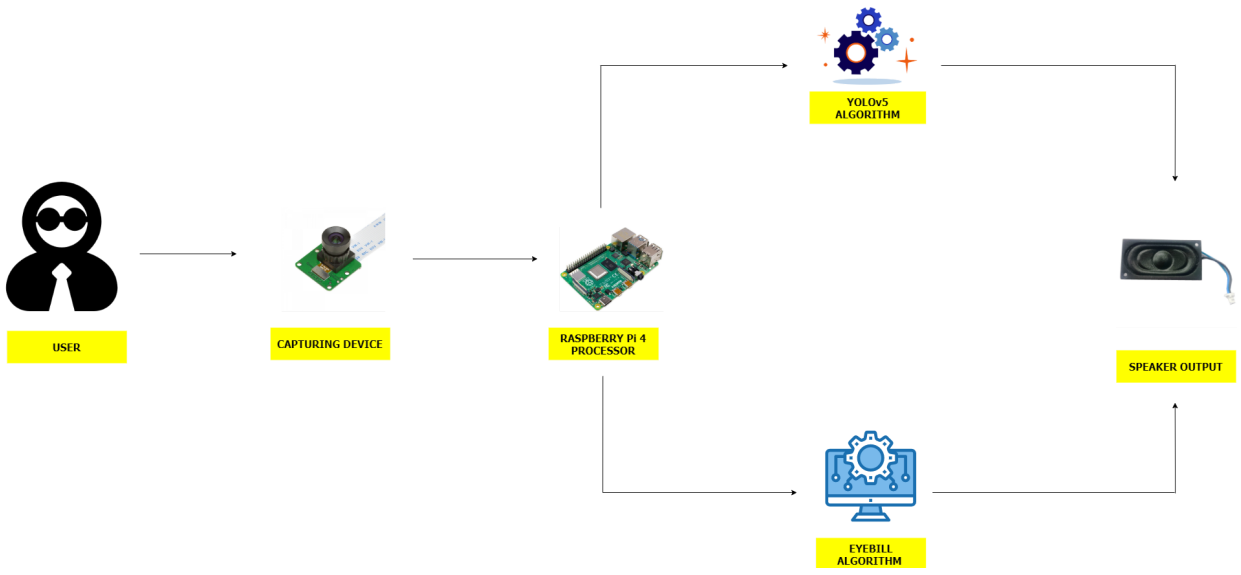


Figure 3.26 System Architecture using YOLOv5 Algorithm

Figure 3.26 shows the Eyebill system using the YOLOv5 algorithm; upon inserting the Philippine Peso bill, the platform uses its capturing platform to photograph an image of the bill. The captured image is fed into the processor, which is the central part of the system. The processed image is used by the model where it identifies the denomination through the YOLOv5 algorithm. After the identification, the platform produces its findings and produces a sound using a speaker indicating the currency of the bill.

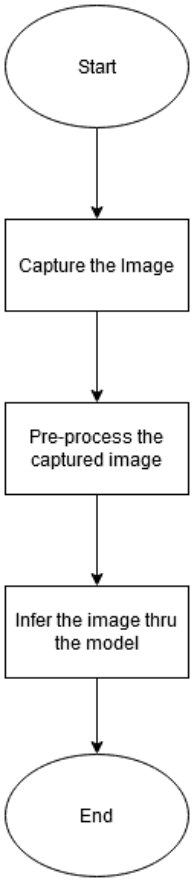


Figure 3.27 YOLOv5 Algorithm Flowchart

Figure 3.27 represents the Eyebill YOLOv5 flowchart; it is the separate steps of a process in sequential order. Flowcharts use special shapes to represent different types of actions or steps in a process. The oval represents the start or endpoint; the rectangles represent the process; the diamond indicates the decision.

A. Capture the image - To start the system, the user inserts the Philippine Peso Bill into the platform. During this process, the system captures the bill.

B. Pre-process the captured image - The system then starts processing the captured image using the YOLOv5 algorithm; it then automatically resizes the image to fit the algorithm.

C. Infer the image thru the model - Finally, the system will start to predict the amount of the inserted bill and produce an output indicating its amount.

Testing of Models using Yolov5

Five (5) capturing devices for Design Option 2 were tested. There are a total of sixty (60) tests conducted for each capturing device, this includes verifying if the algorithm can identify the denomination of a specific Philippine Peso Bill. The prediction result signifies the testing, the 'true' signifies the correct prediction while the 'false' signifies the incorrect bill prediction. The training and testing was conducted by using the same number of input data for each Philippine Peso Bill, the same environment, and the same procedures to guarantee fairness in getting the results and accuracy. The distribution of the Philippine Peso Bill during the testing was equal (having 10 images for each Philippine Peso Bill Denomination). Accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

To compute the cost of the system, a Bill of Materials (BOM), which is a comprehensive inventory of the raw materials of each capturing device based on the raw parts of the system, including the camera, processor, casing, speaker, battery, UV, and LED was created. The total cost can be defined as:

$$Cost = \sum cost\ of\ components\ (PHP)$$

The speed of each model was computed based on starting time - output time as shown in the code below:

```
PATH1
*/content/drive/MyDrive/PD2/TEST/albert test/
test img= [f for f in os. listdir (PATH1) if f.endewith('.jpg') ]
for x in test_img:
start = time. time ()
predict1 (PATH1 + x)
end = time. time 0)
print ("Time predicted = " +("30.7€" & (((end-start)/10) *1000)) + "ms")
```

a. Model Used: iPhone SE

The model used for this testing was trained using the iPhone SE training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.19 YOLOv5 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	59	1
Xiaomi Redmi k30 Pro	60	52	8
Huawei Nova 2i	60	56	4
Honor Play	60	20	40
Honor 8X	60	53	7

The results in table 3.19 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and the iPhone got the highest number with 59 correct predictions while Honor Play got the lowest with only 20 correct predictions.

Table 3.20 YOLOv5 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	98.33%	11.45	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	86.67%	12.14	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	93.33%	11.16	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	33.33%	14.79	10,109.57 PHP
Phoenix 20MP, Sony IMX 183	Honor 8X	88.33%	13.47	28,509.57 PHP

Table 3.20 shows the result of the testing conducted, including the constraints of the system. The iPhone Se got the highest accuracy with 98.33%, while Honor Play got the lowest with 33.33%. In terms of speed, the Huawei Nova 2i yielded the fastest result with 11.16 ms, and Honor Play got the slowest with 14.79 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

b. Model Used: Xiaomi Redmi K30 Pro

The model used for this testing was trained using the Xiaomi Redmi K30 Pro training dataset. The researchers tested this model using all five testing datasets to avoid bias results

Table 3.21 YOLOv5 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	43	17
Xiaomi Redmi k30 Pro	60	54	6
Huawei Nova 2i	60	55	5
Honor Play	60	21	39
Honor 8X	60	43	17

The results in table 3.21 shown above signify the result of the testing conducted to verify if the model works as it is designed. There were a total of 60 tests conducted for all the devices used, Huawei Nova 2i got the highest number with 55 correct predictions while Honor Play got the lowest with only 21 correct predictions.

Table 3.22 YOLOv5 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	71.67%	11.46	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	90%	11.14	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	91.67%	13.64	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	35%	14.19	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	71.67%	14.11	28,509.57 PHP

Table 3.22 shows the result of the testing conducted, including the constraints of the system. Huawei Nova 2 got the highest accuracy with 91.67%, while Honor Play got the lowest with 35%. In terms of speed, the Xiaomi Redmi k30 Pro yielded the fastest result with 11.14 ms, and Honor Play got the slowest with 14.19 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

c. Model Used: Huawei Nova 2i

The model used for this testing was trained using the Huawei Nova 2i training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.23 YOLOv5 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	56	4
Xiaomi Redmi k30 Pro	60	49	11
Huawei Nova 2i	60	60	0
Honor Play	60	38	22
Honor 8X	60	59	1

The results in table 3.23 shown above signify the result of the testing conducted to verify if the model works as it is designed. There were a total of 60 tests conducted for all the devices used, Huawei Nova 2i got the highest number with 60 correct predictions while Honor Play got the lowest with only 38 correct predictions.

Table 3.24 YOLOv5 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	93.33%	11.64	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	81.67%	12.64	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	100%	11.16	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	63.33%	14.79	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	98.33%	13.65	28,509.57 PHP

Table 3.24 shows the result of the testing conducted, including the constraints of the system. The Huawei Nova 2i and iPhone SE got the highest accuracy with 100%, while Honor Play got the lowest with 63.33%. In terms of speed, the Huawei Nova 2i and iPhone SE yielded the fastest result with 11 ms, and Honor Playi got the slowest with 14.79 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

d. Model Used: Honor Play

The model used for this testing was trained using the Honor Play training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.25 YOLOv5 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	48	12
Xiaomi Redmi k30 Pro	60	50	10
Huawei Nova 2i	60	55	5
Honor Play	60	54	6
Honor 8X	60	52	8

The results in table 3.25 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and Huawei Nova 2i got the highest number with 55 correct predictions while iPhone SE got the lowest with only 48 correct predictions.

Table 3.26 YOLOv5 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	80%	15.64	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	83.33%	11.57	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	91.67%	12.16	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	90%	11.64	10,109.57 PHP
Phoenix 20MP, Sony IMX 183	Honor 8X	86.67%	14.67	28,509.57 PHP

Table 3.26 shows the result of the testing conducted, including the constraints of the system. The Huawei Nova 2i got the highest accuracy with 91.67%, while iPhone SE got the lowest with 80%. In terms of speed, the Xiaomi Redmi k30 Pro yielded the fastest result with 11.57 ms, and iPhone SE got the slowest with 15.64 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

e. Model Used: Honor 8x

The model used for this testing was trained using the Honor 8X training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.27 YOLOv5 Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	23	37
Xiaomi Redmi k30 Pro	60	26	34
Huawei Nova 2i	60	50	10
Honor Play	60	25	35
Honor 8X	60	50	10

The results in table 3.27 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and Honor 8x and Huawei Nova 2i got the highest number with 50 correct predictions while iPhone SE got the lowest with only 23 correct predictions.

Table 3.28 YOLOv5 Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	38.33%	14.61	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	43.33%	13.26	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	83.33%	11.78	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	41.67%	12.64	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	83.33%	11.14	28,509.57 PHP

Table 3.28 shows the result of the testing conducted, including the constraints of the system. The Huawei Nova 2i and Honor 8X got the highest accuracy with 83.33%, while iPhone SE got the lowest with 38.33%. In terms of speed, the Honor 8xi yielded the fastest result with 11.14 ms, and iPhone SE got the slowest with 14.61 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

YOLOv5 Overall Test Result:

Table 3.29 YOLOv5 Average Result

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	80%	12.60 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	72%	12.91 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	87.33%	12.78 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	86.33%	13.14 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	58%	12.69 ms	28,509.57 PHP

Table 3.29 shows the overall result of the testing conducted, including the constraints of the system. Huawei Nova 2i got the highest accuracy of 87.33%, while the Honor 8X got the lowest with 58%. In terms of speed, the iPhone SE is the fastest with 12.60 ms, and the Honor Play got the slowest with 13.14 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

YOLOv5 Testing

Figure 3.28 Eyebill System using YOLOv5

Figure 3.28 shows the actual testing using the YOLOv5, there are a total of 60 tests conducted per capturing device and as shown in table 3.32, Huawei Nova 2i got the highest accuracy of 87.33% and iPhone SE is the fastest with 12.60 ms. The creation of the dataset for this algorithm is done through a controlled environment which has the standard as seen in Figure 3.14.

Design Option 3: EfficientDet Algorithm

EfficientDet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a *compound coefficient*. Unlike conventional practice that arbitrarily scales these factors, the EfficientDet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients. The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image. The base EfficientDet-B0 network is based on the inverted bottleneck residual blocks of MobileNetV2, in addition to squeeze-and-excitation blocks [26]

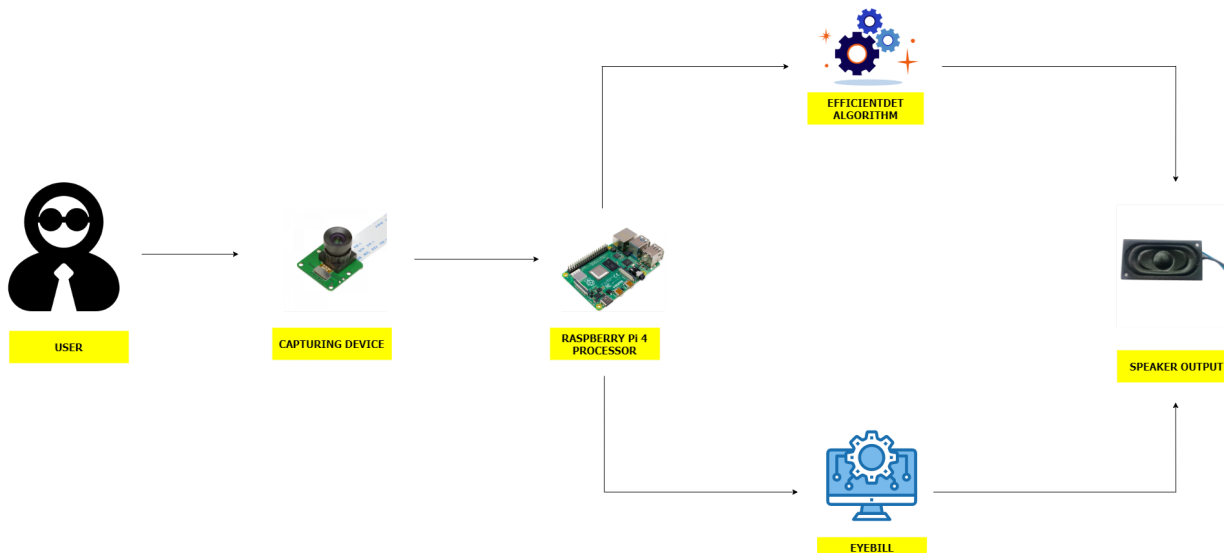


Figure 3.29 System Architecture using EfficientDet Algorithm

Figure 3.29 shows the Eyebill system using the EfficientDet algorithm; upon inserting the Philippine Peso bill, the platform uses its capturing platform to photograph an image of the bill. The captured image is fed into the processor, which is the central part of the system. The processed image is used by the model where it identifies the denomination through the EfficientDet algorithm. After the identification, the platform produces its findings and produces a sound using a speaker indicating the currency of the bill.

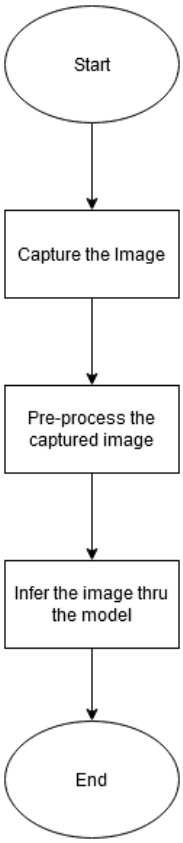


Figure 3.30 EfficientDet Algorithm Flowchart

Figure 3.30 represents the Eyebill EfficientDet flowchart; it is the separate steps of a process in sequential order. Flowcharts use special shapes to represent different types of actions or steps in a process. The oval represents the start or endpoint; the rectangles represent the process; the diamond indicates the decision.

A. Capture the image - To start the system, the user inserts the Philippine Peso Bill into the platform. During this process, the system captures the bill.

B. Pre-process the captured image - The system starts processing the captured image using the EfficientDet algorithm; it automatically resizes the image to fit the algorithm.

C. Infer the image thru the model - Finally, the system will start to predict the amount of the inserted bill and produce an output indicating its amount.

Testing of Models using EfficientDet

The proponents were able to test five (5) capturing devices for Design Option 1. There are a total of sixty (60) tests conducted for each capturing device, this includes verifying if the algorithm can identify the denomination of a specific Philippine Peso Bill. The prediction result signifies the testing, the 'true' signifies the correct prediction while the 'false' signifies the incorrect bill prediction. To measure the accuracy, each capturing device was tested and evaluated. The training and testing was conducted by using the same number of input data for each possible variable, the same environment, and the same procedures to guarantee fairness in getting the results and accuracy. The training and testing was conducted by using the same number of input data for each Philippine Peso Bill, the same environment, and the same procedures to guarantee fairness in getting the results and accuracy. The distribution of the Philippine Peso Bill during the testing was equal (having 10 images for each Philippine Peso Bill Denomination). Accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

To compute the cost of the system, a Bill of Materials (BOM), which is a comprehensive inventory of the raw materials of each capturing device based on the raw parts of the system, including the camera, processor, casing, speaker, battery, UV, and LED was created. The total cost can be defined as:

$$Cost = \sum cost\ of\ components\ (PHP)$$

The speed of each model was computed based on starting time - output time as shown in the code below:

```
PATH1
*/content/drive/MyDrive/PD2/TEST/albert test/
test img= [f for f in os. listdir (PATH1) if f.endewith('.jpg') ]
for x in test_img:
start = time. time ()
predict1 (PATH1 + x)
end = time. time 0)
print ("Time predicted = " +("30.7€" & (((end-start)/10) *1000)) + "ms")
```

a. Model Used: iPhone SE

The model used for this testing was trained using the iPhone SE training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.30 EfficientDet Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	60	0
Xiaomi Redmi k30 Pro	60	60	0
Huawei Nova 2i	60	59	1
Honor Play	60	32	28
Honor 8X	60	53	7

The results in table 3.30 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and iPhone SE and Xiaomi Redmi k30 Pro got the highest number with 60 correct predictions while Honor Play got the lowest with only 32 correct predictions.

Table 3.31 EfficientDet Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	100%	60.30 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	100%	61.77 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	98.33%	60.16 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	53.33%	59.76 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	83.33%	63.56. ms	28,509.57 PHP

Table 3.31 shows the result of the testing conducted, including the constraints of the system. The iPhone SE and Xiaomi Redmi k30 Pro got the highest accuracy with 100% while Honor Play got the lowest with 53.33%. In terms of speed, the Honor Play got the highest with 59.76 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

b. Model Used: Xiaomi Redmi K30 Pro

The model used for this testing was trained using the Xiaomi Redmi K30 Pro training dataset. The researchers tested this model using all five testing datasets to avoid bias results

Table 3.32 EfficientDet Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	60	0
Xiaomi Redmi k30 Pro	60	59	1
Huawei Nova 2i	60	60	0
Honor Play	60	32	28
Honor 8X	60	60	0

The results in table 3.32 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used and the iPhone SE, Huawei Nova 2i, Honor 8x got the highest number with 60 correct predictions while Honor Play got the lowest with only 32 correct predictions.

Table 3.33 EfficientDet Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	100%	63.89 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	98.33%	62.11 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	100%	59.23 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	53.33%	61.56 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	60.23 ms	28,509.57 PHP

Table 3.33 shows the result of the testing conducted including the constraints of the system. The iPhone SE, Huawei Nova 2i, and Honor 8x got the highest accuracy with 100% while Honor Play got the lowest with 53.33%. In terms of speed, the Huawei Nova 2i yielded the fastest result with 59.23 ms, and iPhone SE got the slowest with 63.89 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

c. Model Used: Huawei Nova 2i

The model used for this testing was trained using the Huawei Nova 2i training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.34 EfficientDet Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	60	0
Xiaomi Redmi k30 Pro	60	60	0
Huawei Nova 2i	60	60	0
Honor Play	60	42	18
Honor 8X	60	60	0

The results in table 3.34 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and Honor Play got the lowest with 42 incorrect predictions.

Table 3.35 EfficientDet Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	100%	59.10 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	100%	61.30 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	100%	58.30 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	70%	59.20 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	58.59 ms	28,509.57 PHP

Table 3.35 shows the result of the testing conducted, including the constraints of the system. The Honor Play got the lowest accuracy with 70%. In terms of speed, the Huawei Nova 2i yielded the fastest result with 58.30 ms, and Xiaomi Redmi k30 Pro got the slowest with 61.30 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

d. Model Used: Honor Play

The model used for this testing was trained using the Honor Play training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.36 EfficientDet Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	51	9
Xiaomi Redmi k30 Pro	60	57	3
Huawei Nova 2i	60	60	0
Honor Play	60	48	12
Honor 8X	60	60	0

The results in table 3.36 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used and Huawei Nova 2i and Honor 8x got the highest number with 60 correct predictions while Honor Play got the lowest with only 48 correct predictions.

Table 3.37 EfficientDet Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	85%	62.73 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	95%	63.16 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	100%	60.46 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	80%	59.16 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	100%	60.15 ms	28,509.57 PHP

Table 3.37 shows the result of the testing conducted, including the constraints of the system. The Huawei Nova 2i and Honor 8x got the highest accuracy with 100%, while Honor Play got the lowest with 80%. In terms of speed, almost all capturing devices have an identical speed of 59 to 60 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

e. Model Used: Honor 8x

The model used for this testing was trained using the Honor 8X training dataset. The researchers tested this model using all five testing datasets to avoid bias results.

Table 3.38 EfficientDet Test Result

DEVICE USED FOR THE DATASET	TOTAL NUMBER OF TESTS	PREDICTION RESULT	
		TRUE	FALSE
iPhone SE	60	13	47
Xiaomi Redmi k30 Pro	60	12	48
Huawei Nova 2i	60	30	30
Honor Play	60	60	0
Honor 8X	60	27	33

The results in table 3.38 shown above signify the result of the testing conducted to verify if the model works as it is designed. There are a total of 60 tests conducted for all the devices used, and Honor Play got the highest number with 60 correct predictions while Xiaomi Redmi k30 Pro got the lowest with only 12 correct predictions.

Table 3.39 EfficientDet Test Result with Constraints

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	21.67%	59.45 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	20%	61.30 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	50%	63.49 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	100%	62.16 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	45%	59.44 ms	28,509.57 PHP

Table 3.39 shows the result of the testing conducted, including the constraints of the system. The Honor Play got the highest accuracy with 100%, while Xiaomi Redmi k30 Pro got the lowest with only 20%. In terms of speed, the Honor 8X got the highest speed of 59.44 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

EfficientDet Overall Test Result:

Table 3.40 EfficientDet Average Result

CAPTURING DEVICE	EQUIVALENT PHONE MODEL	ACCURACY	SPEED	COST
MakerFocus RPI Camera Module	iPhone SE	87%	61.10 ms	8,809.57 PHP
Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera	Xiaomi Redmi k30 Pro	90%	61.40 ms	42,209.57 PHP
Arducam for Raspi (IMX298)	Huawei Nova 2i	94%	59.30 ms	9,609.57 PHP
Arducam 16MP (IMX298)	Honor Play	92%	61.13 ms	10,109.57 PHP
Lucid Vision Labs Phoenix 20MP, Sony IMX 183	Honor 8X	47.33%	61.17 ms	28,509.57 PHP

Table 3.40 shows the overall result of the testing conducted, including the constraints of the system. Huawei Nova 2i got the highest accuracy of 94%, while the Honor 8X got the lowest with 47.33%. In terms of speed, the Huawei Nova 2i is the fastest with 59.30 ms, and the Xiaomi Redmi k30 Pro got the slowest with 61.40 ms. Meanwhile, the MakerFocus RPI Camera Module got the lowest cost of 8,809.57 PHP.

EfficientDet Testing

Figure 3.31 Eyebill System using EfficientDet

Figure 3.31 shows the actual testing using the EfficientDet, there are a total of 60 tests conducted per capturing device and as shown in table 3.34, Xiaomi Redmi k30 Pro got the highest accuracy of 90% and Huawei Nova 2i is the fastest with 59.30 ms. The creation of the dataset for this algorithm is done through a controlled environment which has the standard as seen in Figure 3.14.

CHAPTER IV: CONSTRAINTS, TRADE-OFFS, AND STANDARDS

This chapter focuses on the constraints considered in the design option of the Eyebill system. This chapter includes the design trade-offs and sensitivity analysis for each design option, considering the design constraints. The overall ranking of capturing devices was also sought out to determine the most suitable for each design option.

A. Design Constraints

Accuracy

This project is concerned with the output accuracy of the overall system. The design to be implemented must be with high accuracy and an acceptable rate of failure. Bad equipment, poor data processing, or human error can lead to inaccurate results that are not very close to the truth. Accuracy can help increase users' satisfaction. The result of the accuracy depends on the type of camera and its specifications, such as the camera sensor, image resolution, and the camera pixel. To measure the accuracy, each algorithm was tested and evaluated to determine the best algorithm that could provide the users with the best accuracy of the system.

Cost

The project should be able to produce a platform that is affordable and is worth the price. In a business dictionary, cost is an amount that has to be paid or given up to get something. In business, the cost is usually a monetary valuation of effort, material, resources, time and utilities consumed, risks incurred and opportunity forgone in production and delivery of a good or service. For this project, the materials must be cost-efficient as possible, with the overall performance of the design justifying the cost. [8]

Speed

A platform that can produce an output without any significant delay was created. This will avoid the hassle for the users and help them utilize their time in using the platform. Significant delays may cause errors and unnecessary waiting time for the users. The speed from the input to

output should be measured. The testing is done to determine which algorithm can provide the fastest output with accurate results for the user.

B. Design Trade-offs

Design trade-off is a condition or situation that involves losing one quality, factor, or aspect to give way to which constraint should be chosen in the design. The trade-off is a concept where the proponents need to choose the best design depending on the given objective and constraint. In selecting the most feasible design, trade-off analysis is performed on the design and criteria to determine which among them is best suited for final design implementation.

There are two (2) trade-off analyses done in this project. First is choosing the best capturing device for the algorithm of the model considering the accuracy, speed, and cost constraints. After choosing the best capturing device another trade-off analysis was done for which design option is the best still considering the same constraints.

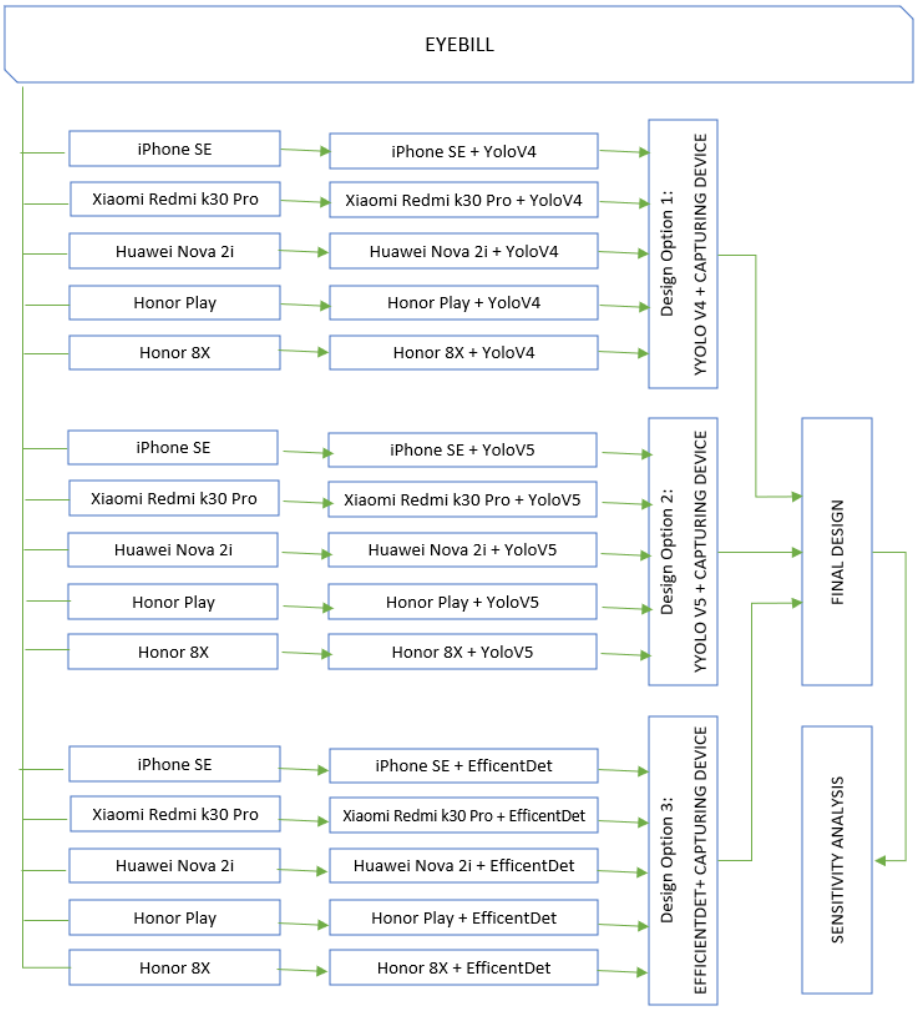


Figure 4.1 Trade-Off Development Flowchart

In Figure 4.1, five (5) capturing devices for each design option were used; the training dataset was then used to create five (5) models for each algorithm. The best capturing device for the model was determined using trade-off analysis which is most suitable for the objectives of the platform. Afterwhich, a sensitivity analysis was implemented on the constraints by tweaking the important factors to determine how different variables affect a particular dependent variable under a given set of assumptions.

1. **Trade-off Analysis (Capturing Device and Yolov4)**

Table 4.1 Tradeoff Analysis Capturing Device Value

Constraints	iPhone SE	Xiaomi Redmi	Huawei Nova 2i	Honor Play	Honor 8X
Accuracy	92.33	94.33	95.2	97.67	61.33
Speed	47.59	35.82	47.47	44.64	47.64
Cost	8,809.57	42,209.57	9,609.57	10,109.57	28,509.57

Table 4.1 shown above signifies the value from the testing done in the Design Option 1 in Chapter 3. The output value helps to determine the best capturing device for the system while considering the constraints given. Each constraint was chosen to help shape the platform to fit the exact needs of the client.

Table 4.2 Tradeoff Analysis Capturing Device and Yolov4 Rank

Capturing Device	Accuracy Rank	Speed Rank	Cost Rank
iPhone SE	4.73	3.77	5.00
Xiaomi Redmi	4.83	5	1.05
Huawei Nova 2i	4.87	3.78	4.59
Honor Play	5	4.02	4.36
Honor 8x	3.14	3.76	1.55

Table 4.2 signifies the rank based on design constraints, accuracy, speed, and cost. Each value was evaluated using the Kirkwood and Anton Son Method, as shown below.

$$Ranking\ Value\ = \left[1 - \left(\frac{HV - LV}{GV}\right)\right] x\ 5$$

Where:

HV = Highest Value
LV = Lowest Value
GV = Governing Value

See Appendix F.1 for the detailed computation of values in Table 4.2

Table 4.3 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	40
Speed	30
Cost	30

Table 4.3 shows the design constraints for accuracy, speed, and cost; the client has more importance to accuracy with less importance in cost and speed of the platform, which gives a criterion importance factor of 40 to accuracy, and a criterion importance factor of 30 in both cost and speed.

Table 4.4 Design Option 1 Trade-off Analysis Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	189.2	113.1	150	**452.3**
Xiaomi Redmi	193.2	150	31.5	**374.66**
Huawei Nova 2i	194.8	113.4	137.7	**446.04**
Honor Play	200	120.6	130.8	**451.4**
Honor 8x	125.6	112.8	46.5	**284.9**

Table 4.4 shows the overall ranking of the design option 1. The iPhone SE got the highest ranking with 452.3 while Honor 8x got the lowest with 284.9. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.1 for the detailed computation of values in Table 4.4

Design Option 1: Capturing Device and Yolov4 Sensitivity Analysis 1

Table 4.5 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	33.33
Speed	33.33
Cost	33.33

Table 4.5 shows the first (1st) sensitivity analysis of Design Option 1. In the analysis, the priority criteria of importance are adjusted such that the different constraints have an equal value of 33.33.

Table 4.6 Design Option 1 Sensitivity Analysis 1 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	157.65	125.65	166.65	**449.96**
Xiaomi Redmi	160.98	166.65	35.00	**362.60**
Huawei Nova 2i	162.32	125.99	152.98	**441.41**
Honor Play	166.65	133.99	145.32	**445.96**
Honor 8x	104.66	125.32	51.66	**281.64**

Table 4.6 shows the overall ranking of the design option 1. The iPhone SE got the highest ranking with 449.96 while Honor 8x got the lowest with 281.64. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.1 for the detailed computation of values in Table 4.6

Design Option 1: Capturing Device and Yolov4 Sensitivity Analysis 2

Table 4.7 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	60
Speed	20
Cost	20

Table 4.7 shows the second (2nd) sensitivity analysis of Design Option 1. In the analysis, the priority criteria of importance are accuracy with a value of 60 and an equal factor of speed and cost with a value of 20.

Table 4.8 Design Option 1 Sensitivity Analysis 2 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	283.8	75.4	100	**459.20**
Xiaomi Redmi	289.8	100	21	**410.74**
Huawei Nova 2i	292.2	75.6	91.8	**459.81**
Honor Play	300	80.4	87.2	**467.60**
Honor 8x	188.4	75.2	31	**294.60**

Table 4.8 shows the overall ranking of the design option 1. Honor Play got the highest ranking with 467.60 while Honor 8x got the lowest with 294.60. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.1 for the detailed computation of values in Table 4.8

Design Option 1: Capturing Device and Yolov4 Sensitivity Analysis 3

Table 4.9 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	60
Cost	20

Table 4.9 shows the third (3rd) sensitivity analysis of Design Option 1. In the analysis, the priority criteria of importance are speed with a value of 60 and an equal factor of accuracy and cost with a value of 20.

Table 4.10 Design Option 1 Sensitivity Analysis 3 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	94.6	226.2	100	**420.80**
Xiaomi Redmi	96.6	300	21	**417.58**
Huawei Nova 2i	97.4	226.8	91.8	**416.07**
Honor Play	100	241.2	87.2	**428.40**
Honor 8x	62.8	225.6	31	**319.40**

Table 4.10 shows the overall ranking of design option 1. The Honor Play got the highest ranking with 428.40 while Honor 8x got the lowest with 319.4. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.1 for the detailed computation of values in Table 4.10

Design Option 1: Capturing Device Sensitivity Analysis 4

Table 4.11 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	20
Cost	60

Table 4.11 shows the fourth (4th) sensitivity analysis of Design Option 1. In the analysis, the priority criteria of importance have cost with a value of 60 and an equal factor of accuracy and speed with a value of 20.

Table 4.12 Design Option 1 Sensitivity Analysis 4 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	94.6	75.4	300	**470.00**
Xiaomi Redmi	96.6	100	63	**259.58**
Huawei Nova 2i	97.4	75.6	275.4	**448.47**
Honor Play	100	80.4	261.6	**442.00**
Honor 8x	62.8	75.2	93	**231.00**

Table 4.12 shows the overall ranking of design option 1. The iPhone SE got the highest ranking with 470, while Honor 8x got the lowest with 231. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.1 for the detailed computation of values in Table 4.12

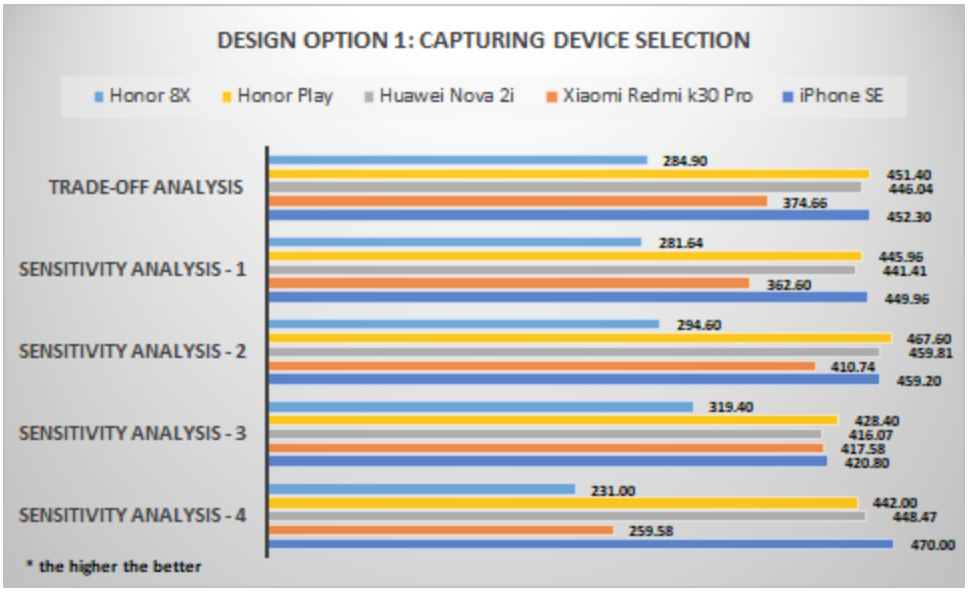


Figure 4.2 Summary of Trade-Off, Sensitivity Analysis, and Overall Ranking of Capturing Device on Design Option 1

As shown in figure 4.2, there are 4 Sensitivity Analyses performed aside from the Trade-Off Analysis for the capturing device of Design Option 1, which are represented on the y-axis of the figure. On the other hand, the x-axis represented the overall rankings. From this bar chart, one can see that for every analysis, the iPhone SE is leading and holds the highest overall ranking.

2. **Trade-off Analysis (Capturing Device and Yolov5)**

Table 4.13 Trade-off Analysis Capturing Device Value

Constraints	iPhone SE	Xiaomi Redmi	Huawei Nova 2i	Honor Play	Honor 8X
Accuracy	80.33	79.33	83.33	70.67	54.67
Speed	12.6	12.91	12.78	13.14	12.69
Cost	8,809.57	42,209.57	9,609.57	10,109.57	28079.57

Table 4.13 shown above signifies the value from the testing done on Design Option 2. The output value helps to determine the best capturing device for the system while considering the constraints given. Each constraint was chosen to help shape the platform to fit the exact needs of the client.

Table 4.14 Trade-off Analysis Capturing Device and Yolov5 Rank

Capturing Device	Accuracy Rank	Speed Rank	Cost Rank
iPhone SE	4.59	5	5.00
Xiaomi Redmi	4.13	4.88	1.05
Huawei Nova 2i	5.00	4.93	4.59
Honor Play	4.95	4.8	4.36
Honor 8x	3.33	4.97	1.55

Table 4.14 signifies the rank based on design constraints, accuracy, speed, and cost. Each value was evaluated using the Kirkwood and Anton Son Method, as shown below:

$$Ranking\ Value\ = \left[1 - \left(\frac{HV - LV}{GV}\right)\right] x\ 5$$

Where:

HV = Highest Value, LV = Lowest Value, GV = Governing Value

See Appendix F.2 for the detailed computation of values in Table 4.14

Table 4.15 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	40
Speed	30
Cost	30

Table 4.15 shows the design constraints for accuracy, speed, and cost; the client has more importance to accuracy with less importance in cost and speed of the platform, which gives a criterion importance factor of 40 to accuracy, and a criterion importance factor of 30 in both cost and speed.

Table 4.16 Design Option 2 Trade-off Analysis Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	183.6	150	150	**483.60**
Xiaomi Redmi	165.2	146.4	31.5	**343.10**
Huawei Nova 2i	200	147.9	137.7	**485.60**
Honor Play	198	144	130.8	**472.80**
Honor 8x	133.2	149.1	46.5	**328.8**

Table 4.16 shows the overall ranking of design option 1. The Honor Play got the highest ranking with 485.60 while the Honor 8x got the lowest with 328.8. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.2 for the detailed computation of values in Table 4.16

Design Option 2: Capturing Device Sensitivity Analysis 1

Table 4.17 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	33.33
Speed	33.33
Cost	33.33

Table 4.17 shows the first (1st) sensitivity analysis of Design Option 2. In the analysis, the priority criteria of importance are adjusted such that the different constraints have an equal value of 33.33.

Table 4.18 Design Option 2 Sensitivity Analysis 1 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	152.98	166.65	166.65	**486.28**
Xiaomi Redmi	137.65	162.65	35.00	**335.30**
Huawei Nova 2i	166.65	164.32	152.98	**483.95**
Honor Play	164.98	159.98	145.32	**470.29**
Honor 8x	110.99	165.65	51.66	**328.30**

Table 4.18 shows the overall ranking of design option 2. The iPhone SE got the highest ranking with 486.28, while Xiaomi Redmi got the lowest with 328.30. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.2 for the detailed computation of values in Table 4.18

Design Option 2: Capturing Device Sensitivity Analysis 2

Table 4.19 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	60
Speed	20
Cost	20

Table 4.19 shows the second (2nd) sensitivity analysis of Design Option 2. In the analysis, the priority criteria of importance are accuracy with a value of 60 and an equal factor of speed and cost with a value of 20.

Table 4.20 Design Option 2 Sensitivity Analysis 2 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	275.4	100	100	**475.40**
Xiaomi Redmi	247.8	97.6	21	**366.40**
Huawei Nova 2i	300	98.6	91.8	**490.40**
Honor Play	297	96	87.2	**480.20**
Honor 8x	199.8	99.4	31	**330.20**

Table 4.20 shows the overall ranking of design option 1. The Honor Play got the highest ranking with 490.40, while Huawei Nova 2i got the lowest with 330.2. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.2 for the detailed computation of values in Table 4.20

Design Option 2: Capturing Device Sensitivity Analysis 3

Table 4.21 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	60
Cost	20

Table 4.21 shows the third (3rd) sensitivity analysis of Design Option 2. In the analysis, the priority criteria of importance are speed with a value of 60 and an equal factor of accuracy and cost with a value of 20.

Table 4.22 Design Option 2 Sensitivity Analysis 3 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	91.8	300	100	**491.80**
Xiaomi Redmi	82.6	292.8	21	**396.40**
Huawei Nova 2i	100	295.8	91.8	**487.60**
Honor Play	99	288	87.2	**474.20**
Honor 8x	66.6	298.2	31	**395.80**

Table 4.22 shows the overall ranking of design option 2. The iPhone SE got the highest ranking with 491.80, while Honor 8x got the lowest with 395.80. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.2 for the detailed computation of values in Table 4.22

Design Option 2: Capturing Device Sensitivity Analysis 4

Table 4.23 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	20
Cost	60

Table 4.23 shows the fourth (4th) sensitivity analysis of Design Option 2. In the analysis, the priority criteria of importance have cost with a value of 60 and an equal factor of accuracy and speed with a value of 20.

Table 4.24 Design Option 2 Sensitivity Analysis 4 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	91.8	100	300	**491.80**
Xiaomi Redmi	82.6	97.6	63	**243.20**
Huawei Nova 2i	100	98.6	275.4	**474**
Honor Play	99	96	261.6	**456.60**
Honor 8x	66.6	99.4	93	**259**

Table 4.24 shows the overall ranking of design option 2. The iPhone SE got the highest ranking with 491.80 while Xiaomi Redmi got the lowest with 243.20. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.2 for the detailed computation of values in Table 4.24

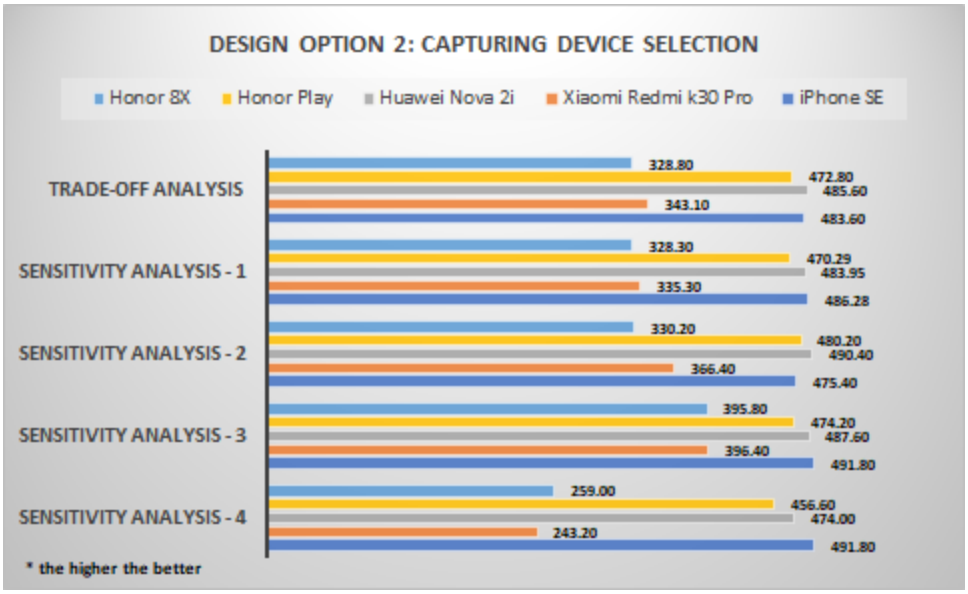


Figure 4.3 Summary of Trade-Off, Sensitivity Analysis, and Overall Ranking of Capturing Device on Design Option 2

As shown in figure 4.3, there are 4 Sensitivity Analyses performed aside from the Trade-Off Analysis for the capturing device of Design Option 2 which are represented on the y-axis of the figure. On the other hand, the x-axis represented the overall rankings. From this bar chart, one can easily see that for every analysis, the Huawei Nova 2i is leading and holds the highest overall ranking.

3. Trade-off Analysis (Capturing Device and EfficientDet)

Table 4.25 Tradeoff Analysis Capturing Device Value

Constraints	iPhone SE	Xiaomi Redmi	Huawei Nova 2i	Honor Play	Honor 8X
Accuracy	87	90	94	92	47.33
Speed	61.1	61.4	59.3	61.13	61.17
Cost	8,809.57	42,209.57	9,609.57	10,109.57	28079.57

Table 4.25, shown above, signifies the value from the testing done on Design Option 3. The output value helps to determine the best capturing device for the system while considering the constraints given. Each constraint was chosen to help shape the platform to fit the exact needs of the client.

Table 4.26 Tradeoff Analysis Capturing Device and EfficientDet Rank

Capturing Device	Accuracy Rank	Speed Rank	Cost Rank
iPhone SE	4.63	4.86	5.00
Xiaomi Redmi	4.79	4.83	1.05
Huawei Nova 2i	5.00	5	4.59
Honor Play	4.89	4.86	4.36
Honor 8x	2.52	4.85	1.55

Table 4.26 signifies the rank based on design constraints, accuracy, speed, and cost. Each value was evaluated using the Kirkwood and Anton Son Method, as shown below:

$$Ranking\ Value\ = \left[1 - \left(\frac{HV - LV}{GV}\right)\right] x\ 5$$

Where:

HV = Highest Value, LV = Lowest Value, GV = Governing Value

See Appendix F.3 for the detailed computation of values in Table 4.26

Table 4.27 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	40
Speed	30
Cost	30

Table 4.27 shows the design constraints for accuracy, speed, and cost; the client has more importance to accuracy with less importance in cost and speed of the platform, which gives a criterion importance factor of 40 to accuracy, and a criterion importance factor of 30 in both cost and speed.

Table 4.28 Design Option 3 Trade-off Analysis Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	185.2	145.8	150	**480.91**
Xiaomi Redmi	191.6	144.9	31.5	**367.89**
Huawei Nova 2i	200	150	137.7	**487.7**
Honor Play	195.6	145.8	130.8	**472.34**
Honor 8x	100.8	145.5	46.5	**292.7**

Table 4.28 shows the overall ranking of design option 1. The Huawei Nova 2i got the highest ranking with 487.70 while the Honor 8x got the lowest with 292.7. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.3 for the detailed computation of values in Table 4.28

Design Option 3: Capturing Device Sensitivity Analysis 1

Table 4.29 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	33.33
Speed	33.33
Cost	33.33

Table 4.29 shows the first (1st) sensitivity analysis of Design Option 3. In the analysis, the priority criteria of importance are adjusted such that the different constraints have an equal value of 33.33.

Table 4.30 Design Option 3 Sensitivity Analysis 1 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	154.32	161.98	166.65	**482.87**
Xiaomi Redmi	159.65	160.98	35.00	**355.54**
Huawei Nova 2i	166.65	166.65	152.98	**486.28**
Honor Play	162.98	161.98	145.32	**470.41**
Honor 8x	83.99	161.65	51.66	**297.22**

Table 4.30 shows the overall ranking of the design option 3. The Huawei Nova 2i got the highest ranking with 486.28, while Honor 8x got the lowest with 297.22. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.3 for the detailed computation of values in Table 4.30

Design Option 3: Capturing Device Sensitivity Analysis 2

Table 4.31 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	60
Speed	20
Cost	20

Table 4.31 shows the design constraints for accuracy, speed, and cost; the client has more importance to accuracy with less importance in cost and speed of the platform, which gives a criterion importance factor of 60 to accuracy, and a criterion importance factor of 20 in both cost and speed.

Table 4.32 Design Option 3 Sensitivity Analysis 2 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	277.80	97.20	100.00	**474.86**
Xiaomi Redmi	287.40	96.60	21.00	**404.83**
Huawei Nova 2i	300.00	100.00	91.80	**491.8**
Honor Play	293.40	97.20	87.20	**478.02**
Honor 8x	151.20	97.00	31.00	**279.05**

Table 4.32 shows the overall ranking of the design option 3. The Huawei Nova 2i got the highest ranking with 491.80 while Honor 8x got the lowest with 279.05. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.3 for the detailed computation of values in Table 4.32

Design Option 3: Capturing Device Sensitivity Analysis 3

Table 4.33 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	60
Cost	20

Table 4.33 shows the third (3rd) sensitivity analysis of Design Option 3. In the analysis, the priority criteria of importance are speed with a value of 60 and an equal factor of accuracy and cost with a value of 20.

Table 4.34 Design Option 3 Sensitivity Analysis 3 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	92.6	291.6	100	**484.15**
Xiaomi Redmi	95.8	289.8	21	**406.54**
Huawei Nova 2i	100	300	91.8	**491.8**
Honor Play	97.8	291.6	87.2	**476.67**
Honor 8x	50.4	291	31	**372.35**

Table 4.34 shows the overall ranking of the design option 3. The Huawei Nova 2i got the highest ranking with 491.80, while Honor 8x got the lowest with 372.35. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.3 for the detailed computation of values in Table 4.34

Design Option 3: Capturing Device Sensitivity Analysis 4

Table 4.35 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	20
Cost	60

Table 4.35 shows the fourth (4th) sensitivity analysis of Design Option 3. In the analysis, the priority criteria of importance have cost with a value of 60 and an equal factor of accuracy and speed with a value of 20.

Table 4.36 Design Option 3 Sensitivity Analysis 4 Overall Ranking

Capturing Device	Accuracy Score	Speed Score	Cost Score	Overall Ranking
iPhone SE	92.6	97.2	300	489.75
Xiaomi Redmi	95.8	96.6	63	255.34
Huawei Nova 2i	100	100	275.4	475.4
Honor Play	97.8	97.2	261.6	456.67
Honor 8x	50.4	97	93	240.35

Table 4.36 shows the overall ranking of the design option 3. The iPhone SE got the highest ranking with 489.75, while Honor 8x got the lowest with 240.35. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.3 for the detailed computation of values in Table 4.36

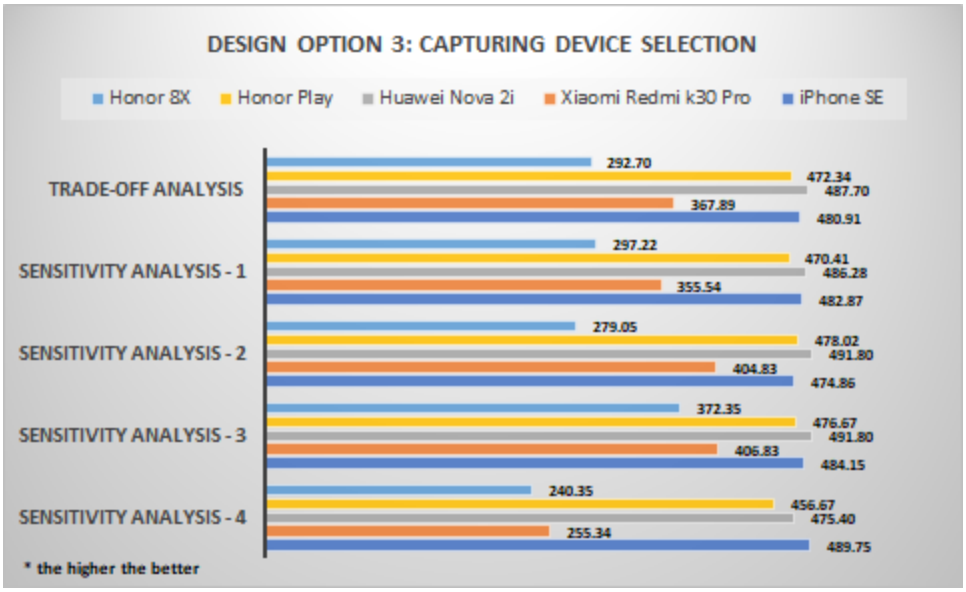


Figure 4.4 Summary of Trade-Off, Sensitivity Analysis, and Overall Ranking of Capturing Device on Design Option 3

As shown in figure 4.4, there are 4 Sensitivity Analyses performed aside from the Trade-Off Analysis for the capturing device of Design Option 3, which are represented on the y-axis of the figure. On the other hand, the x-axis represented the overall rankings. From this bar chart, one can easily see that for every analysis, the Huawei Nova 2i is leading and holds the highest overall ranking.

4. Trade-off Analysis of Design Options

There are two (2) trade-off analyses done in this project. First is choosing the best capturing device for the algorithm of the model considering the accuracy, speed, and cost constraints. After choosing the best capturing device another trade-off analysis was done for which design option is the best still considering the same constraints.

Table 4.37 Final Trade-Off, Sensitivity Analysis, and Overall Ranking

TRADE-OFF ANALYSIS								
Design Criteria	Unit	Criterion Importance Factor	Design Option 1		Design Option 2		Design Option 3	
			Value	Rank	Value	Rank	Value	Rank
ACCURACY	%	40	92.33	4.91	87.33	4.73	94	5.00
SPEED	ms	30	47.59	1.35	12.78	5	59.3	1.08
COST	PHP	30	8,809.57	5	9,609.57	4.59	9,609.57	4.59

Table 4.37 signifies the rank based on the design constraints, accuracy, speed, and cost of each design option. Each value was evaluated using the Kirkwood and Anton Son Method, as shown below:

$$Ranking\ Value\ = \left[1 - \left(\frac{HV - LV}{GV}\right)\right] x\ 5$$

Where:

HV = Highest Value

LV = Lowest Value

GV = Governing Value

See Appendix G.4 for the detailed computation of values in Table 4.37

Table 4.38 Design Options Trade-off Analysis Overall Ranking

Constraints	Design Option 1	Design Option 2	Design Option 3
Accuracy	196.4	189.2	200
Speed	40.5	150	32.4
Cost	150	137.7	137.7
Overall Ranking	**386.9**	**476.9**	**370.1**

Table 4.38 shows the overall ranking of the three (3) design options in which design option 2 showed promising results. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.4 for the detailed computation of values in Table 4.38

Design Options Sensitivity Analysis 1

Table 4.39 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	33.33
Speed	33.33
Cost	33.33

Table 4.39 shows the first (1st) sensitivity analysis of the Design Options. In the analysis, the priority criteria of importance are adjusted such that the different constraints have an equal value of 33.33.

Table 4.40 Design Option 2 Sensitivity Analysis 1 Overall Ranking

Constraints	Design Option 1	Design Option 2	Design Option 3
Accuracy	163.65	157.65	166.65
Speed	45.00	166.65	36.00
Cost	166.65	152.98	152.98

Overall Ranking	**375.30**	**477.29**	**355.63**

Table 4.40 shows the overall ranking of the design options. Design Option 2 got the highest ranking with 477.29 while Design Option 3 got the lowest with 355.63. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.4 for the detailed computation of values in Table 4.40

Design Options Sensitivity Analysis 2

Table 4.41 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	60
Speed	20
Cost	20

Table 4.41 shows the second (2nd) sensitivity analysis of the Design Options. In the analysis, the priority criteria of importance are accuracy with a value of 60 and an equal factor of speed and cost with a value of 20.

Table 4.42 Design Option 2 Sensitivity Analysis 2 Overall Ranking

Constraints	Design Option 1	Design Option 2	Design Option 3
Accuracy	294.6	283.8	300
Speed	27	100	21.6
Cost	100	91.8	91.8
Overall Ranking	**421.6**	**475.6**	**413.4**

Table 4.42 shows the overall ranking of the design options. Design Option 2 got the highest ranking with 475.6 while Design Option 3 got the lowest with 413.4. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.4 for the detailed computation of values in Table 4.42

Design Options Sensitivity Analysis 3

Table 4.43 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	60
Cost	20

Table 4.43 shows the third (3rd) sensitivity analysis of the Design Options. In the analysis, the priority criteria of importance are the speed with a value of 60 and an equal factor of accuracy and cost with a value of 20.

Table 4.44 Design Option 2 Sensitivity Analysis 3 Overall Ranking

Constraints	Design Option 1	Design Option 2	Design Option 3
Accuracy	98.2	94.6	100
Speed	81	300	64.8
Cost	100	91.8	91.8
Overall Ranking	**279.2**	**486.4**	**256.6**

Table 4.44 shows the overall ranking of the design options. Design Option 2 got the highest ranking with 486.4, while Design Option 3 got the lowest with 256.6. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.4 for the detailed computation of values in Table 4.44

Design Options Sensitivity Analysis 4

Table 4.45 Design Constraints Importance Factor

Criteria	Importance Factor
Accuracy	20
Speed	20
Cost	60

Table 4.45 shows the fourth (4th) sensitivity analysis of the Design Options. In the analysis, the priority criteria of importance have cost with a value of 60 and an equal factor of accuracy and speed with a value of 20.

Table 4.46 Sensitivity Analysis 4 Overall Ranking

Constraints	Design Option 1	Design Option 2	Design Option 3
Accuracy	98.2	94.6	100
Speed	27	100	21.6
Cost	300	275.4	275.4
Overall Ranking	**425.2**	**470**	**397**

Table 4.46 shows the overall ranking of the design options. Design Option 2 got the highest ranking with 470 while Design Option 3 got the lowest with 397. Each value was evaluated using the formula:

$$value = \text{i} * (\text{ranking})$$

See Appendix G.4 for the detailed computation of values in Table 4.46

Table 4.47 Summary of Trade-Off, Sensitivity Analysis, and Overall Ranking of all Design Options

Overall Ranking	DO1			Total	DO2			Total	DO3			Total
	A (5)	S (1.47)	C (4.36)		A (4.11)	S (5)	C (5)		A (4.81)	S (1.07)	C (4.59)	
Trade-Off Analysis	196.4	40.5	150	386.9	189.2	150	137.7	**476.9**	200	32.4	137.7	370.1
SA1	163.65	45	166.65	375.30	157.65	166.65	152.98	**477.29**	166.65	36.06	152.98	355.63
SA2	294.60	27	100	421.6	283.8	100	91.8	**475.6**	300	21.6	91.8	413.4
SA3	98.2	81	100	279.2	94.6	300	91.8	**486.4**	100	64.8	91.8	256.6
SA4	98.2	27	300	425.2	94.6	100	275.4	**470**	100	21.6	275.4	397

The results in table 4.47 signify that considering the criterion of importance given by the client, design option 2 is the best design. In the analysis, the priority criteria of importance are adjusted such that the different constraints receive the most importance. For Sensitivity Analysis 1 (SA1) all constraints have equal value, SA2 prioritizes accuracy, SA3 prioritizes speed, and SA4 prioritizes cost. Applying Kirkwood and Anton Son's formula the corresponding ranking is computed which results in design option 2 as the best design. This just shows that the chosen design is an optimal choice for different instances of constraints importance.

See Appendix G.4 for the detailed computation of values in Table 4.47

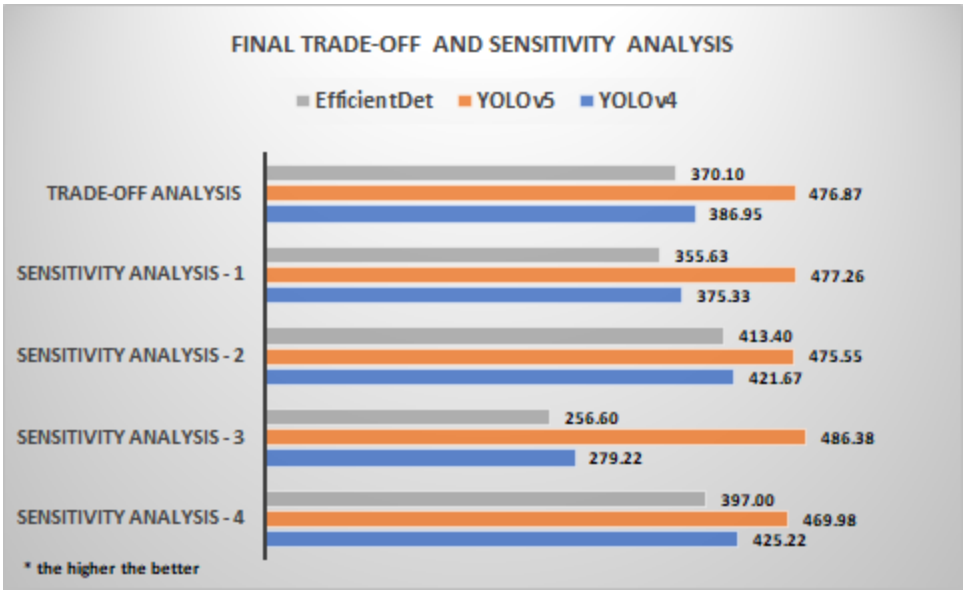


Figure 4.5 Summary of Trade-Off, Sensitivity Analysis, and Overall Ranking of Capturing Device on All Design Options

As shown in figure 4.5, there are 4 Sensitivity Analyses performed aside from the Trade-Off Analysis for the Design Options, which are represented on the y-axis of the figure. On the other hand, the x-axis represented the overall rankings. From this bar chart, one can easily see that adjusting the priority criteria of importance affects the result of the best design option.

CHAPTER V: FINAL DESIGN

This chapter provides information about the actual and implemented design for the platform. The final implemented design is the design option with the highest score based on the result of the tradeoff analysis.

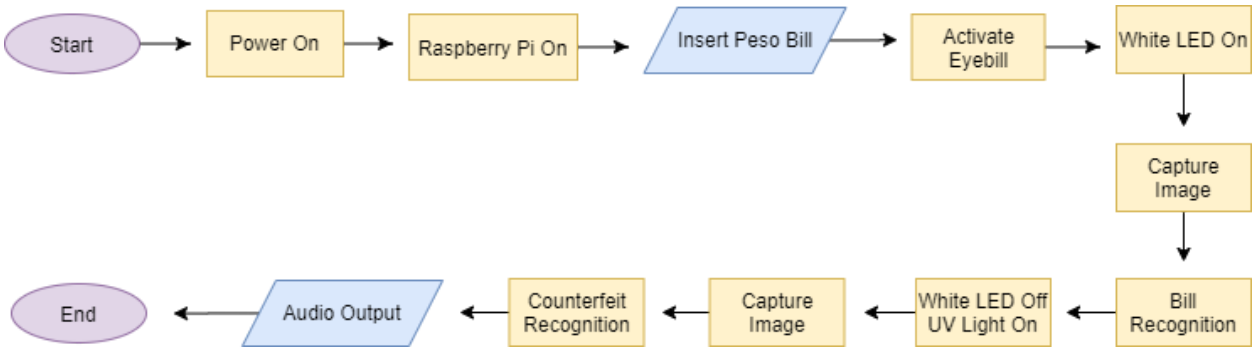


Figure 5.1 Eyebill System Flowchart using the YOLOv5 Algorithm

Figure 5.1 shows the Eyebill system using the Design Option 2: YOLOv5 algorithm, upon starting the platform, the Raspberry Pi automatically turns on. The user inserts the Philippine Peso bill, and upon activation, the white LED is powered-on. The system captures an image of the bill, and upon the bill recognition, it identifies whether the bill is fake or not. After the identification, the platform produces its findings and produces a sound using a speaker indicating the bill's currency.

YOLOv5 Training Process

For the whole training process, Google Colab was used as the platform. Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis, and education [27]

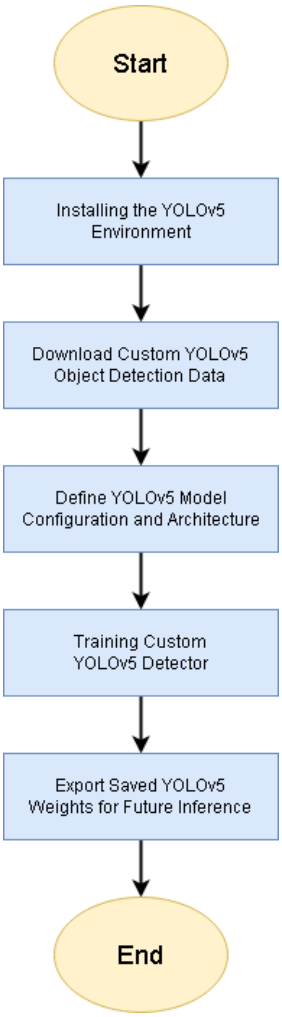


Figure 5.2 Training Process using YOLOv5 Algorithm

Figure 5.2 shows the training process of the YOLOv5 Algorithm. The training starts upon installing the YOLOv5 environment as shown in figure 5.3:

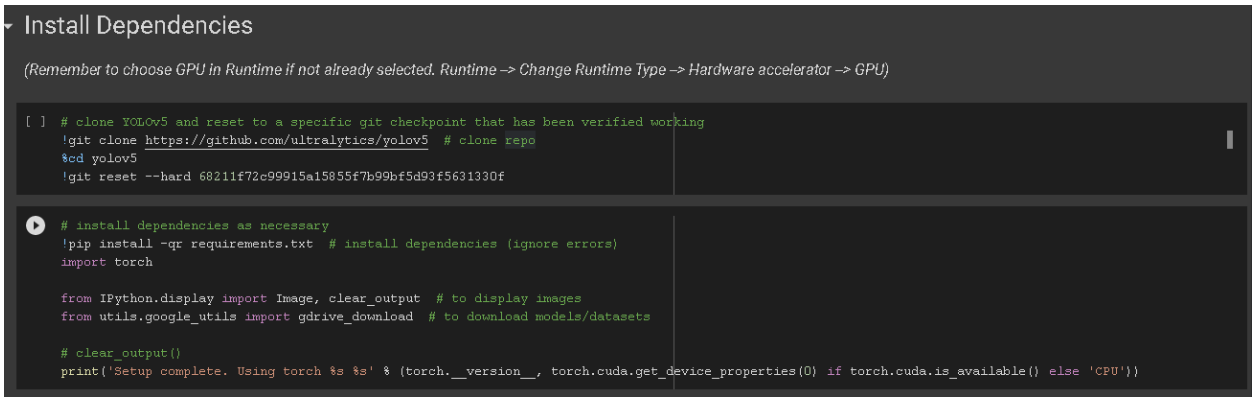


Figure 5.3 Installing the YOLOv5 Environment

After installing the YOLOv5 environment, the next step would be downloading the custom dataset as shown in figure 5.4.

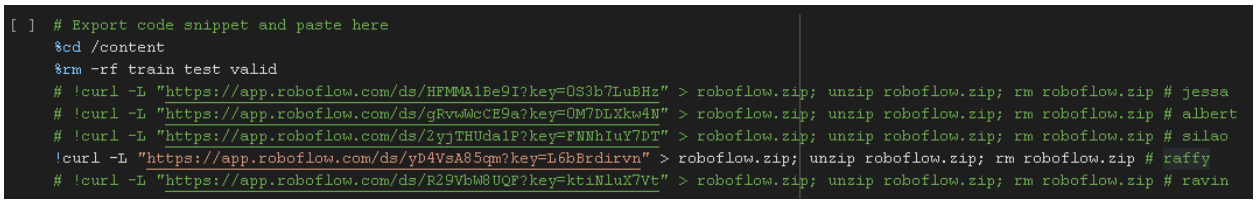


Figure 5.4 Download Custom Dataset

After downloading the custom dataset, a custom training config for YOLOv5 is needed as shown below.

```
%%writetemplate /content/yolov5/models/custom_yolov5m.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.67  # model depth multiple
width_multiple: 0.75  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]
```

```
# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, BottleneckCSP, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, BottleneckCSP, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, BottleneckCSP, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, BottleneckCSP, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

Figure 5.5 Custom Training Config for YOLOv5

After writing the custom training config for YOLOv5, the model configuration and architecture are defined.

After 100 epochs, a final model can now be saved as shown in figure 5.7.

```
# train yolov5m on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 40 --epochs 100 --data '../data.yaml' --cfg ./models/custom_yolov5m.yaml --weights '' --name yolov5m_results  --cache

# $ python train.py --data coco.yaml --cfg yolov5s.yaml --weights '' --batch-size 64
#                                          yolov5m                                40
#                                          yolov5l                                24
#                                          yolov5x                                16
```

Figure 5.6 Custom Training Config for YOLOv5

Upon training the custom detector, the trained weights can now be saved for future inference as shown in figure 5.7

Export Trained Weights for Future Inference

Now that you have trained your custom detector, you can export the trained weights you have made here for inference on your device elsewhere

```
[ ]  %cp /content/yolov5/runs/exp0_yolov5m_results/weights/best.pt /content/gdrive/My\ Drive
```

Figure 5.7 Export Saved YOLOv5 Weights for Future Inference

The Final Design

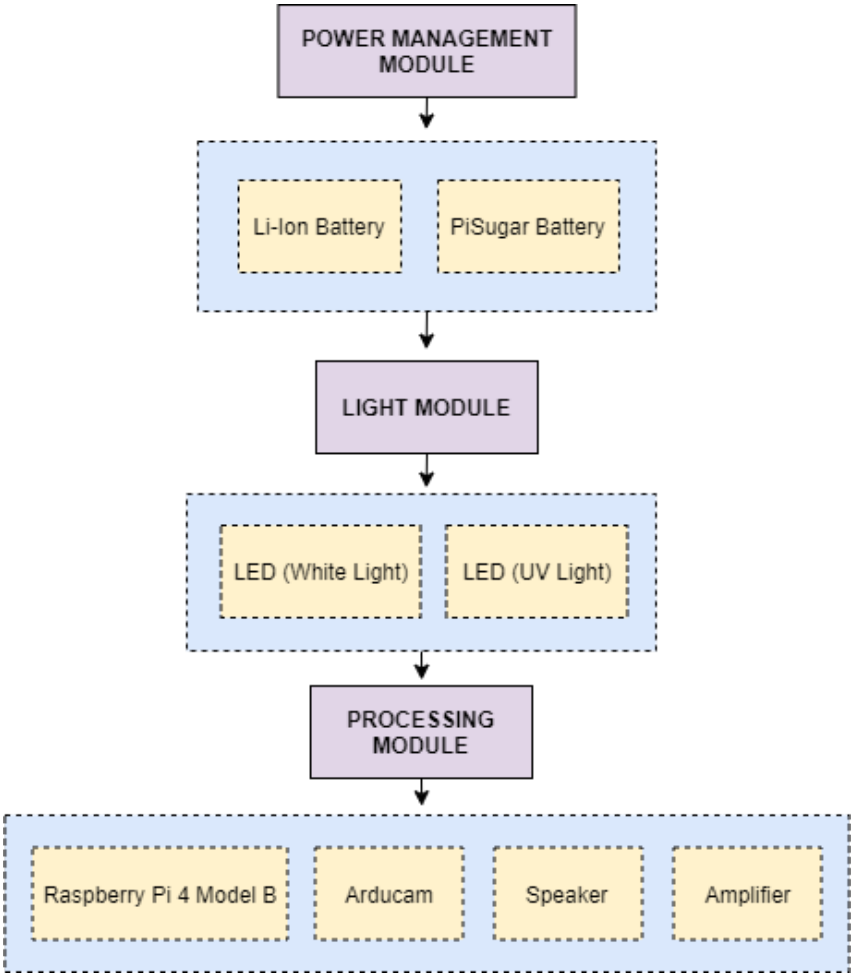


Figure 5.8 Final System Design

Figure 5.8 represents the final system design. The whole system was composed of three major sub-categories: The power management system, the light module, and the processing module. The final system design is the ideal design for the Eyebill considering the objectives of the project.

A. Power Management Module

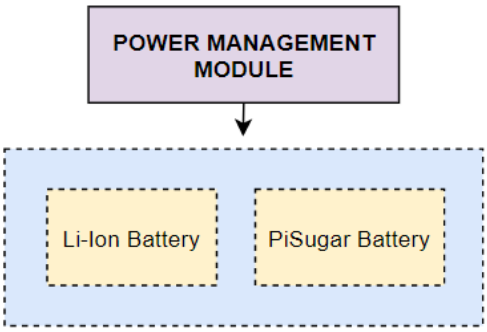


Figure 5.9 Power Management System Chart

Figure 5.9 signifies the Power Management Module of the Eyebill system, which consists of the Li-ion battery and the PiSugar2 Pro Module. This module mainly focuses on how the system will work using the battery and its power consumption. The components mentioned are only the ideal components considering the objectives of the project.

Li-ion Battery - A lithium-ion battery or Li-ion battery is a type of rechargeable battery. Lithium-ion batteries are commonly used for portable electronics and electric vehicles and are growing in popularity for military and aerospace applications. In the batteries, lithium ions move from the negative electrode through an electrolyte to the positive electrode during discharge and back when charging. Li-ion batteries use an intercalated lithium compound as the positive electrode material and typically graphite at the negative electrode. The Li-ion battery was chosen for the Eyebill system because it is rechargeable and small. Compared with traditional battery technology, lithium-ion batteries charge faster, last longer, and have a higher power density for more battery life in a lighter package. It also charges fast for convenience and slows for longevity.

PiSugar2 Pro Module - PiSugar2 Pro is a portable battery platform designed for Raspberry Pi. It integrates UPS, RTC, I2C battery management and custom buttons, the continuous output capability of 5V3A

Figure 5.10 PiSugar2 Pro2 Pro

Figure 5.10 shows the PiSugar2 Pro, it was chosen as the battery for the Eyebill system because compared to other choices (PiJuice or UBEC), it is cheaper and has a high capacity. The following computation indicates the total power consumption for the Eyebill system to function accordingly as well as the energy needed, voltage requirement, and its maximum operating time when idle:

Total Power = P1 + P2 + P3 + … + Pn

Total Power = 19.5W + 1W + 3W + 3W + 1W + 2.5W

Total Power = 30 W

Energy Needed = Total Power * Maximum Operating Time

Energy Needed = 30 W * 0.691 seconds (1 minute/ 60 seconds) (1 hour / 60 minutes)

Energy Needed = 5.76 Wh

Required Voltage = Nominal Voltage * Number of cells.

Required Voltage = 3.7V * 1 cell

Required Voltage = 3.7 Volts

Maximum Operating Time when Idle (minutes) = Battery Capacity / (Total Power / Required Voltage)

Maximum Operating Time when Idle (minutes) = 5000 / (30W / 3.7 Volts)

Maximum Operating Time when Idle (minutes) = 616.67 hours * (60 minutes/ 1 hour)

Maximum Operating Time when Idle (minutes) = 37,002.2 minutes

Table 5.1 Components Table

Components	Power Consumption
Raspi 4 Model B	6.5V, 3A , 19.5 W
Arducam MIPI IMX298 16MP	5V, 200mA , 1W
5mm LED	3W
UV 365 nm	3.4V, 600mA, 3W
Mini Oval Speaker	1W
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	2.5W
For battery:	
PiSugar2 Pro	3.7VDC, 5000mAH

Table 5.1 shows the components chosen for the final design of the Eyebill system, including the power consumption of each component.

B. Light Module

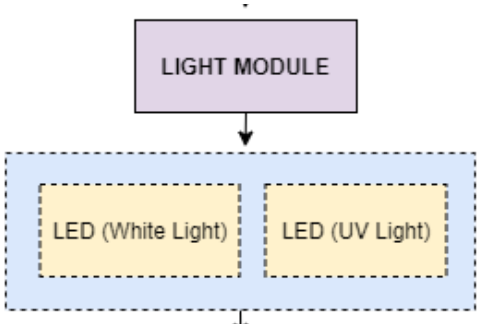


Figure 5.11 Light Module System Chart

Figure 5.11 shows the Light Module of the Eyebill system, which focuses on the light-emitting diode (LED) that is responsible for illuminating the controlled environment of the platform. The components mentioned are only the ideal components considering the objectives of the project.

LED (Power) and LED (white light) - A light-emitting diode (LED) is a semiconductor device that emits light when an electric current is passed through it. Light is produced when the particles that carry the current (known as electrons and holes) combine within the semiconductor material. Since light is generated within the solid semiconductor material, LEDs are described as solid-state devices. The term solid-state lighting, which also encompasses organic LEDs (OLEDs), distinguishes this lighting technology from other sources that use heated filaments (incandescent and tungsten halogen lamps) or gas discharge (fluorescent lamps).

Figure 5.12 Epiled 3W High Power LED Bead Emitter

Figure 5.12 shows the 3W High Power LED, it was chosen for the Eyebill system because of its small size compared to other types of LED. It has a low consumption but high brightness.

LED (UV Light) - UV light is fluorescent tubes with a dark blue or purple filter to remove other parts of the spectrum and leave just UVA light. It is often referred to as ultraviolet 'light,' but UV is a type of electromagnetic radiation with wavelengths shorter than visible light and longer than X-rays. UV (ultraviolet) light bulbs are often referred to as black light bulbs or BLB (blacklight blue) bulbs. UV lights feature small wavelengths that are measured in nanometers (nm). Nanometers often affect the UV light bulb, with the bulb emitting either UVA, UVB, or UVC UV radiation. None of these bulbs can penetrate human skin and are safe to use.

Figure 5.13 UV365 nm

Figure 5.13 shows the UV365 nm which was used for the Eyebill system because of its small size, compactability, low power consumption, and 365 nm is also the wavelength of the UV used by counterfeit devices to detect the luminescence of counterfeit money.

ABS Casing - Acrylonitrile Butadiene Styrene (ABS) is an opaque thermoplastic and amorphous polymer. "Thermoplastic" (as opposed to "thermoset") refers to the way the material responds to heat. Thermoplastics become liquid (i.e. have a "glass transition") at a certain temperature (221 degrees Fahrenheit in the case of ABS plastic). They can be heated to their melting point, cooled, and re-heated again without significant degradation. Instead of burning, thermoplastics like ABS liquefy, which allows them to be easily injection molded and then subsequently recycled.

Figure 5.14 ABS Casing

Figure 5.14 shows the ABS Casing, it was chosen for the Eyebill system because of its strong resistance to corrosive chemicals and/or physical impacts. ABS Casing is very easy to machine and has a low melting temperature making it particularly simple to use in injection molding manufacturing processes or 3D printing on an FDM machine. It is also the most common material used in 3D printing, it is very sturdy and has a long lifespan.

C. Processing Module

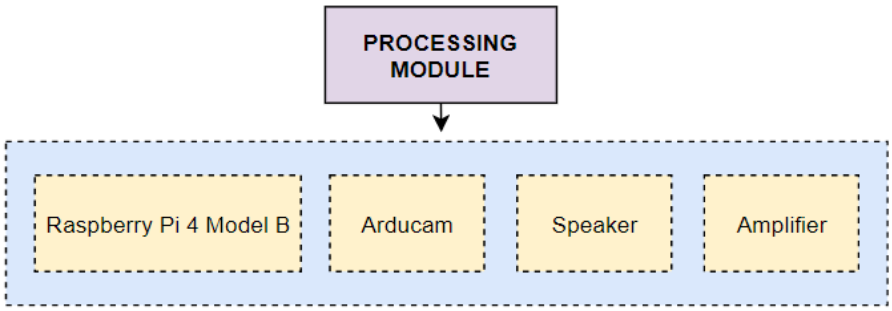


Figure 5.15 Processing Module System Chart

Figure 5.15 shows the Processing Module, which consists of Raspberry Pi 4 Model B, Arducam, and the Speaker; these components serve as the brain of the Eyebill system, which processes the algorithm as well as the audio output. The components mentioned are only the ideal components considering the objectives of the project.

Raspberry Pi 4 Model B - The Raspberry Pi 4 Model B is the latest version of the low-cost Raspberry Pi computer. The Pi isn't like your typical device; in its cheapest form, it doesn't have a case and is simply a credit-card-sized electronic board -- of the type you might find inside a PC or laptop, but much smaller.

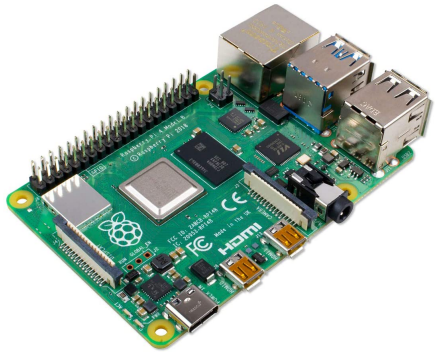


Figure 5.16 Raspberry Pi 4 Model B

Table 5.2 Raspberry Pi 4 Model B Specifications

Specifications	
Chip	Broadcom BCM2711
Core Architecture	Arm Cortex-A72 based processor
CPU	1.5GHz
GPU	500 MHz VideoCore VI
Memory	4GB LPDDR4 RAM
Operating System	Boots from Micro SD card, running a Linux OS / Raspian
Dimensions	3.5 x 2.3 x 0.76 inches (88 x 58 x 19.5mm)
Power	3A, 5V

Table 5.2 shows the specifications of the Raspberry Pi 4 Model B, including its chip, core architecture, CPU, GPU, Memory, OS, Dimensions, and Power. It was chosen for the Eyebill system because of its size, compactibility, high processing speed, and its compatibility with the model that was used on the device.

Arducam MIPI IMX298 16MP - The Arducam IMX298 camera breakout incorporates SONY 1/2.8−inch CMOS, which adopts Exmor-R technology to achieve high-speed image capturing with high sensitivity and low noise performance. “RGBW coding” color filter is employed, and RGB primary color mosaic is reproduced on the chip. High sensitivity, low dark current, and smear-free features are achieved.

Figure 5.17 Arducam MIPI IMX298 16MP

Figure 5.17 shows the Arducam MIPI IMX298, it was chosen as the camera for the Eyebill system based on the testing done. Each capturing device was evaluated using the Kirkwood and Anton Son Method while considering the constraints given by the client as shown in the previous pages.

Speaker - Speakers are made up of a cone, an iron coil, a magnet, and housing (case). When the speaker receives electrical input from a device, it sends the current, causing it to move back and forth. This motion then vibrates the outer cone, generating sound waves picked up by our ears.

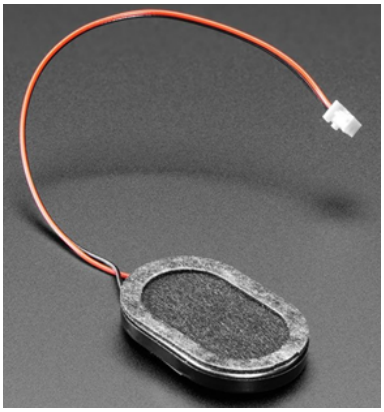


Figure 5.18 1W Molex PicoBlade Speaker

Figure 5.18 shows a 1W Molex PicoBlade speaker, it is small and comes with nice skinny wires with a connector on end. It has a 1.25mm pitch 2-pin cable, which makes plugging into a board easy.

Schematic Diagram

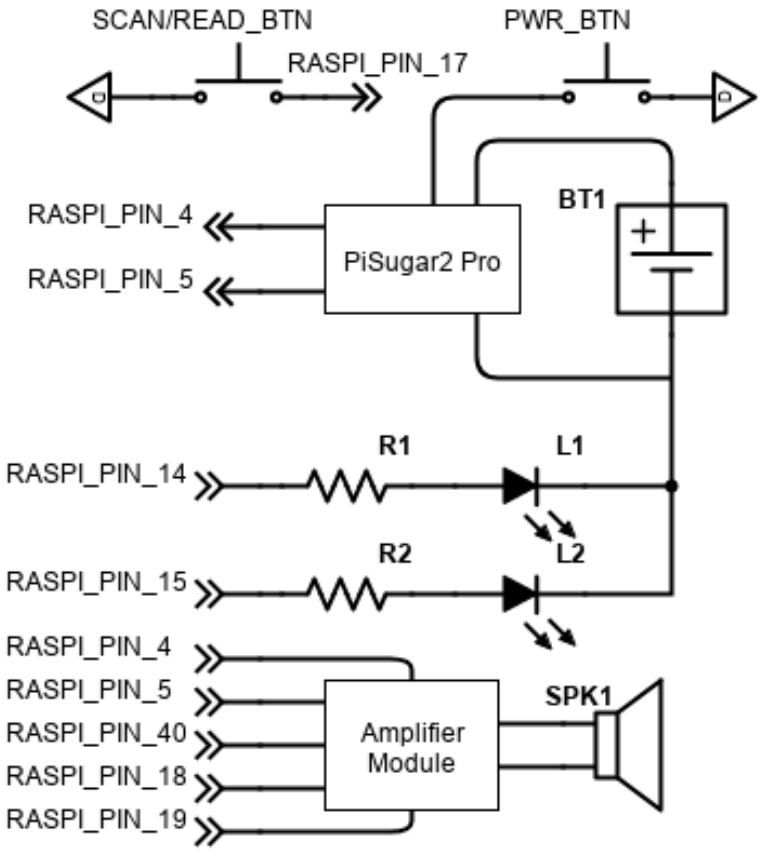


Figure 5.19 Schematic Diagram

Figure 5.19 signifies the schematic diagram of the Eyebill system, which consists of the battery, LED, speaker, resistor, and the GPIO pins of Raspberry Pi 4. The R1, R2, and R3 are the resistors, SPK1 is the speaker, the L1 and L2 are LED and UV light, the Raspi_Pin4, Raspi_Pin5, Raspi_Pin14, Raspi_Pin15, and Raspi_Pin18 are the GPIO pins of the Raspberry Pi 4, and the battery powers all parts.

Table 5.3 BOM of the Eyebill System

Description	Qty	Type	Cost
Arducam 16MP (IMX298)	1	Camera	2,900 PHP
Raspberry Pi 4 Model B	1	Processor	3,249.75 PHP
ABS Casing	1	Casing	700 PHP
Button	1	Button	10 PHP
Mini Oval Speaker	1	Speaker	264.33 PHP
Adafruit Mono 2.5W Class D Audio Amplifier - PAM8302	1	Amplifier	430 PHP
PiSugar2 Pro Battery	1	Battery	2402.49 PHP
Ultraviolet Light	1	UV	150 PHP
LED Light	1	LED	3.00 PHP
TOTAL:			**9,679.57 PHP**

The results in table 5.3 shown above signify the bill of materials with a camera cost of 2,900PHP, processor cost of 3,249.75PHP, casing cost of 700PHP, button cost of 10PHP, speaker cost of 264.33PHP, battery cost of 2402.49PHP, UV cost of 150PHP and LED light cost of 3.00PHP with a total of 9,679.57PHP.

Table 5.4 Weight of Each Component

Components	Weight
Raspberry Pi 4 Model B	46 g
Arducam MIPI IMX298 16MP	4.99 g
Epiled 3W High Power LED Bead Emitter	25mg
UV 365 nm	25mg
Mini Oval Speaker	1.4 g
Casing	95.56 g
For battery:	
PiSugar2 Pro	82.78 g

Table 5.4 shows the weight of each component chosen for the final design of the Eyebill system, the total weight is 230.73g as shown in the computation below:

Total Weight = W1 + W2 + W3 + … + Wn

Total Weight = 46g + 4.99g + 1.4g + 25mg + 25mg + 95.56g + 82.78g

Total Weight = 230.71g

Testing of the Eyebill System

Figure 5.20 Bill Recognition (right) and Bill Authentication (left)

The left image on Figure 5.20 is a one thousand (1000) Philippine Peso bill which was lit by an ordinary LED. This image was used to determine the denomination while the right image uses UV light to determine the authentication.

```
image 1/1 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase1/1000rb.JPG:
 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase2/1000rb.JPG

Predicted Class :1000 || 3
Matched Features : 49.333
Denomination: 1000  || Prediction Time : 0.455s
Time/Image : 0.939s
```

Figure 5.21 Testing Output

Figure 5.21 shows the result of the testing done on Figure 5.5. The predicted class is a one thousand (1000) Philippine Peso bill with 49.33 matching features which signifies that the bill is authentic with a denomination of one thousand (1000) Philippine Peso bill and 0.46 predicted time.

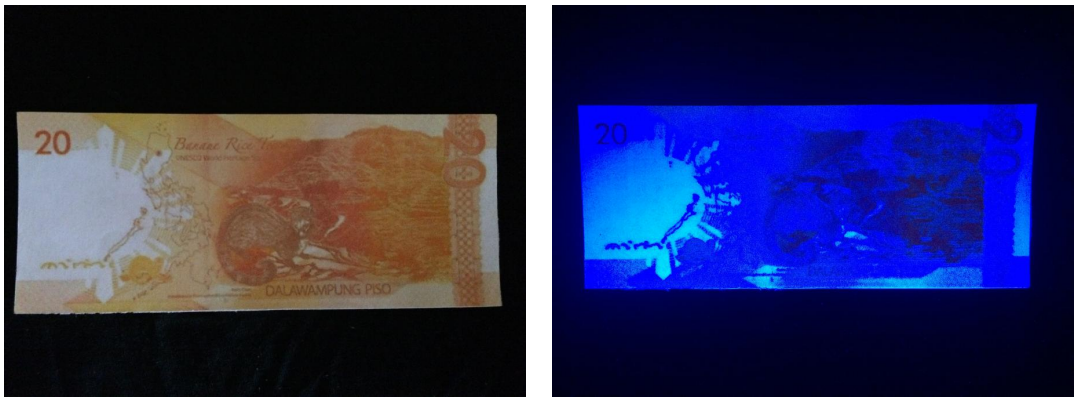


Figure 5.22 Bill Recognition (right) and Bill Authentication (left)

The left image on Figure 5.22 is a twenty (20) Philippine Peso bill which was lit by an ordinary LED. This image was used to determine the denomination while the right image uses UV light to determine the authentication.

```
image 1/1 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase1/20fb.JPG:
 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase2/20fb.JPG

Predicted Class :20 || 3
Predicted Class :500 || 1
Matched Features : 0.000
Denomination: Counterfeit!  || Prediction Time : 0.476s
Time/Image : 1.120s
```

Figure 5.23 Testing Output

Figure 5.23 shows the result of the testing done on Figure 5.7. The predicted class is a twenty (20) Philippine Peso bill with 0 matching features which signifies that the bill is counterfeit without a denomination and a 0.48 predicted time.

Figure 5.24 Bill Recognition (right) and Bill Authentication (left)

The left image on Figure 5.24 is a fifty (50) Philippine Peso bill which was lit by an ordinary LED, this image was used to determine the denomination while the right image uses UV light to determine the authentication.

```
image 1/1 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase1/50rf.JPG:
 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase2/50rf.JPG

Predicted Class :50 || 3
Matched Features : 122.000
Denomination: 50  || Prediction Time : 0.446s
Time/Image : 0.691s
```

Figure 5.25 Testing Output

Figure 5.25 shows the result of the testing done on Figure 5.9. The predicted class is a fifty (50) Philippine Peso bill with 122 matching features which signifies that the bill is authentic with a denomination of fifty (50) Philippine Peso bill and 0.69 predicted time.

Figure 5.26 Bill Recognition (right) and Bill Authentication (left)

The left image on Figure 5.26 is a two-hundred (200) Philippine Peso bill which was lit by an ordinary LED. This image was used to determine the denomination while the right image uses UV light to determine the authentication.

```
image 1/1 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase1/200ff.JPG:
 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase2/200ff.JPG

Predicted Class :200 || 3
Matched Features : 0.250
Denomination: Counterfeit!  || Prediction Time : 0.847s
Time/Image : 1.561s
```

Figure 5.27 Testing Output

Figure 5.27 shows the result of the testing done on Figure 5.11. The predicted class is a two-hundred (200) Philippine Peso bill with 0.250 matching features which signifies that the bill is counterfeit without a denomination and a 0.85 predicted time.

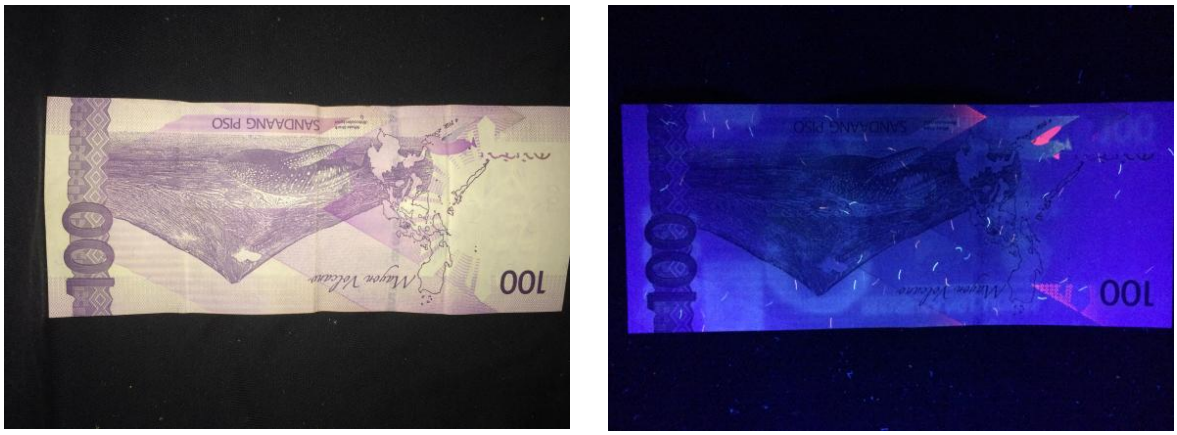


Figure 5.28 Bill Recognition (right) and Bill Authentication (left)

The left image on Figure 5.28 is a one-hundred (100) Philippine Peso bill which was lit by an ordinary LED. This image was used to determine the denomination while the right image uses UV light to determine the authentication.

```
image 1/1 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase1/100rb.JPG:
 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase2/100rb.JPG

Predicted Class :100 || 3
Matched Features : 92.200
Denomination: 100  || Prediction Time : 0.540s
Time/Image : 1.229s
```

Figure 5.29 Testing Output

Figure 5.29 shows the result of the testing done on Figure 5.13. The predicted class is a one-hundred (100) Philippine Peso bill with 92.20 matching features which signifies that the bill is authentic with a denomination of one-hundred (100) Philippine Peso bill and 0.54 predicted time.

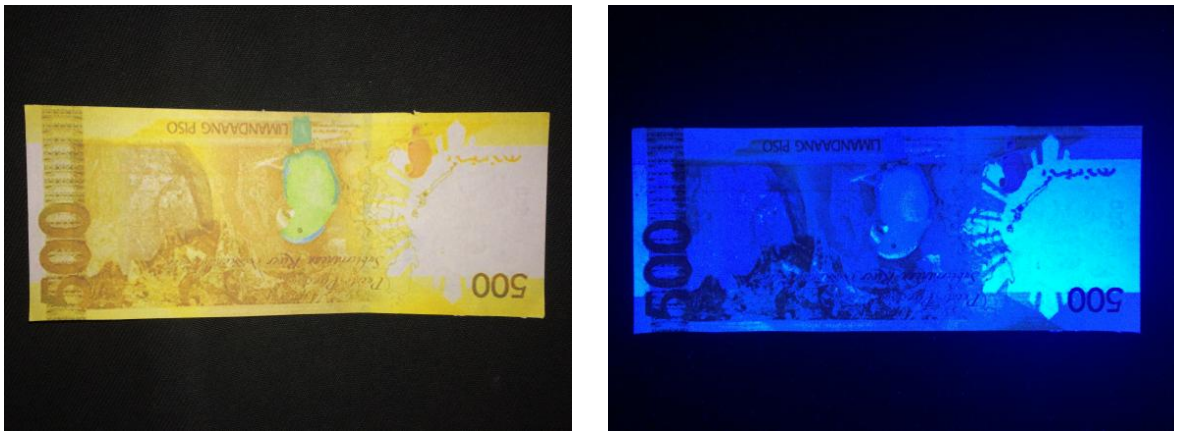


Figure 5.30 Bill Recognition (right) and Bill Authentication (left)

The left image on Figure 5.30 is a five-hundred (500) Philippine Peso bill which was lit by an ordinary LED. This image was used to determine the denomination while the right image uses UV light to determine the authentication.

```
image 1/1 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase1/500fb.JPG:
 /home/bravo/Desktop/LAST SEM 2020 TIP/PD2/testing area/yolov5/phase2/500fb.JPG

Predicted Class :500 || 3
Matched Features : 0.000
Denomination: Counterfeit!  || Prediction Time : 0.462s
Time/Image : 1.807s
```

Figure 5.31 Testing Output

Figure 5.31 shows the result of the testing done on Figure 5.30. The predicted class is a five-hundred (500) Philippine Peso bill with 0 matching features which signifies that the bill is counterfeit without a denomination and a 0.46 predicted time.

Summary of Findings

The Eyebill system is designed to help the visually impaired or blind people in recognizing the denomination of a Philippine Peso Bill. The project was mainly broken down into three (3) design options which focus mainly on the algorithms used. The three design options include YOLOv4, YOLOv5, EfficientDet. The design options were tested using five (5) different capturing devices (*please refer to chapter 3 for a detailed explanation of each capturing device*) which resulted in fifteen (15) models. Smartphones were used for testing while taking into consideration the similarities of the megapixel between the capturing devices and phone cameras. The MakerFocus RPI Camera Module got the lowest cost with 8.809.57 while Allied Vision Prosilica GT6400 APS-C TFL-Mount Color CMOS Camera is the highest with 42,209.57 PHP.

All three (3) design options are object recognition which can recognize multiple objects in a single frame. Design Option 1 is YOLOv4 which is a real-time recognition system that recognizes multiple objects from an image and makes a boundary box around the object. By doing experimental results, the YOLOv4 accuracy resulted from 61.33% to 92.33% while the speed ranges from 35.82 ms to 47.64ms. Design Option 2 is YOLOv5 which is the first object detection model that combines bounding box prediction and object classification into a single end to end differentiable network. YOLOv5 accuracy resulted from 72% to 87.33% while the speed ranges from 12.60 ms to 13.14 ms. Design Option 3 is EfficientDet, a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a *compound coefficient*. The EfficientDet accuracy resulted from 47.33% to 94% while the speed ranges from 59.30 ms to 61.40 ms. Considering the criterion importance given by the client and applying Kirkwood and Anton Son formula, design option 2 resulted as the best design.

Moreover, the Eyebill system uses the ORB algorithm which determines if a Philippine Peso bill is authentic or counterfeit. ORB is a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First, it uses FAST to find key points, then applies the Harris corner measure to find top N points among them. To further assess the attainment of the project objectives, a series of tests were conducted.

Assessment of the Attainment of the Project Objectives

- Project objective #1: To improve the accuracy in determining the denomination of the Philippine Peso Bill by using a controlled environment compared to the current solution.

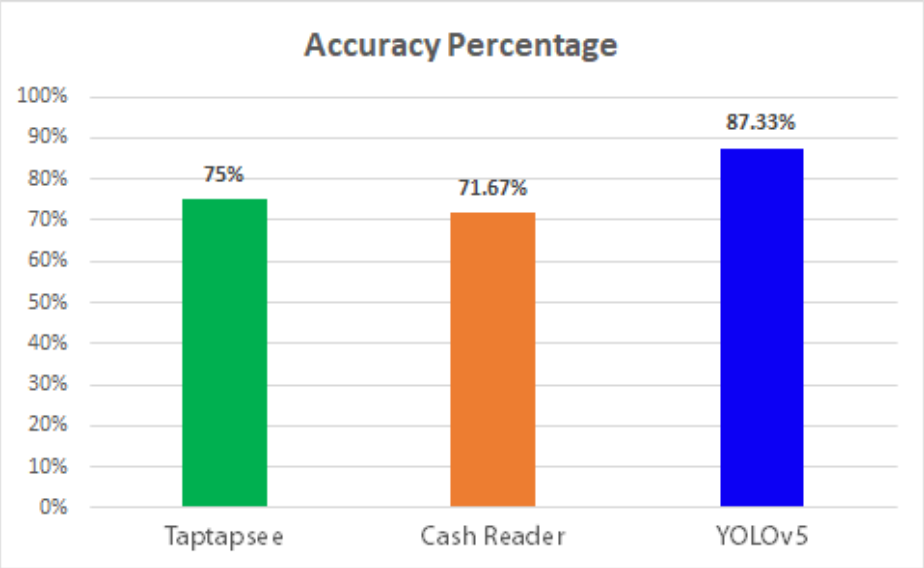


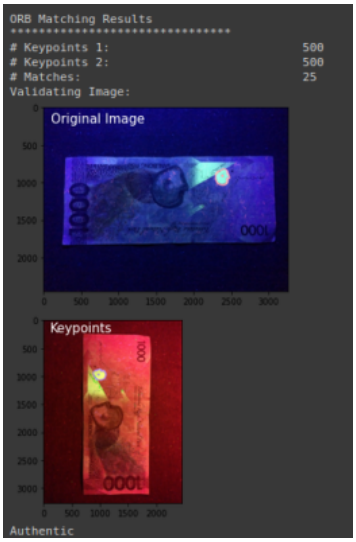
Figure 5.32 Testing Result Comparing Eyebill and the Current Solutions

Figure 5.32 shows the testing result comparing Eyebill, Taptapsee, and Cash Reader (*please refer to Chapter 1 for a detailed explanation of the current solutions)*. Based on the testing mentioned in Table 1.1 and Table 1.2, TapTapsee got the highest accuracy rate of 75% using Honor 8x as the capturing device, and Cash Reader got the highest accuracy of 70% using Honor 8x as the capturing device. Meanwhile, the Eyebill system got an 87.33% accuracy rate (*see Chapter 3 for the detailed explanation of the testing*). This just shows that the Eyebill system was able to produce a higher accuracy rate of 15.66% compared to Cash Reader and a rate of 12.33% compared to Taptapsee. The testing conducted on the existing solutions was done in the same environments using the different capturing devices. Moreover, a total of 10 tests were conducted for each denomination of the Philippine Peso Bill (20, 50, 100, 200, 500, 1000), these resulted in a total of 60 tests for each capturing device.

- Project objective #2: To simulate a platform that can complete its function without the need to rely on an internet connection.

The Eyebill system uses Python as its programming language. Considering that Python and its packages can work in an offline environment, it can be easily concluded that the Eyebill system can serve its function without the need to rely on an internet connection.

- Project objective #3: To simulate a device that can identify whether a Philippine Peso Bill is counterfeit or authentic.



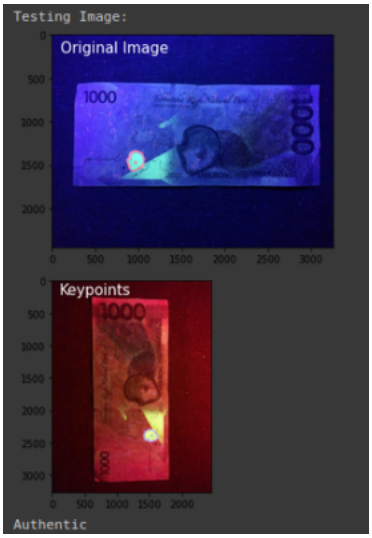
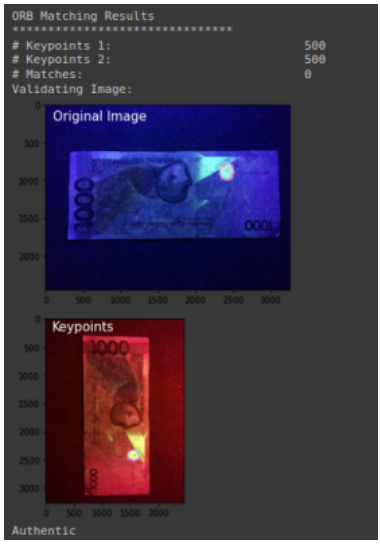


Figure 5.33 ORB Baseline (left) and Testing Result (right)



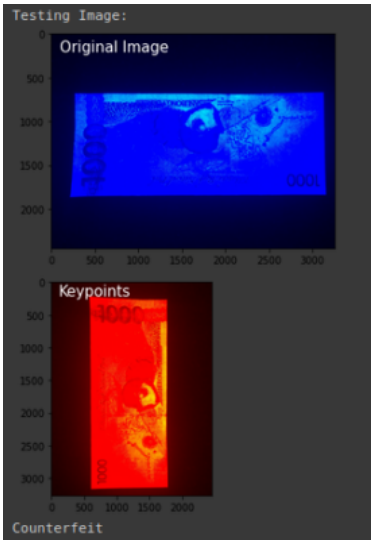
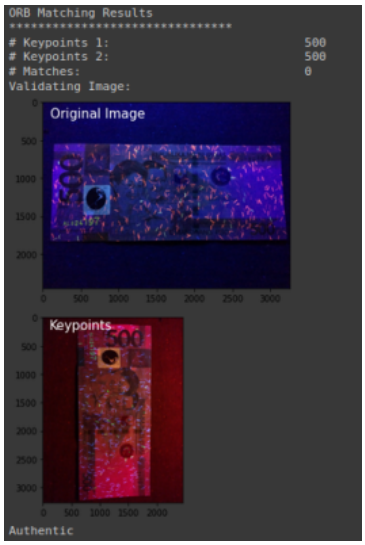


Figure 5.34 ORB Baseline (left) and Testing Result(right)



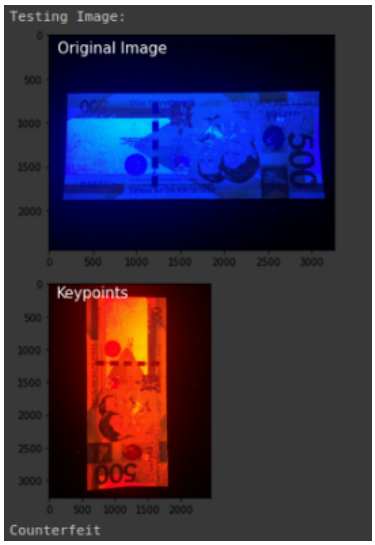
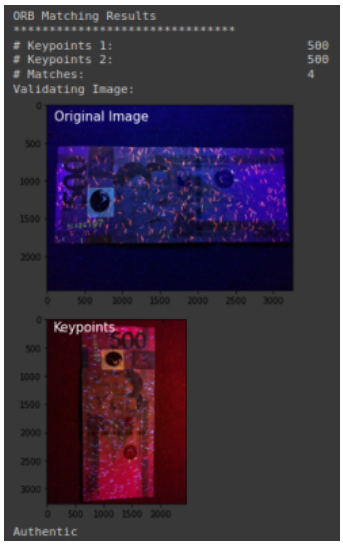


Figure 5.35 ORB Baseline (left) and Testing Result (right)



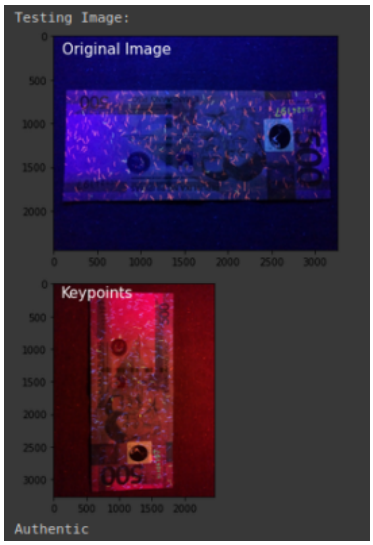
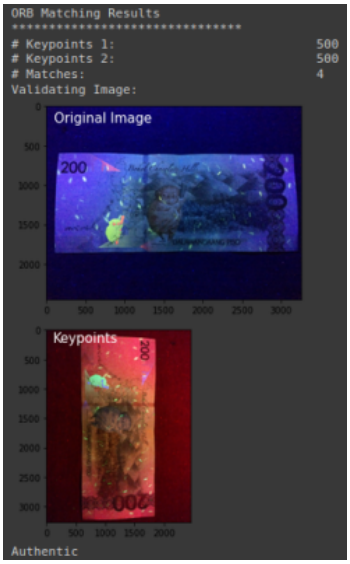


Figure 5.36 ORB Baseline (left) and Testing Result (right)



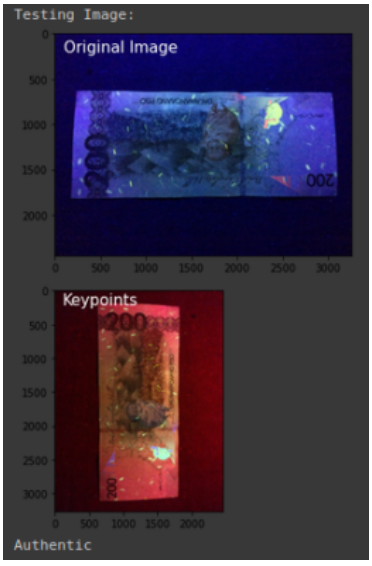
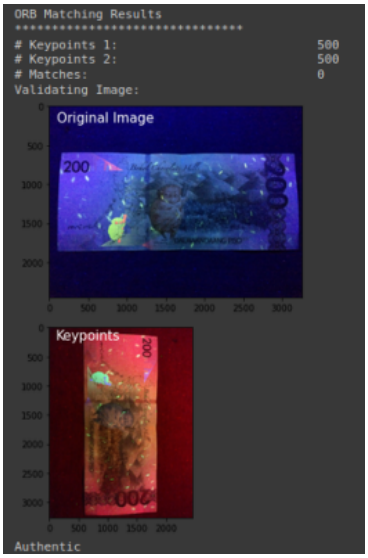


Figure 5.37 ORB Baseline (left) and Testing Result (right)



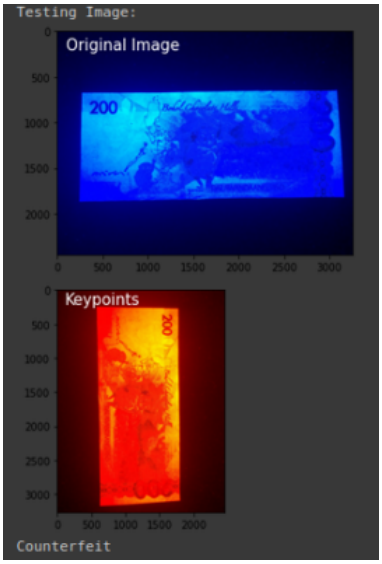


Figure 5.38 ORB Baseline (left) and Testing Result (right)

Figures 5.33 - 5.38 shows the result of the testing using ORB Algorithm. The ORB Baseline on the left side of each figure serves as the basis of the ORB Algorithm to verify if the Philippine Peso Bill is authentic or counterfeit while the right side of each figure is the result. This shows that the Eyebill system can successfully differentiate if the Philippine Peso Bill is counterfeit or authentic.

Assessment of the Attainment of Client Requirements

- Develop a platform that can be used specifically for the blind for them to accurately determine the amount of money.

The Eyebill system is a platform that was created specifically for the blind that helps them to determine the amount of money.

- Make sure that the platform can work offline.

The Eyebill system uses Python as its programming language. Considering that Python and its packages can work in an offline environment, it can be easily concluded that the Eyebill system can serve its function without the need to rely on an internet connection.

- Considering that the target users are blind people, the platform should be easy to access and easy to use.

The existing solutions are available on mobile phones which are built for multiple purposes. Solutions such as TapTapsee and Cashreader require the users to navigate through the mobile device for the application and use the application in determination of the Philippine Peso Bill. Having a platform specifically designed to assist the visually impaired and blind people in determining the denomination of a Philippine Peso Bill is preferred by and ideal for the client and possible users. The platform for the Eyebill System was made and designed for the desired function of the client. The platform only needs to be turned on, insert the Philippine Peso Bill, and press a button to start determining the denomination of the input. To guarantee the ease of access and user comfort while using the platform, the platform was designed as follows:

- The speaker was placed on top of the platform which will provide a clear value of the denomination of the Philippine Peso bill.
- The activate button for the system was placed on the right side of the platform prototype layout where it could be easily reached by the right thumb for ease of access.

- Braille Alphabet translations were engraved on the platform prototype's external casing to avoid confusion for the functions of the ports and buttons on the platform. The Braille Alphabet translations were placed near the buttons and ports.
- The colors yellow and red were used for the casing. The solid color yellow was painted on the top of the platform while red was painted on the sides of the platform's external casing. The aforementioned colors are one of the most visible colors to those who are visually impaired and are not totally blind.
- Type - C and Micro USB Port were made available as charging ports.

- The platform should be able to produce an output fast and must avoid significant delays.

The Eyebill system can produce a denomination of a Philippine Peso Bill with a speed of 12.78 ms or 0.01278 seconds which is considered fast, therefore it can be easily concluded that the Eyebill can avoid significant delay.

- The platform should be cost-effective.

The materials used for the Eyebill system are further explained in Figure 5.3, each component is important to the overall performance of the design which also justifies the cost.

Conclusion

Based on the testing conducted by the proponents on the existing solutions and the final design for the project, the objective which is to improve the accuracy in determining the denomination of the Philippine Peso Bill was successfully achieved. The Eyebill System scored 87.33% accuracy with its results, while the existing solutions such as TapTapsee and Cashreader had 75% and 71.67% in accuracy respectively. Based on the results of the testings, the Eyebill System is 8.33% more accurate than the TapTapsee and is 15.66% more accurate than the Cashreader.

Once the model of the Eyebill System is implemented on the prototype which has a controlled environment for the input, the accuracy would not be affected. The speed of the model in determining the denomination of the Philippine Peso bill runs between 12.60 ms and 13.14 ms. The given speed is

subjected to change. The time spent in turning the LED Light on, capturing an image of the Phillipine Peso bill for recognition, turning the LED Light off, turning the UV Light on, and capturing another image for authenticity and counterfeit detection would be considered as the total speed of the Eyebill system.

Moreover, the proponents were able to complete the project while considering the constraints of accuracy, speed, and cost. The assessment proved that the identified project objective and the client requirements were successfully achieved.

Recommendation

A list of features for further improvement of the design project was provided. These are the following:

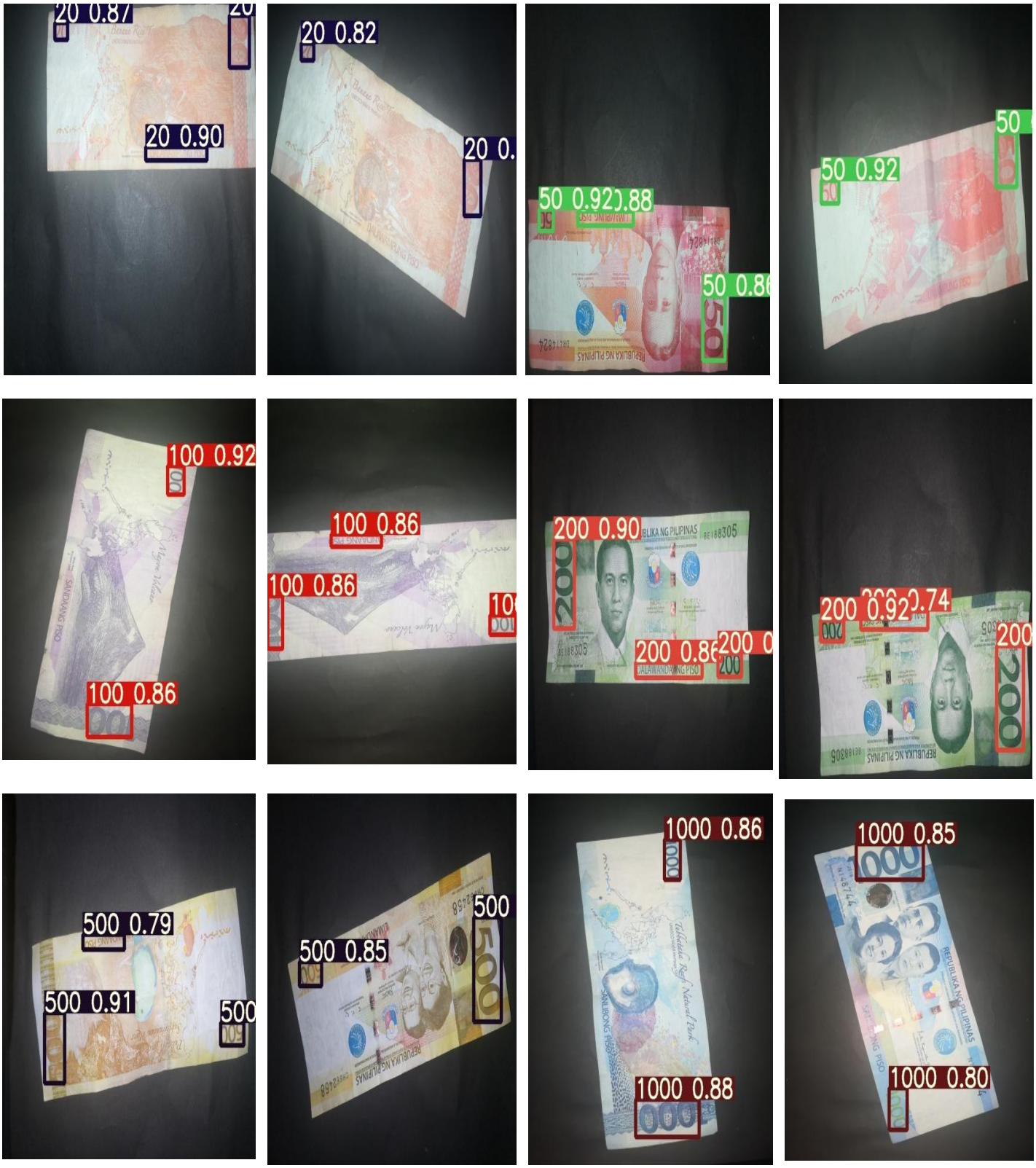
- The testing must be done in a Raspberry Pi to produce a more realistic result in getting the actual speed of the algorithm.
- The prototype must be developed while considering its portability.
- Adding a hardware accelerator stick such as Intel Movidius Vision Processing Units (VPUs) further accelerates the performance of Eyebill.
- Designing a custom IO board using Kicad which helps in reducing the size of the Eyebill.
- Designing a creative way to make an Eyebill a multi-purpose tool such as a purse or a wallet.
- A specification that users can update the Eyebill system if a feature has been added.
- The testing must be done by the blind or visually impaired to produce a more realistic result.

Appendices

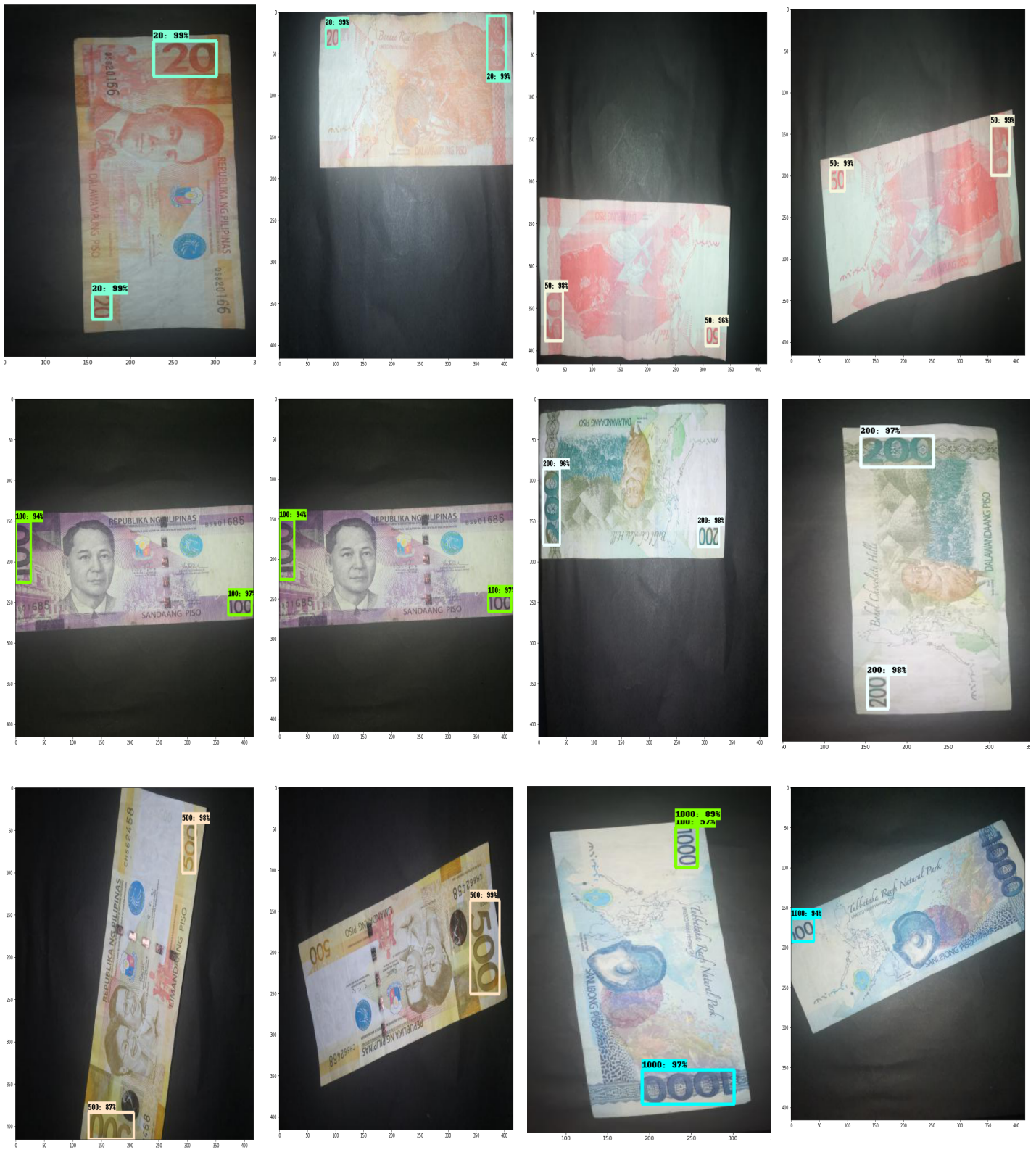
Appendix A. Eyebill Testing

Design Option 1: YOLOv4 Algorithm

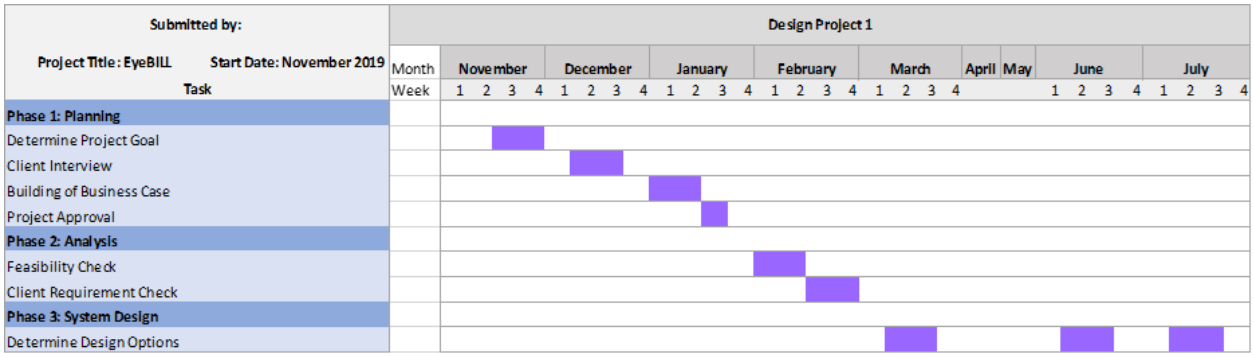
Design Option 2: YOLOv5 Algorithm

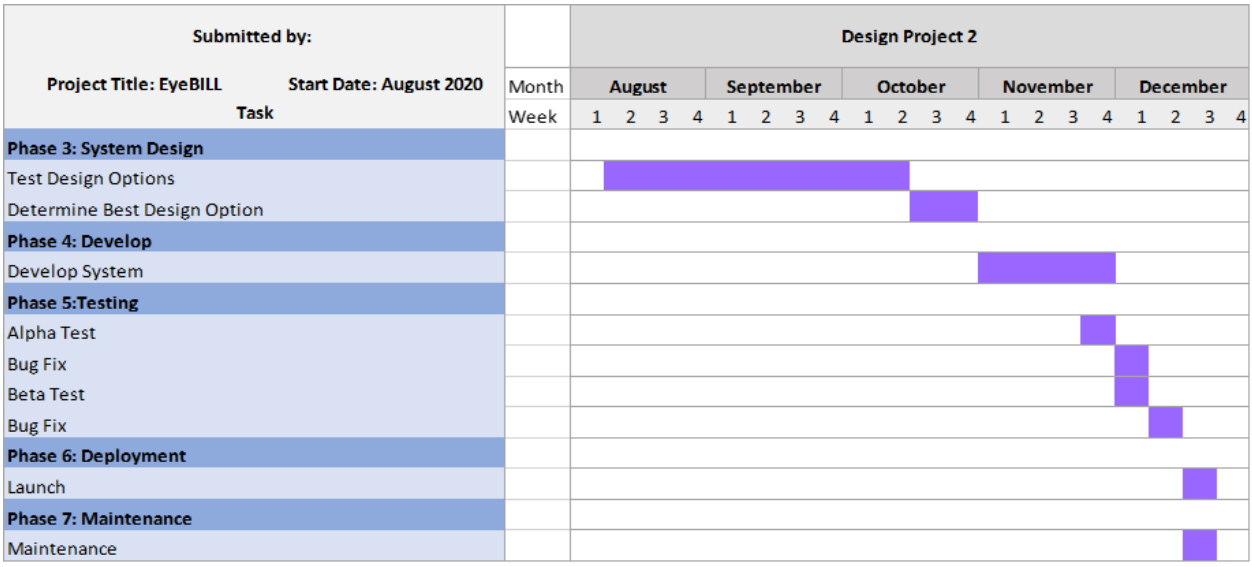


Design Option 3: EfficientDet Algorithm



Appendix B. Gantt Chart





Appendix C. 5 Why's

EFFECT 1:: Too slow in determining the amount of money

WHY 1: Why is the determination of the amount too low?

a. Lots of processes

WHY 2: Why a lot of processes?

The system needs to perform the following processes:

a. Open the app
b. Choose currency
c. Scan the picture of the money

WHY 3a: Why open the app?

The app is installed on a device that is used for a general-purpose.

WHY 3b. Why choose a currency?

The app is for worldwide currency.

WHY 3c: Why scan the picture of the money?

It is the way for the system to work and produce an output

WHY 4: Why is the app installed in a device that is used for a general-purpose?

This is the only compatible platform for the existing system

WHY 5: Why is this the only compatible platform for the system?

The system was designed only for this platform.

EFFECT 2: High percentage of error occurs in determining the amount

WHY 1: Why is there a high percentage of error in determining the amount?

The way the system starts processing is by taking pictures of the money which is kind of challenging since their target client is blind people.

WHY 2: Why do you need to take a picture of the money?

The processing requires a picture of the money

WHY 3: Why does the system require a picture of the money?

A way for the system to have the image of money or currency

WHY 4: Why does the system need the image of money?

To produce output and identify the exact amount

WHY 5: Why does the system need to produce an exact amount?

For the blind to identify the amount of their money.

EYE BILL: Assistive Bill Recognition Device for Visually Impaired and Blind People

User's Manual

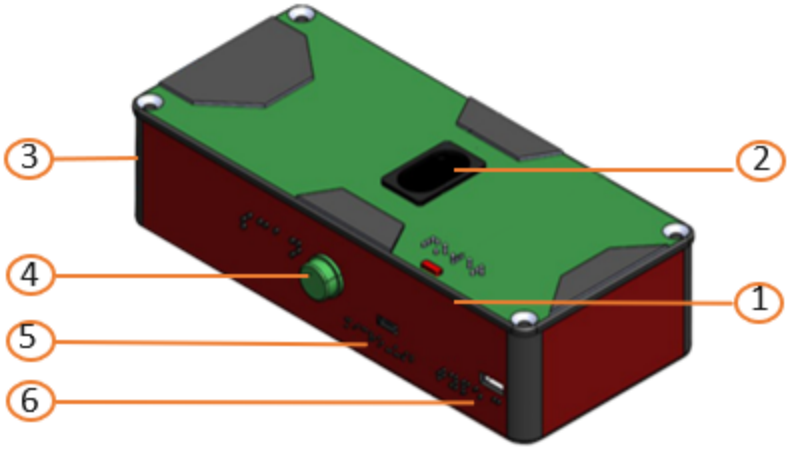
General Description

EYE BILL: Assistive Bill Recognition Device for Visually Impaired and Blind People is a device that can determine the denomination of a Philippine Peso Bill. The device could also identify if the input is authentic or counterfeit.

Accessories

- Eye Bill Device
- AC Adaptor
- USB to Type-C Cable
- User's Manual

Components and Functions



No.	Component	Function
1	Power Button	Turns the device on or off.
2	Speaker	Output Medium.
3	Input Slot	Slot for the bill input.
4	System Start Button	Press this switch to start
5	Micro USB Port	Charging port for Micro USB.
6	Type-C Port	Charging port for Type-C.

Components and Functions

Step 1: Power up the Eye Bill device

Press the power button to turn on the Eye bill device.

Step 2: Insert Bill

Insert the bill to be determined in the slot for the bill.

Step 3: Start the system

Press the system start button to start the system by capturing an image of the output.

Step 4: Wait for the output

The result of the reading is released as an audio through the speaker of the device.

Appendix E. Eyebill System Code

```
#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import os
import csv
from PIL import Image
import matplotlib.pyplot as plt
from collections import Counter
import operator
import ntpath
#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
import argparse
import time
from pathlib import Path
import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random

from models.experimental import attempt_load
from utils.datasets import LoadStreams,
LoadImages
from utils.general import check_img_size,
non_max_suppression, apply_classifier,
scale_coords, xyxy2xywh, \
    strip_optimizer, set_logging, increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device,
load_classifier, time_synchronized
#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz =
opt.source, opt.weights, opt.view_img,
opt.save_txt, opt.img_size
    webcam = source.isnumeric() or
source.endswith('.txt') or
source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://'))

    # Directories
    save_dir =
Path(increment_path(Path(opt.project) /
opt.name, exist_ok=opt.exist_ok))  # increment
run
    (save_dir / 'labels' if save_txt else
save_dir).mkdir(parents=True, exist_ok=True)  #
make dir

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = device.type != 'cpu'  # half precision only
supported on CUDA

    # Load model
    model = attempt_load(weights,
map_location=device)  # load FP32 model
    imgsz = check_img_size(imgsz,
s=model.stride.max())  # check img_size
    if half:
        model.half()  # to FP16

    # Second-stage classifier
    classify = False
    if classify:
        modelc = load_classifier(name='resnet101',
n=2)  # initialize

modelc.load_state_dict(torch.load('weights/resnet
101.pt',
map_location=device)['model']).to(device).eval()
```

```
    # Set Dataloader
    vid_path, vid_writer = None, None
    if webcam:
        view_img = True
        cudnn.benchmark = True  # set True to
speed up constant image size inference
        dataset = LoadStreams(source,
img_size=imgsz)
    else:
        save_img = True
        dataset = LoadImages(source,
img_size=imgsz)

    # Get names and colors
    names = model.module.names if
hasattr(model, 'module') else model.names
    colors = [[random.randint(0, 255) for _ in
range(3)] for _ in names]

    ave_time = []
    # Run inference
    t0 = time.time()
    img = torch.zeros((1, 3, imgsz, imgsz),
device=device)  # init img
    _ = model(img.half() if half else img) if
device.type != 'cpu' else None  # run once

    #phase 1
    # capture under normal light !! Done !!
    #phase 2
    #create/capute uv light
    # os.system("rm -rf phase2")
    # os.system("mkdir phase2")
    def path_leaf(path):
        head, tail = ntpath.split(path)
        return tail or ntpath.basename(head)

    for path, img, im0s, vid_cap in dataset:
        tt_image =
os.getcwd()+"/phase2/"+path_leaf(path) #phase
2
        img = torch.from_numpy(img).to(device)
        img = img.half() if half else img.float()  #
uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        # Inference
        t1 = time_synchronized()
        pred = model(img,
augment=opt.augment)[0]

        # Apply NMS
        pred = non_max_suppression(pred,
opt.conf_thres, opt.iou_thres,
classes=opt.classes,
agnostic=opt.agnostic_nms)
        t2 = time_synchronized()

        # Apply Classifier
        if classify:
            pred = apply_classifier(pred, modelc, img,
im0s)

        dict_sample={}
        print("\n",tt_image,"\n")
        # Process detections
        for i, det in enumerate(pred):  # detections
per image
            # Saving output predictions~~~#much
faster if without
            if webcam:  # batch_size >= 1
                p, s, im0 = Path(path[i]), '%g: ' % i,
im0s[i].copy()
            else:
                p, s, im0 = Path(path), '', im0s
```

```
            save_path = str(save_dir / p.name)
            txt_path = str(save_dir / 'labels' / p.stem)
+ ('_%g' % dataset.frame if dataset.mode ==
'video' else '')
            s += '%gx%g ' % img.shape[2:]  # print
string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]
# normalization gain whwh
            if len(det):
                # Rescale boxes from img_size to im0
size
                det[:, :4] = scale_coords(img.shape[2:],
det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  #
detections per class || #c is class "n" number of
detection
                    s += '%g %ss, ' % (n, names[int(c)])
# add to string
                    #
print(dict_sample[max(dict_sample,
key=dict_sample.get)]) #value[key] 3[1000]
                    dict_sample[names[int(c)]] = int(n)
                    print("Predicted Class
:"+str(names[int(c)])+" || "+ str(int(n)))

                cl = max(dict_sample,
key=dict_sample.get) #class(key) of highest
value
                dn = dict_sample[max(dict_sample,
key=dict_sample.get)] #value of highest
class(key)

                if dn >=2:
                    print("Denomination:
"+str(kaze_valid(cl,tt_image))," || Prediction Time
: %.3fs"%(t2 - t1))
                else:
                    print("\nPlease try again!")
                t3 = time_synchronized()
                print("Time/Image : %.3fs"%(t3 -
t1)+"\n\n")
                ave_time.append(t3 - t1)

            else:
                print("No Prediction!\n\n")
                # Saving output predictions~~~#much
faster if without
    #            # Write results
    #            for *xyxy, conf, cls in reversed(det):
    #                if save_txt:  # Write to file
    #                    xywh =
(xyxy2xywh(torch.tensor(xyxy).view(1, 4)) /
gn).view(-1).tolist()  # normalized xywh
    #                    line = (cls, *xywh, conf) if
opt.save_conf else (cls, *xywh)  # label format
    #                    with open(txt_path + '.txt', 'a') as
f:
    #                        f.write(('%g ' *
len(line)).rstrip() % line + '\n')

    #                if save_img or view_img:  # Add
bbox to image
    #                    label = '%s %.2f' %
(names[int(cls)], conf)
    #                    plot_one_box(xyxy, im0,
label=label, color=colors[int(cls)],
line_thickness=3)

    #        # Stream results
    #        if view_img:
    #            cv2.imshow(str(p), im0)
```

```
    #             if cv2.waitKey(1) == ord('q'):  # q to
quit
    #                 raise StopIteration

    #         # Save results (image with detections)
    #         if save_img:
    #             if dataset.mode == 'images':
    #                 cv2.imwrite(save_path, im0)
    #             else:
    #                 if vid_path != save_path:  # new
video
    #                     vid_path = save_path
    #                     if isinstance(vid_writer,
cv2.VideoWriter):
    #                         vid_writer.release()  # release
previous video writer
    #                     fourcc = 'mp4v'  # output video
codec
    #                     fps =
vid_cap.get(cv2.CAP_PROP_FPS)
    #                     w =
int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDT
H))
    #                     h =
int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGH
T))
    #                     vid_writer =
cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*fourcc), fps, (w, h))
    #                 vid_writer.write(im0)

    # if save_txt or save_img:
    #     s =
f"\n{len(list(save_dir.glob('labels/*.txt')))} labels
saved to {save_dir / 'labels'}" if save_txt else ''
    #     print(f"Results saved to {save_dir}{s}")

    print('Overall time: (%.3fs) || Overall Average
time: (%.3fs) || Overall Min time: (%.3fs) || Overall
Max time: (%.3fs)' % (time.time() -
t0,sum(ave_time)/len(ave_time),min(ave_time),m
ax(ave_time)))# total time
#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
def most_frequent(List):
    return max(set(List), key = List.count)
##########################Kaze##########
################
def kaze_valid(denomination,tt_image):
    path_valid = "/home/bravo/Desktop/LAST SEM
2020 TIP/PD2/testing area/DATASET UV LIGHT
REAL/" # 4 images
    # path_valid = "/home/bravo/Desktop/LAST
SEM 2020 TIP/PD2/testing area/DATASET UV
LIGHT REAL (copy)/" # 80 images

    hey2 = [file for file in
os.listdir(path_valid+denomination)]
    total = []
    average = []
    for i in hey2:
        real = path_valid+denomination+"/"+i
        valid_image = [file for file in os.listdir(real) if
file.endswith(('.png','.jpg','.jpeg','.JPG'))]
        akaze = cv2.ORB_create()
        test = cv2.imread(tt_image,
cv2.IMREAD_GRAYSCALE)
        # test = cv2.imread(tt_image,
cv2.IMREAD_COLOR)
        test = cv2.resize(test, (500, 500))
        kpts2, desc2 =
akaze.detectAndCompute(test, None)

        for img1 in valid_image:
            valid = cv2.imread(real+"/"+img1,
cv2.IMREAD_GRAYSCALE)
```

```
        valid = cv2.resize(valid, (500, 500))
        kpts1, desc1 =
akaze.detectAndCompute(valid, None)
        matcher =
cv2.DescriptorMatcher_create(cv2.DescriptorMat
cher_BRUTEFORCE_HAMMING)
        nn_matches = matcher.knnMatch(desc1,
desc2, 2)
        matched1 = []
        nn_match_ratio = 0.5 # .5

        for m, n in nn_matches:
            if m.distance < nn_match_ratio *
n.distance:

matched1.append(kpts1[m.queryIdx])

        average.append(len(matched1))
        # print(average)

        if sum(average)/len(average) >3:
            print("Matched Features :
%.3f"%(sum(average)/len(average)))
            return denomination

    if sum(average)/len(average) <=3:
        print("Matched Features :
%.3f"%(sum(average)/len(average)))
        return "Counterfeit!"

    elif sum(average)/len(average) >3:
        print("Matched Features :
%.3f"%(sum(average)/len(average)))
        return denomination

    else:
        print("Please try again!")
########################Kaze#########
################
```

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+',
type=str, default='yolov5s.pt', help='model.pt
path(s)')
    parser.add_argument('--img-size', type=int,
default=640, help='inference size (pixels)')
    parser.add_argument('--conf-thres', type=float,
default=0.25, help='object confidence threshold')
    parser.add_argument('--iou-thres', type=float,
default=0.45, help='IOU threshold for NMS')
    parser.add_argument('--device', default='',
help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img',
action='store_true', help='display results')
    parser.add_argument('--save-txt',
action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf',
action='store_true', help='save confidences in
--save-txt labels')
    parser.add_argument('--classes', nargs='+',
type=int, help='filter by class: --class 0, or --class
0 2 3')
    parser.add_argument('--agnostic-nms',
action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment',
action='store_true', help='augmented inference')
    parser.add_argument('--update',
action='store_true', help='update all models')
    parser.add_argument('--project',
default='runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp',
help='save results to project/name')
    parser.add_argument('--exist-ok',
action='store_true', help='existing project/name
ok, do not increment')
```

```
    parser.add_argument('--source', type=str,
default='data/images', help='source')  # file/folder,
0 for webcam
    opt = parser.parse_args()
    print(opt)

    with torch.no_grad():
        if opt.update:  # update all models (to fix
SourceChangeWarning)
            for opt.weights in ['yolov5s.pt',
'yolov5m.pt', 'yolov5l.pt', 'yolov5x.pt']:
                detect()
                strip_optimizer(opt.weights)
        else:
            detect()
```

Appendix F. Computation for Tradeoff Analysis Capturing Device and Algorithm Rank

1. Trade Off Analysis Capturing Device and Yolov4 Rank

Trade-Off Analysis YOLOV4												
DESIGN CRITERIA	UNIT	IMPORTANCE FACTOR	iPhone SE		Xiaomi Redmi k30 Pro		Huawei Nova 2i		Honor Play		Honor 8X	
			VALUE	RANK	VALUE	RANK	VALUE	RANK	VALUE	RANK	VALUE	RANK
ACCURACY	%	40	92.33	4.73	94.33	4.83	95.2	4.87	97.67	5.00	61.33	3.14
SPEED	ms	30	47.59	3.77	35.82	5	47.47	3.78	44.64	4.02	47.64	3.76
COST	Php	30	8,809.57	5.00	42,209.57	1.05	9,609.57	4.59	10,109.57	4.36	28,509.57	1.55

IPhone SE

- Accuracy = 1 - (((97.67 - 92.33)/97.67) * 5) = **4.73**
- Speed = 1 - (((47.59 - 35.82)/47.59) * 5) = **3.77**
- Cost = **5**

Xiaomi Redmi k30 Pro

- Accuracy = 1 - (((97.67 - 94.33)/97.67) * 5) = **4.83**
- Speed = **5**
- Cost = 1 - (((42,209.57 - 8,809.57) / 42,209.57) * 5) = **1.05**

Huawei Nova 2i

- Accuracy = 1 - (((97.67 - 95.2)/97.67) * 5) = **4.87**
- Speed = 1 - (((47.47 - 35.82)/47.47) * 5) = **3.78**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**

Honor Play

- Accuracy = **5**
- Speed = 1 - (((44.64 - 35.82)/44.64) * 5) = **4.02**
- Cost = 1 - (((10109.57 - 8,809.57) / 10109.57) * 5) = **4.36**

Honor 8X

- Accuracy = 1 - (((97.67 - 61.33)/97.67) * 5) = **3.14**
- Speed = 1 - (((47.64 - 35.82)/47.64) * 5) = **3.76**
- Cost = 1 - (((28509.57 - 8,809.57)/28509.57) * 5) = **1.55**

2. Trade Off Analysis Capturing Device and Yolov5 Rank

Trade-Off Analysis Yolov5												
DESIGN CRITERIA	UNIT	IMPORTANCE FACTOR	iPhone SE		Xiaomi Redmi k30 Pro		Huawei Nova 2i		Honor Play		Honor 8X	
			VALUE	RANK	VALUE	RANK	VALUE	RANK	VALUE	RANK	VALUE	RANK
ACCURACY	%	40	80	4.59	72	4.13	87.33	5.00	86.334	4.95	58	3.33
SPEED	ms	30	12.6	5	12.91	4.88	12.78	4.93	13.14	4.8	12.69	4.97
COST	Php	30	8,809.57	5.00	42,209.57	1.05	9,609.57	4.59	10,109.57	4.36	28,509.57	1.55

IPhone SE

- Accuracy = 1 - (((87.33 - 80)/87.33) * 5) = **4.59**
- Speed = **5**
- Cost = **5**

Xiaomi Redmi k30 Pro

- Accuracy = 1 - (((87.33 - 72)/87.33) * 5) = **4.13**
- Speed = 1 - (((12.78 - 12.6)/12.78) * 5) = **4.8**
- Cost = 1 - (((42,209.57 - 8,809.57) / 42,209.57) * 5) = **1.05**

Huawei Nova 2i

- Accuracy = **5**
- Speed = 1 - (((12.91 - 12.6)/12.91) * 5) = **4.88**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**

Honor Play

- Accuracy = 1 - (((87.33 - 86.334)/87.33) * 5) = **4.95**
- Speed = 1 - (((13.14 - 12.6)/13.14) * 5) = **4.88**
- Cost = 1 - (((10109.57 - 8,809.57)/ 10109.57) * 5) = **4.36**

Honor 8X

- Accuracy = 1 - (((87.33 - 58)/87.33) * 5) = **3.33**
- Speed = 1 - (((12.69 - 12.6)/12.69) * 5) = **4.97**
- Cost = 1 - (((28509.57 - 8,809.57) / 28509.57) * 5) = **1.55**

3. Trade Off Analysis Capturing Device and EfficientDet Rank

Trade-Off Analysis EfficientDet												
DESIGN CRITERIA	UNIT	IMPORTANCE FACTOR	iPhone SE		Xiaomi Redmi k30 Pro		Huawei Nova 2i		Honor Play		Honor 8X	
			VALUE	RANK	VALUE	RANK	VALUE	RANK	VALUE	RANK	VALUE	RANK
ACCURACY	%	40	87	4.63	90	4.79	94	5.00	92	4.89	47.33	2.52
SPEED	ms	30	61.1	4.86	61.4	4.83	59.3	5	61.13	4.86	61.17	4.85
COST	Php	30	8,809.57	5.00	42,209.57	1.05	9,609.57	4.59	10,109.57	4.36	28,509.57	1.55

IPhone SE

- Accuracy = 1 - (((94 - 87)/94) * 5) = **4.63**
- Speed = 1 - (((61.1 - 59.3)/61.1) * 5) = **4.86**
- Cost = **5**

Xiaomi Redmi k30 Pro

- Accuracy = 1 - (((90 - 87)/90) * 5) = **4.79**
- Speed = 1 - (((61.1 - 59.3)/61.1) * 5) = **4.83**
- Cost = 1 - (((42,209.57 - 8,809.57) / 42,209.57) * 5) = **1.05**

Huawei Nova 2i

- Accuracy = **5**
- Speed = **5**
- Cost = 1 - (((9,609.57 - 8,809.57)/9,609.57) * 5) = **4.59**

Honor Play

- Accuracy = 1 - (((92 - 87)/92) * 5) = **4.89**
- Speed = 1 - (((61.13 - 59.3)/61.13) * 5) = **4.86**
- Cost = 1 - (((10109.57 - 8,809.57) / 10109.57) * 5) = **4.36**

Honor 8X

- Accuracy = 1 - (((47.33 - 87)/47.33) * 5) = **2.52**
- Speed = 1 - (((61.17 - 59.3)/61.17) * 5) = **4.85**
- Cost = 1 - (((28509.57 - 8,809.57) / 28509.57) * 5 = **1.55**

Appendix G. Computation for Design Options Trade-off and Sensitivity Analysis

1. Design Option 1 Trade-off and Sensitivity Analysis Overall Ranking

Design Option 1: YOLOv4			
Capturing Device	**Accuracy Rank**	**Speed Rank**	**Cost Rank**
iPhone SE	4.73	3.77	5.00
Xiaomi Redmi	4.83	5	1.05
Huawei Nova 2i	4.87	3.78	4.59
Honor Play	5	4.02	4.36
Honor 8x	3.14	3.76	1.55

iPhone SE:

Trade-off

- Accuracy = 4.73 * 40
- Cost = 5 * 30
- Speed = 3.77 * 30

Sensitivity Analysis 1

- Accuracy = 4.83 * 33.33
- Cost = 5 * 33.33
- Speed = 3.77 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.83 * 60
- Cost = 5 * 20
- Speed = 3.77 * 20

Sensitivity Analysis 3

- Accuracy = 4.83 * 20
- Cost = 5 * 60
- Speed = 3.77 * 20

Sensitivity Analysis 4

- Accuracy = 4.83 * 20
- Cost = 5 * 20
- Speed = 3.77 * 60

Xiaomi Redmi:

Trade-off

- Accuracy = 4.83 * 40
- Cost = 5 * 30
- Speed = 1.05 * 30

Sensitivity Analysis 1

- Accuracy = 4.83 * 33.33
- Cost = 5 * 33.33
- Speed = 1.05 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.83 * 60
- Cost = 5 * 20
- Speed = 1.05 * 20

Sensitivity Analysis 3

- Accuracy = 4.83 * 20
- Cost = 5 * 60
- Speed = 1.05 * 20

Sensitivity Analysis 4

- Accuracy = 4.83 * 20
- Cost = 5 * 20
- Speed = 1.05 * 60

Huawei Nova 2i

Trade-off

- Accuracy = 4.87 * 40
- Cost = 3.78 * 30
- Speed = 4.59 * 30

Sensitivity Analysis 1

- Accuracy = 4.87 * 33.33
- Cost = 3.78 * 33.33
- Speed = 4.59 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.87 * 60
- Cost = 3.78 * 20
- Speed = 4.59 * 20

Sensitivity Analysis 3

- Accuracy = 4.87 * 20
- Cost = 3.78 * 60
- Speed = 4.59 * 20

Sensitivity Analysis 4

- Accuracy = 4.87 * 20
- Cost = 3.78 * 20
- Speed = 4.59 * 60

Honor Play

Trade-off

- Accuracy = 5 * 40
- Cost = 4.02 * 30
- Speed = 4.36 * 30

Sensitivity Analysis 1

- Accuracy = 5 * 33.33
- Cost = 4.02 * 33.33
- Speed = 4.36 * 33.33

Sensitivity Analysis 2

- Accuracy = 5 * 60
- Cost = 4.02 * 20
- Speed = 4.36 * 20

Sensitivity Analysis 3

- Accuracy = 5 * 20
- Cost = 4.02 * 60
- Speed = 4.36 * 20

Sensitivity Analysis 4

- Accuracy = 5 * 20
- Cost = 4.02 * 20
- Speed = 4.36 * 60

Honor 8X:

Trade-off

- Accuracy = 3.14 * 40
- Cost = 3.76 * 30
- Speed = 1.55 * 30

Sensitivity Analysis 1

- Accuracy = 3.14 * 33.33
- Cost = 3.76 * 33.33
- Speed = 1.55 * 33.33

Sensitivity Analysis 2

- Accuracy = 3.14 * 60
- Cost = 3.76 * 20
- Speed = 1.55 * 20

Sensitivity Analysis 3

- Accuracy = 3.14 * 20
- Cost = 3.76 * 60
- Speed = 1.55 * 20

Sensitivity Analysis 4

- Accuracy = 3.14 * 20
- Cost = 3.76 * 20
- Speed = 1.55 * 60

2. Design Option 2 Trade-off and Sensitivity Analysis Overall Ranking

Design Option 2: YOLOv5			
Capturing Device	**Accuracy Rank**	**Speed Rank**	**Cost Rank**
iPhone SE	4.59	5	5.00
Xiaomi Redmi	4.13	4.88	1.05
Huawei Nova 2i	5.00	4.93	4.59
Honor Play	4.95	4.8	4.36
Honor 8x	3.33	4.97	1.55

Iphone SE:

Trade-off

- Accuracy = 4.59 * 40
- Cost = 5 * 30
- Speed = 5 * 30

Sensitivity Analysis 1

- Accuracy = 4.59 * 33.33
- Cost = 5 * 33.33
- Speed = 5 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.59 * 60
- Cost = 5 * 20
- Speed = 5 * 20

Sensitivity Analysis 3

- Accuracy = 4.59 * 20
- Cost = 5 * 60
- Speed = 5 * 20

Sensitivity Analysis 4

- Accuracy = 4.59 * 20
- Cost = 5 * 20
- Speed = 5 * 60

Xiaomi Redmi:

Trade-off

- Accuracy = 4.13 * 40
- Cost = 4.88 * 30
- Speed = 1.05 * 30

Sensitivity Analysis 1

- Accuracy = 4.13 * 33.33
- Cost = 4.88 * 33.33
- Speed = 1.05 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.13 * 60
- Cost = 4.88 * 20
- Speed = 1.05 * 20

Sensitivity Analysis 3

- Accuracy = 4.13 * 20
- Cost = 4.88 * 60
- Speed = 1.05 * 20

Sensitivity Analysis 4

- Accuracy = 4.13 * 20
- Cost = 4.88 * 20
- Speed = 1.05 * 60

Huawei Nova 2i

Trade-off

- Accuracy = 5 * 40
- Cost = 4.93 * 30
- Speed = 4.59 * 30
-

Sensitivity Analysis 1

- Accuracy = 5 * 33.33
- Cost = 4.93 * 33.33
- Speed = 4.59 * 33.33

Sensitivity Analysis 2

- Accuracy = 5 * 60
- Cost = 4.93 * 20
- Speed = 4.59 * 20

Sensitivity Analysis 3

- Accuracy = 5 * 20
- Cost = 4.93 * 60
- Speed = 4.59 * 20

Sensitivity Analysis 4

- Accuracy = 5 * 20
- Cost = 4.93 * 20
- Speed = 4.59 * 60

Honor Play

Trade-off

- Accuracy = 4.59 * 40
- Cost = 4.8 * 30
- Speed = 4.36 * 30

Sensitivity Analysis 1

- Accuracy = 4.95 * 33.33
- Cost = 4.8 * 33.33
- Speed = 4.36 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.95 * 60
- Cost = 4.8 * 20
- Speed = 4.36 * 20

Sensitivity Analysis 3

- Accuracy = 4.95 * 20
- Cost = 4.8 * 60
- Speed = 4.36 * 20

Sensitivity Analysis 4

- Accuracy = 4.95 * 20
- Cost = 4.8 * 20
- Speed = 4.36 * 60

Honor 8X:

Trade-off

- Accuracy = x * 40
- Cost = x * 30
- Speed = x * 30

Sensitivity Analysis 1

- Accuracy = 3.33 * 33.33
- Cost = 4.97 * 33.33
- Speed = 1.55 * 33.33

Sensitivity Analysis 2

- Accuracy = 3.33 * 60
- Cost = 4.97 * 20
- Speed = 1.55 * 20

Sensitivity Analysis 3

- Accuracy = 3.33 * 20
- Cost = 4.97 * 60
- Speed = 1.55 * 20

Sensitivity Analysis 4

- Accuracy = 3.33 * 20
- Cost = 4.97 * 20
- Speed = 1.55 * 60

3. Design Option 3: Trade-off and Sensitivity Analysis Overall Ranking

Design Option 1: EfficientDet			
Capturing Device	**Accuracy Rank**	**Speed Rank**	**Cost Rank**
iPhone SE	4.63	4.86	5.00
Xiaomi Redmi	4.79	4.83	1.05
Huawei Nova 2i	5.00	5	4.59
Honor Play	4.89	4.86	4.36
Honor 8x	2.52	4.85	1.55

Iphone SE:

Trade-off

- Accuracy = 4.63 * 40
- Cost = 4.86 * 30
- Speed = 5 * 30

Sensitivity Analysis 1

- Accuracy = 4.63 * 33.33
- Cost = 4.86 * 33.33
- Speed = 5 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.63 * 60
- Cost = 4.86 * 20
- Speed = 5 * 20

Sensitivity Analysis 3

- Accuracy = 4.63 * 20
- Cost = 4.86 * 60
- Speed = 5 * 20

Sensitivity Analysis 4

- Accuracy = 4.63 * 20
- Cost = 4.86 * 20
- Speed = 5 * 60

Xiaomi Redmi:

Trade-off

- Accuracy = 4.79 * 40
- Cost = 4.83 * 30
- Speed = 1.05 * 30

Sensitivity Analysis 1

- Accuracy = 4.79 * 33.33
- Cost = 4.83 * 33.33
- Speed = 1.05 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.79 * 60
- Cost = 4.83 * 20
- Speed = 1.05 * 20

Sensitivity Analysis 3

- Accuracy = 4.79 * 20
- Cost = 4.83 * 60
- Speed = 1.05 * 20

Sensitivity Analysis 4

- Accuracy = 4.79 * 20
- Cost = 4.83 * 20
- Speed = 1.05 * 60

Huawei Nova 2i

Trade-off

- Accuracy = 5 * 40
- Cost = 5 * 30
- Speed = 4.59 * 30

Sensitivity Analysis 1

- Accuracy = 5 * 33.33
- Cost = 5 * 33.33
- Speed = 4.59 * 33.33

Sensitivity Analysis 2

- Accuracy = 5 * 60
- Cost = 5 * 20
- Speed = 4.59 * 20

Sensitivity Analysis 3

- Accuracy = 5 * 20
- Cost = 5 * 60
- Speed = 4.59 * 20

Sensitivity Analysis 4

- Accuracy = 5 * 20
- Cost = 5 * 20
- Speed = 4.59 * 60

Honor Play:

Trade-off

- Accuracy = 4.89 * 40
- Cost = 4.86 * 30
- Speed = 4.36 * 30

Sensitivity Analysis 1

- Accuracy = 4.89 * 33.33
- Cost = 4.86 * 33.33
- Speed = 4.36 * 33.33

Sensitivity Analysis 2

- Accuracy = 4.89 * 60
- Cost = 4.86 * 20
- Speed = 4.36 * 20

Sensitivity Analysis 3

- Accuracy = 4.89 * 20
- Cost = 4.86 * 60
- Speed = 4.36 * 20

Sensitivity Analysis 4

- Accuracy = 4.89 * 20
- Cost = 4.86 * 20
- Speed = 4.36 * 60

Honor 8X:

Trade-off

- Accuracy = 2.52 * 40
- Cost = 4.85 * 30
- Speed = 1.55 * 30

Sensitivity Analysis 1

- Accuracy = 2.52 * 33.33
- Cost = 4.85 * 33.33
- Speed = 1.55 * 33.33

Sensitivity Analysis 2

- Accuracy = 2.52 * 60
- Cost = 4.85 * 20
- Speed = 1.55 * 20

Sensitivity Analysis 3

- Accuracy = 2.52 * 20
- Cost = 4.85 * 60
- Speed = 1.55 * 20

Sensitivity Analysis 4

- Accuracy = 2.52 * 20
- Cost = 4.85 * 20
- Speed = 1.55 * 60

4. **Final: Design Options Trade-off and Sensitivity Analysis Overall Ranking**

DESIGN CRITERIA	YOLOv4		YOLOv5		EfficientDet	
	VALUE	RANK	VALUE	RANK	VALUE	RANK
ACCURACY	92.33	4.91	87.33	4.73	94	5.00
SPEED	47.59	1.35	12.78	5	59.3	1.08
COST	8,809.57	5	9,609.57	4.59	9,609.57	4.59
Overall Ranking		**386.95**		**476.87**		**370.10**

Tradeoff Analysis

YOLOv4:

- Accuracy = 1 - (((94 - 92.33)/94) * 5) = **4.91**
- Speed = 1 - (((47.59 - 12.78)/47.59) * 5) = **1.35**
- Cost = **5**
- Overall Ranking = (4.91*40)+(1.35*30)+(5*30) = 386.95

YOLOv5:

- Accuracy = 1 - (((94 - 87.33)/94) * 5) = **4.73**
- Speed = **5**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (4.73*40) + (5*30)+(4.59*30) = **476.87**

EfficientDet:

- Accuracy = **5**
- Speed = 1 - (((59.3 - 12.78)/59.3) * 5) = **1.08**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (5*40)+(1.08*30)+(4.59*30) = **370.10**

Sensitivity Analysis 1

YOLOv4:

- Accuracy = 1 - (((94 - 92.33)/94) * 5) = **4.91**
- Speed = 1 - (((47.59 - 12.78)/47.59) * 5)
  - = **1.35**
- Cost = **5**
- Overall Ranking = (4.91*33.33)+(1.35*33.33)+(5*33.33) = **375.33**

YOLOv5:

- Accuracy = 1 - (((94 - 87.33)/94) * 5) = **4.73**
- Speed = **5**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (4.73*33.33) + (5*33.33) + (4.59*33.33) = **477.26**

EfficientDet:

- Accuracy = **5**
- Speed = 1 - (((59.3 - 12.78)/59.3) * 5) = **1.08**

- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (5*33.33) + (1.08*33.33)+(4.59*33.33) = **355.63**

Sensitivity Analysis 2

YOLOv4:

- Accuracy = 1 - (((94 - 92.33)/94) * 5) = **4.91**
- Speed = 1 - (((47.59 - 12.78)/47.59) * 5) = **1.35**
- Cost = **5**
- Overall Ranking = (4.91*60) + (1.35*20)+(5*20) = **421.67**

YOLOv5:

- Accuracy = 1 - (((94 - 87.33)/94) * 5) = **4.73**
- Speed = **5**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (4.73*60) + (5*20)+(4.59*20) = **475.55**

EfficientDet:

- Accuracy = **5**
- Speed = 1 - (((59.3 - 12.78)/59.3) * 5) = **1.08**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (5*60) + (1.08*20) + (4.59*20) = **413.40**

Sensitivity Analysis 3

YOLOv4:

- Accuracy = 1 - (((94 - 92.33)/94) * 5) = **4.91**
- Speed = 1 - (((47.59 - 12.78)/47.59) * 5) = **1.35**
- Cost = **5**
- Overall Ranking = (4.91*20) + (1.35*60)+(5*20) = **279.22**

YOLOv5:

- Accuracy = 1 - (((94 - 87.33)/94) * 5) = **4.73**
- Speed = **5**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (4.73*20) + (5*60) + (4.59*20) = **486.38**

EfficientDet:

- Accuracy = **5**
- Speed = 1 - (((59.3 - 12.78)/59.3) * 5) = **1.08**
- Cost = 1 - (((9,609.57 - 8,809.57) /9,609.57) * 5) = **4.59**
- Overall Ranking = (5*20) + (1.08*60)+(4.59*20) = **256.6**

Sensitivity Analysis 4

YOLOv4:

- Accuracy = 1 - (((94 - 92.33)/94) * 5) = **4.91**
- Speed = 1 - (((47.59 - 12.78)/47.59) * 5) = **1.35**
- Cost = **5**
- Overall Ranking = (4.91*20) + (1.35*20) + (5*60) = **425.22**

YOLOv5:

- Accuracy = 1 - (((94 - 87.33)/94) * 5) = **4.73**
- Speed = **5**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (4.73*20) + (5*20)+(4.59*60) = **469.98**

EfficientDet:

- Accuracy = **5**
- Speed = 1 - (((59.3 - 12.78)/59.3) * 5) = **1.08**
- Cost = 1 - (((9,609.57 - 8,809.57) / 9,609.57) * 5) = **4.59**
- Overall Ranking = (5*20) + (1.08*20) + (4.59*60) = **397**

References

[1] "What Do Blind People See? Facts and Research", Healthline, 2020. [Online]. Available: https://www.healthline.com/health/eye-health/what-do-blind-people-see#misconceptions. [Accessed: 05- Feb- 2020]

[2] 2020. [Online]. Available: https://www.doh.gov.ph/node/10735. [Accessed: 20- Jan- 2020]

[3] Hellingman, J. (2011). Current Philippine Banknotes. Retrieved 2 January 2021, from https://www.bohol.ph/article276.html

[4]"How Blind People Identify Paper Money - Blind Coin Collector", Blind Coin Collector, 2020. [Online]. Available: https://blindcoincollector.com/2019/02/18/how-blind-people-identify-paper-money/. [Accessed: 21- Jan- 2020]

[5]"TapTapSee - Blind and Visually Impaired Assistive Technology - powered by CloudSight.ai Image Recognition API", Taptapseeapp.com, 2020. [Online]. Available: https://taptapseeapp.com/. [Accessed: 20- Jan- 2020]

[6]"Home | Cash Reader Banknote reader application for the blind", Cash Reader Banknote reader application for the blind, 2020. [Online]. Available: https://cashreader.app/en/. [Accessed: 20- Jan- 2020]

[7] Ltd, M. (2021). Complete Guide To Image Sensor Pixel Size. ePHOTOzine. Retrieved 2 January 2021, from https://www.ephotozine.com/article/complete-guide-to-image-sensor-pixel-size-29652.

[8] Garden, H., HowStuffWorks, Tech, Electronics, Photography, & Cameras. (2021). How Digital Cameras Work. HowStuffWorks. Retrieved 2 January 2021, from https://electronics.howstuffworks.com/cameras-photography/digital/digital-camera3.htm.

[9] What is bill of materials (BOM)? - Definition from WhatIs.com (techtarget.com)

[10] Factors affecting CPU performance - Computer systems - AQA - GCSE Computer Science Revision - AQA - BBC Bitesize: www.bbc.co.uk/bitesize/guides/z7qqmsg/revision/5

[11] Griffey, J., 2021. Chapter 3: Personal Multimedia Devices For Capturing And Consuming. [online] Journals.ala.org. Available at: <https://journals.ala.org/index.php/ltr/article/view/4767/5695> [Accessed 2 January 2021].

[12] Krapf, R., 2021. Image Capturing Device. US9706125B2.

[13] Gallagher, J. (2021). What is a Processor?. Career Karma. Retrieved 2 January 2021, from https://careerkarma.com/blog/what-is-a-processor/.

[14] The Electrical Engineering Handbook by Peter Y.K.CheungGeorge A.Constantinides, WayneLuk. (2021) (p. Multiprocessor).

[15] Community - Competitive Programming - Competitive Programming Tutorials - The Importance of Algorithms. Topcoder.com. (2021). Retrieved 2 January 2021, from https://www.topcoder.com/community/competitive-programming/tutorials/the-importance-of-algorithms/.

[16] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. (2012). “Imagenet Classification with Deep Convolutional Neural Networks.” In Advances in Neural Information Processing Systems, 1097–1105.

[17] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. (2017). “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” IEEE transactions on pattern analysis and machine intelligence, 39(6): 1137–49.

[18] Introduction To Feature Detection And Matching. (2021). Retrieved 4 January 2021, from https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d

[19] Papers with Code - COCO test-dev Benchmark (Object Detection). Paperswithcode.com. (2021). Retrieved 2 January 2021, from https://paperswithcode.com/sota/object-detection-on-coco.

[20] Maithani, M. (2021). Guide to YOLOv5 for Real-Time Object Detection - Analytics India Magazine. Analytics India Magazine. Retrieved 2 January 2021, from https://analyticsindiamag.com/YOLOv5/.

[21] PyTorch. Pytorch.org. (2021). Retrieved 2 January 2021, from https://pytorch.org/hub/ultralytics_YOLOv5/.

[22]kaze. (2021). Retrieved 5 January 2021, from

http://www.robesafe.com/personal/pablo.alcantarilla/kaze.html#:~:text=KAZE%20Features%20is%20a%20

novel,in%20a%20nonlinear%20scale%20space.&text=By%20means%20of%20nonlinear%20diffusion,image%20in%20the%20scale%20space.

[23] Roos, D., Shiguemori, E., & Lorena, A. (2015). Comparing ORB and AKAZE for visual odometry of unmanned aerial vehicles. Retrieved 4 January 2021, from http://www.epacis.net/ccis2016/papers/paper_121.pdf

[24] Tyagi, D. (2021). What's new in YOLOv4?. Medium. Retrieved 2 January 2021, from https://towardsdatascience.com/whats-new-in-YOLOv4-323364bb3ad3#:~:text=YOLO%20is%20short%20for%20You,classes%20and%20even%20unseen%20classes.

[25] Salowetz, J. (2020). How to Train A Custom Object Detection Model with YOLO v5. Medium. Retrieved 2 January 2021, from https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-YOLO-v5-917e9ce13208.

[26] Papers with Code - EfficientNet Explained. Paperswithcode.com. (2021). Retrieved 2 January 2021, from https://paperswithcode.com/method/efficientnet.

[27] Colaboratory – Google. Research.google.com. (2021). Retrieved 2 January 2021, from https://research.google.com/colaboratory/faq.html.

Other references:

[1] Hsiung, C. (2021). tzutalin/labelImg. GitHub. Retrieved 2 January 2021, from https://github.com/tzutalin/labelImg.

[2] Google Colaboratory. Colab.research.google.com. (2021). Retrieved 2 January 2021, from https://colab.research.google.com/drive/1zZtW_Vt177tvu-7BbYi9cTZip5UAky6Z?usp=sharing.

[3] Google Colaboratory. Colab.research.google.com. (2021). Retrieved 2 January 2021, from https://colab.research.google.com/drive/1CBuBqM7iswZrGhX9dO2pEwwHCZzvGsGh?usp=sharing.

[4] A. S. Alon, R. M. Dellosa, N. U. Pilueta, H. D. Grimaldo and E. T. Manansala, "EyeBill-PH: A Machine Vision of Assistive Philippine Bill Recognition Device for Visually Impaired," 2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 2020, pp. 312-317, DOI: 10.1109/ICSGRC49013.2020.9232557.