electronics

MDPI

Article

YOLO-Based Object Detection for Separate Collection of Recyclables and Capacity Monitoring of Trash Bins

Aria Bisma Wahyutama and Mintae Hwang *

Department of Information and Communication Engineering, Changwon National University, Changwon 51140, Korea; wahyutama@gs.cwnu.ac.kr
* Correspondence: mthwang@cwnu.ac.kr; Tel.: +82-55-213-3832

Abstract: This study describes the development of a smart trash bin that separates and collects recyclables using a webcam and You Only Look Once (YOLO) real-time object detection in Raspberry Pi, to detect and classify these recyclables into their correct categories. The classification result rotates the trash bin lid and reveals the correct trash bin compartment for the user to throw away trash. The performance of the YOLO model was evaluated to measure its accuracy, which was 91% under an optimal computing environment and 75% when deployed in Raspberry Pi. Several Internet of Things hardware, such as ultrasonic sensors for measuring trash bin capacity and GPS for locating trash bin coordinates, are implemented to provide capacity monitoring controlled by Arduino Uno. The capacity and GPS information are uploaded to Firebase Database via theESP8266 Wi-Fi module. To deliver the capacity monitoring feature, the uploaded trash bin capacity information is displayed on the mobile application in the form of a bar level developed in the MIT App Inventor for the user to quickly take action if required. The system proposed in this study is intended to be implemented in a rural area, where it can potentially solve the recyclable waste separation problem.

Keywords: smart trash bin; separates and collects recyclable; YOLO real-time object detection; IoT hardware; capacity monitoring; mobile application

check for updates

Citation: Wahyutama, A.B.; Hwang, M. YOLO-Based Object Detection for Separate Collection of Recyclables and Capacity Monitoring of Trash Bins. *Electronics* **2022**, *11*, 1323. https://doi.org/10.3390/electronics11091323

Academic Editor: George A. Tsihrintzis

Received: 17 March 2022
Accepted: 19 April 2022
Published: 21 April 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Waste management has become a major global problem. As reported in [1], there are 1.2 million t of illegally abandoned waste across South Korea, according to the Ministry of Environment, owing to a series of events in recent years that have disrupted the waste management system. According to the data presented in [2], the total waste generated continues to increase proportionately with population and economic growth. Furthermore, in 2019, Municipal Solid Waste (MSW) generation exceeded an estimated 705 million t, which is considered an alarming number.

Waste separation is highly important because good separation enhances the quality of recyclables, reduces MSW, and optimizes incineration [3]. Moreover, separating recyclables is dictated by the laws of several countries; hence, it is illegal to abandon recyclables without separating them. Despite the importance of separating waste, there are still several people who do not care to adhere to this rule. There are three approaches to achieving better waste management: reducing (buying only highly necessary items), reusing (using a used item if possible), and recycling (turning garbage into something useful, instead of releasing it on a landfill) [4]. This paper presents the implementation of a smart trash bin to help address the waste problem, specifically recyclability, by adopting You Only Look Once (YOLO) object detection.

YOLO object detection is installed in a Raspberry Pi with a webcam to read an image as the input, and then fed to the YOLO object detection to be classified. Once the object has been classified, the trash bin lid rotates to reveal the correct compartment for the detected recyclable. Moreover, to provide capacity monitoring as an extra feature, a set of Internet of

Things (IoT) hardware was also implemented with Arduino Uno to control the connected hardware. The IoT hardware comprises an ultrasonic sensor attached below the trash bin lid, which measures the trash bin capacity based on the distance, and a GPS to retrieve the trash bin location information. The data from the ultrasonic and GPS are gathered by an Arduino Uno and then uploaded to the cloud DB (database), which, in this case, is the Firebase Database, using the ESP8266 Wi-Fi module. Once uploaded, the mobile application can retrieve and display the data in actual time, as a capacity-monitoring feature.

The objective of this study is to develop a smart trash bin for recyclables that can automatically detect their categories and reveal the correct compartment for the user to throw them in. In addition, this study attempts to measure and monitor the trash bin capacity in real time, obtain the trash bin location using GPS, and upload all information to the Firebase Database to be displayed on the mobile application. This capacity and location coordinate information can be beneficial to two users: the public and garbage collector officers, and can be considered part of the recycling management system. The garbage collector uses the application as a warning system for them to quickly empty the trash bin before it becomes 100% full. For the public, it would show which trash bin is available with low capacity, thereby preventing them from throwing the trash into a full trash bin. This overall system is intended to help address the recyclables waste separation problem in rural areas by providing a trash bin that can automatically detect and classify recyclables.

2. Related Works

YOLO is an approach in machine learning that adopts a regression problem to spatially separate the bounding boxes of a certain object and associated class probabilities that achieve object classification. YOLO can classify objects up to 155 frames per second (FPS) in actual time while achieving twice the mean average precision (mAP) of other object classifiers. YOLO is a single convolutional network that simultaneously predicts multiple bounding boxes on multiple objects and then generates a class probability for that object [5]. YOLO has several advantages over other CNN algorithms in terms of object detection. It is a unified object detection model that is easy to construct and train in correspondence with its loss function, resulting in the YOLO training of the entire model in parallel. Moreover, YOLO is better at generalising object representations, making it an advanced object detection method [6]. The architecture of YOLO is presented in Figure 1.

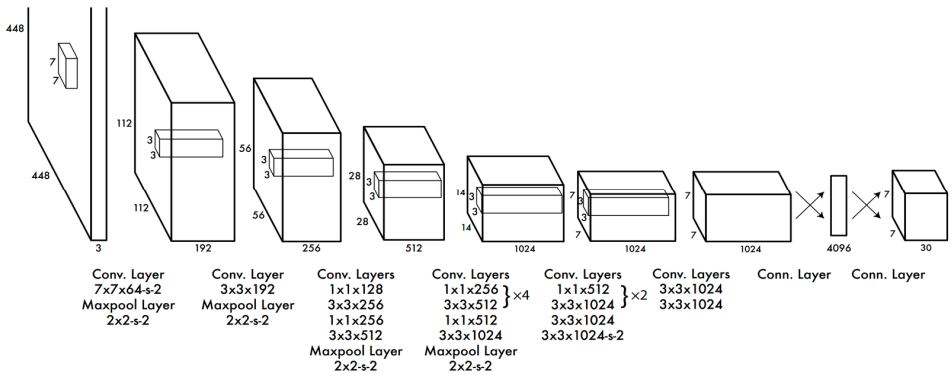


Figure 1. YOLO algorithm architecture.

The Internet of Things (IoT), often known as the Internet of everything, is a paradigm for technology envisioned as a global network of communication that enables machines and devices to interact with each other [7].

Previous research related to smart trash bins has been conducted, as presented in [8]. However, the aforementioned works only monitored the capacity of trash bins using ultrasonic and motion sensors without any recyclable separation collection, which is the primary improvement of this study. The capacity information is delivered only when the trash bin is full, via SMS to the user and a speaker attached to the trash bin; however,

this study only delivers capacity information in real time. Other similar works classify recyclables using Machine or Deep Learning in [9–12]. In [9], the authors compared the trash classification using five Deep Learning models: DenseNet121, DenseNet169, InceptionResnetV2, and MobileNet. The study exhibited optimal results, with the accuracy of the models somewhere between 89% and 95%. In [10], a classification model called WasteNet, which is based on a Convolutional Neural Network, resulted in an accuracy of 97%. Another piece of research [11] adopted a faster R-CNN for classification and exhibited accuracy results from 70% to 84%. Similarly, Ref. [12] employed R-CNN to classify trash with an accuracy score of 64%. All the aforementioned studies adopted the TrashNet dataset, except for [12].

Among several previous studies, this research presents key differences and novelties. This study employs Raspberry Pi to deploy and run the model, which has a considerably lower performance than a full-size personal computer. Although [10] used an NVIDIA Jetson Nano to deploy the model, it is quite difficult and expensive to obtain the device compared to Raspberry Pi. Another original work in this paper is that by including a capacity-monitoring capability, extra information would be provided to the user, in addition to classifying recyclables.

Monitoring is an activity that involves observing a system using a collection of sensors, which can be performed either periodically (samples are taken at a regular interval) or reactively (an external event triggers activation of sensor reading to help diagnose the problem that occurs and solves it). Monitoring activities can also be adopted as additional information to decide on an action by a stakeholder or management. After the action is performed, the sensor data can be read again to check whether the action has improved the situation, worsened the situation, or solved the problem [13].

Firebase is a combination of Google's numerous cloud services, such as instant messaging, user authentication, actual databases, storage, and hosting. The Firebase Database is a cloud-hosted database that stores data in a JSON format and can be synchronised in actual time to every connected client [14,15]. Therefore, based on this characteristic, the Firebase Database was chosen to store all of the data in the monitoring feature in this study and deliver real-time data communication between IoT hardware and applications.

3. System Architecture and Design

3.1. System Architecture and Flowchart

The architecture of the proposed system can be categorised into two parts: hardware and software. The hardware comprises a webcam for obtaining video input, a Raspberry Pi 4 manufactured by Raspberry Pi Foundation in Pencoed, South Wales, United Kingdom, an Arduino Uno manufactured by Arduino in Scarmagno, Italy, an ultrasonic sensor manufactured by Kuongshun Electronic in Shenzhen, China, a servo manufactured by TowerPro in Florida, United States, a GPS module manufactured by Ublox in Surendranagar, India, and an ESP8266 Wi-Fi module manufactured by Espressif Systems in Shanghai, China. This system chooses Raspberry Pi as the main computing device mainly because of its size, which is sufficiently small to be installed in a regular-size trash bin. Moreover, Raspberry Pi is considered to be easier to find and more affordable in terms of price while carrying sufficient computing performance compared to other alternatives such as NVIDIA Jetson Nano, Odroid XU4, and Asus Tinker Board. The software comprises the Firebase Database for storing the data and a mobile application for displaying information regarding the capacity of the trash bin. The Raspberry Pi 4, webcam, and servo are used for object detection and classification, while the ultrasonic sensor, GPS, and ESP8266 Wi-Fi are adopted for capacity monitoring purposes. The system architecture is illustrated in Figure 2.

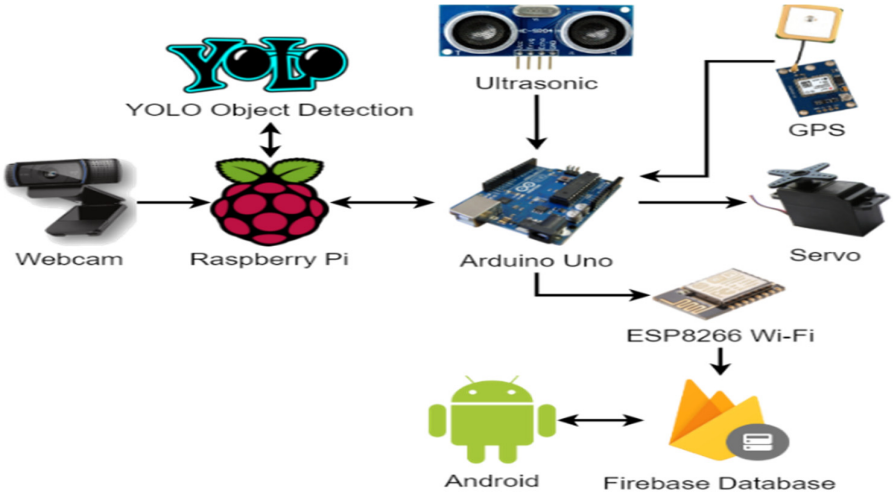


Figure 2. System architecture.

To further illustrate the system, an End-to-End (E2E) diagram is presented in Figure 3. The collection of trash bins spread around the rural area uploads the capacity and GPS location information to the Firebase Database, then, the mobile application retrieves and displays the uploaded information. The public can use the mobile application to find and choose the nearest empty trash bin to throw their recyclables, and the garbage collector officer can monitor the trash bin that is full or nearly full. Hence, they can immediately empty the trash bin.

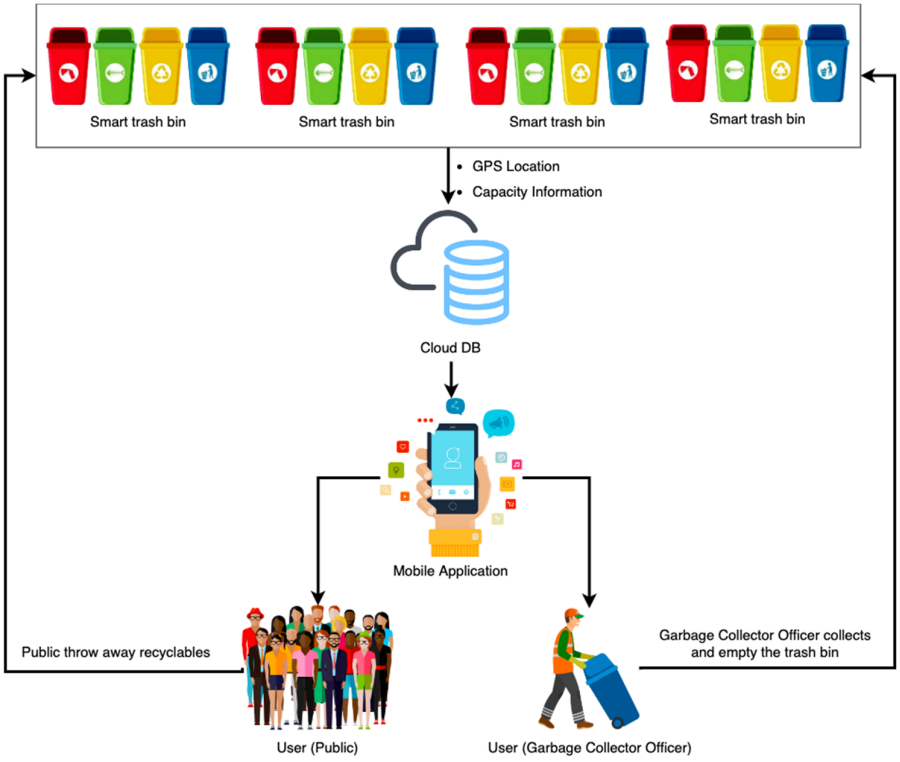


Figure 3. End-to-end diagram.

The flowchart starts from the webcam to retrieve the video image that feeds it to the Raspberry Pi, where YOLO object detection is deployed. When a user places recyclables in front of the camera, it triggers YOLO to detect and classify the object. After the object has been classified, the result is sent to Arduino Uno, where it rotates the servo to a

certain angle according to the classification result. If YOLO cannot classify the object or it results in an error, it will automatically rescan the object to trigger YOLO again, until it can classify the object. Servo rotation causes the trash bin to reveal a compartment dedicated to a particular recyclable classification. An ultrasonic sensor was attached above the compartment to measure the distance from the top of the compartment to the topmost object inside the compartment. Therefore, a shorter distance implies that the trash bin is fuller and vice versa.

If the distance measurement value from the ultrasonic sensor gathered by Arduino Uno exhibits a variation compared to the previous measurement value, it is sent to the ESP8266 Wi-Fi module to upload the data to the Firebase Database. However, if there is no variation compared to the previous measurement value, the Arduino will ignore it and will not send the data to the ESP8266 Wi-Fi module. The uploaded data are retrieved by a mobile application to monitor the trash bin capacity in real time. The flowchart of the system is presented in Figure 4.

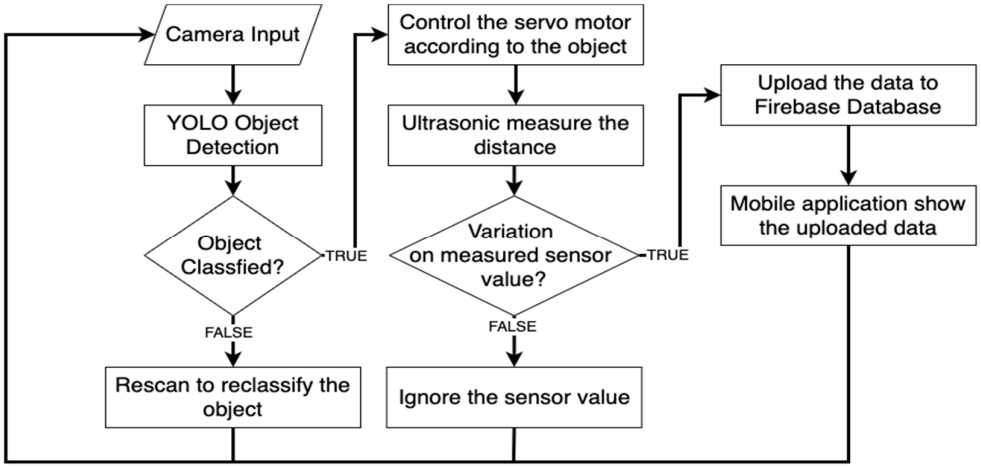


Figure 4. System flowchart.

3.2. YOLO Model

The first step in preparing the YOLO model is to gather numerous images of recyclable trash as the initial dataset, which can be achieved by downloading the image pack from a dataset site such as Kaggle or downloading the images from a search engine such as Google, although Google is not advisable because it takes a long time to gather sufficient data for the model. Alternatively, a web crawling technique can also be adopted to gather images from the internet.

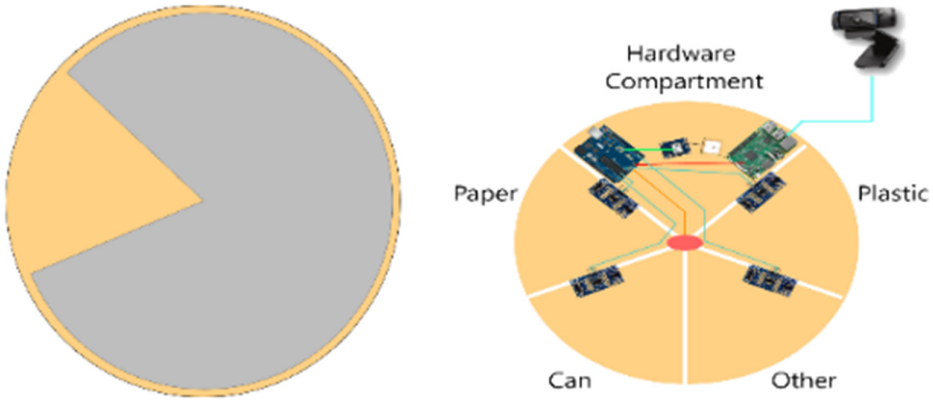
The images that were collected were separated into two sets: the training set for the learning process of the model and the validation set for model verification. The separation ratio can vary in each scenario; however, in this study, the separation ratios for the training and validation sets were 80% and 20%, respectively. To improve the final accuracy and efficiency of the trained YOLO model, it is necessary to inflate the image by implementing data augmentation technology. Data augmentation can be performed using two methods: white-box (conventional) and black-box (using deep neural networks) methods [16]. This study adopted the conventional method because it is widely used. Data augmentation inflates the learning data by randomly transforming all of the images in the training set in various ways, such as rotating, enlarging, inverting, and adding noise, to make the model have a much better understanding of the dataset.

After data augmentation has been applied to the training set, labelling all the images is necessary because the model learns an image–label pair to create a prediction. After the labelling was complete, YOLO custom model learning was performed. Currently, the latest version of YOLO object detection is version 5; however, in this study, the version used is the tiny model of YOLO version 4. This is because the tiny YOLO version 4 model

is a lightweight version of the full-size YOLO, which is highly suitable for small, low-end boards, such as Raspberry Pi. The tiny YOLO version 4 model is imported into Google Colab, which is a web-based code editor built for the interactive python notebook file (ipynb), similar to Jupyter Notebook, which is provided by Google for educational and scientific research purposes. Although there are some restrictions in terms of usage time, Google Colab can be run using either a tensor processing unit (TPU) or a graphical processing unit (GPU) for free, and already has a basic environment library installed.

3.3. Hardware Design

The trash bin is circular in shape and comprises two major parts: the trash bin body and lid. The trash bin body has five separators that divide it into five compartments. Four compartments are assigned for recyclable trash (paper, cans, plastic, and others) with one ultrasonic sensor located at the top of each compartment, while the last compartment is where other hardware, such as Arduino and Raspberry Pi, are located. The trash bin lid also has a circular shape with a triangular cut-out to create an opening at the lid, making the shape similar to that of a Pacman, allowing the user to place the recyclable trash into the bin. The trash bin lid rotates, owing to the servo fitted in the middle of the trash bin body (indicated with an oval shape in Figure 5), to align the trash bin lid cut-out with the correct compartment, according to the result of YOLO object detection with the default position of the cut-out in the hardware compartment. The hardware design of the proposed system is illustrated in Figure 5.



(**a**) (**b**)

Figure 5. Hardware design for the trash bin lid (**a**) and trash bin body (**b**).

As aforementioned, capacity monitoring requires an ultrasonic sensor to measure the distance from the top of the trash bin lid to the top-most object inside the compartment, a GPS module to locate the location coordinate, and an ESP8266 Wi-Fi module to upload the reading data to the Firebase Database. The hardware requirement list is presented in Table 1, and the diagram schematic for capacity monitoring is illustrated in Figure 6.

Table 1. Hardware requirement.

Hardware	Description
Arduino Uno	Two Arduino Unos, one for controlling the servo and another for controlling ultrasonic, GPS, and ESP8266
ESP8266 Wi-Fi Module	To upload gathered data to Firebase Database
GPS (NEO-6M)	To locate trash bin coordinate information
Ultrasonic (HC-R04)	To measure the distance from the lid to the recyclable trash
Raspberry Pi 4B+	To run the YOLO object detection
Servo Motor (MG946R)	To rotate the trash bin lid
USB Camera (WNA-PC200)	To capture the recyclable trash video image

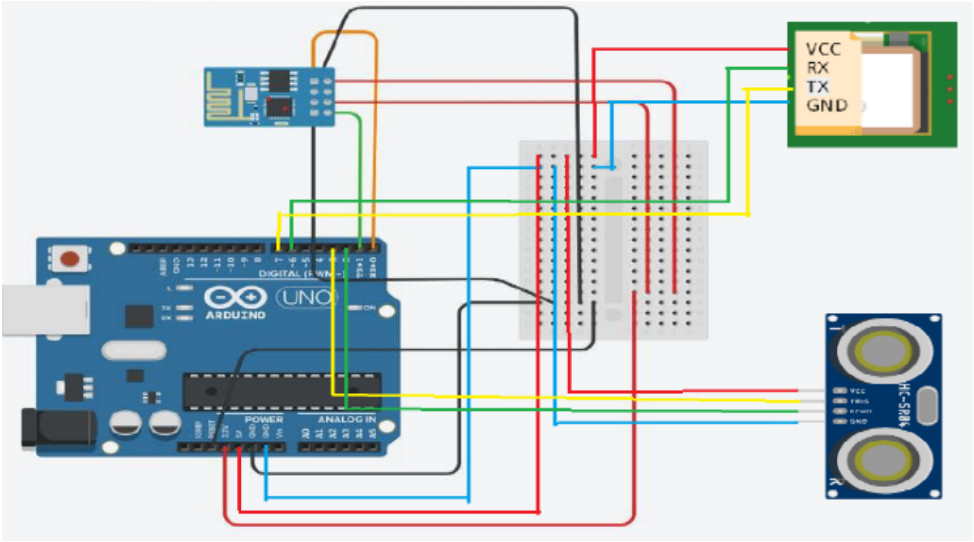


Figure 6. Schematic diagram for capacity monitoring.

4. System Implementation

4.1. YOLO Implementation

The development environment and library used for the YOLO learning and implementation are presented in Table 2.

Table 2. Development Environment Tools.

Tools	Description
YOLO Mark	For labelling the object on the boundary box
Selenium	Web crawling for gathering images as a dataset
Keras	Data augmentation
Google Colab	Code editor to code custom YOLO model

The images for the dataset were collected using three methods: manual downloading from a search engine, web crawling, and downloading the dataset from the dataset provider website. The first method involves downloading from a search engine, and, in this case, Google, Naver, and Bing are employed. This method is conducted by visiting each search engine's homepage and typing keywords in the search bar. Once the results appear, the images are curated and successively manually downloaded to ensure their appropriateness. In general, the dataset gathered using this method comprised 30 images from each search engine for each category, resulting in a total of 120 images. Although this method is highly trivial, it is time-consuming because the model requires hundreds or thousands of images to learn the dataset properly. Therefore, this study does not rely solely on this method. The second method automatically downloads images through programming using a framework called Selenium. Selenium is an open-source framework, primarily used for automating web applications for several purposes, that supports multiple programming languages [17]. Essentially, this study adopted selenium to download all the images that exist on a certain website program, using the Python programming language. The last method was adopting a pre-existing and pre-configured dataset provided by a community on a website, namely, Kaggle, which holds several different datasets for numerous machine or deep learning purposes. This study used the garbage classification dataset provided by cchangcs [18]. With the combination of all methods, the total dataset included 1000 images, with 250 images of paper, 250 images of plastic, 250 images of can, and 250 images of other recyclables.

The collected images were then divided into a training set and a validation set with the aforementioned ratio, where the augmented data were applied to the training set using

the image data generator library provided by Keras. Several transformations were applied to all the images in the training set, such as rescale, rotate, shift, and zoom, resulting in an increase in the number of images to 3000 images in the training set. An example of data augmentation is illustrated in Figure 7.

Figure 7. Data augmentation.

The parameter configurations are presented in Table 3.

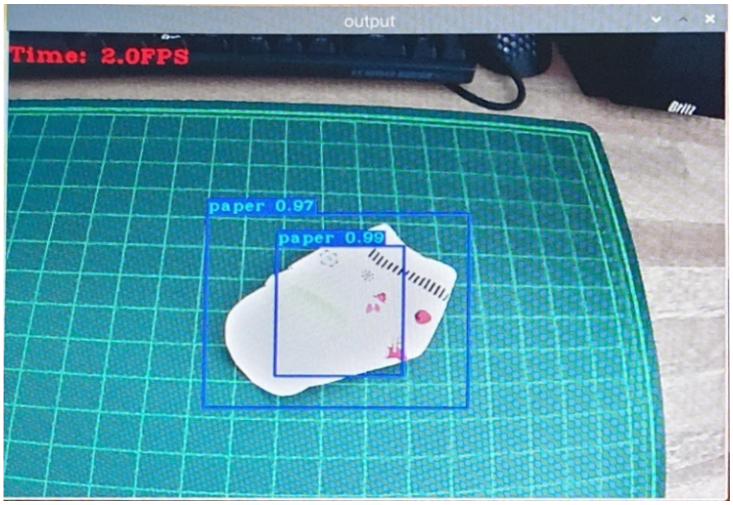
Table 3. Parameter configuration.

Neural Net	Convolutional	YOLO
Batch = 64 Subdivisions = 16 Width = 320 Height = 320 Channels = 3 Momentum = 0.9 Decay = 0.0005 Angle = 0 Saturation = 1.5 Exposure = 1.5 Hue = 0.1 learning_rate = 0.00261 burn_in = 1000 max_batches = 9000 policy = steps steps = 7200, 8100 scales = 0.1, 0.1	Size = 1 Stride = 1 Pad = 1 Filters = 24 Activation = linear	Mask = 3, 4, 5 Anchors = 10, 14, 23, 27, 37, 59, 81, 82, 135, 169, 344, 319 Classes = 3 num = 6 jitter = 0.3 scale_x_y = 1.05 cls_normalizer =1.0 iou_normalizer =0.07 iou_loss = ciou ignore_thresh = 0.7 truth_thresh = 1 random = 0 resize = 1.5 rms_kind = greedynms beta_nms = 0.6

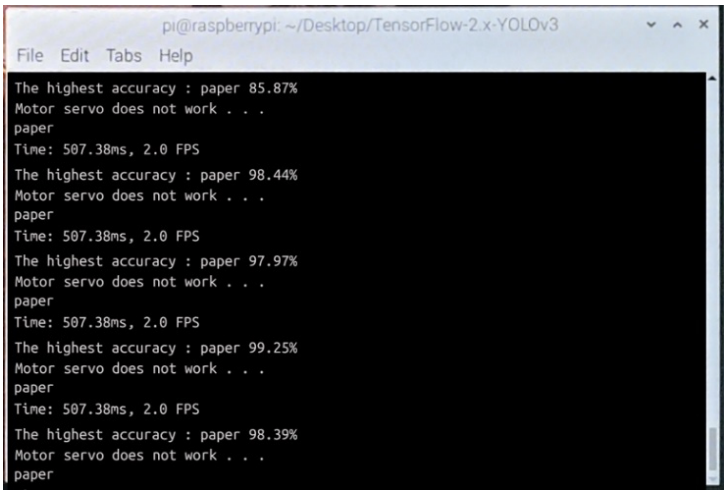
In the process of learning the YOLO model, a label that correctly corresponds to each image in the training set is required. This is crucial because the YOLO model learns the distribution of each image and connects the distribution to the assigned label. Therefore, whenever a new image is presented, the model can predict the label according to what it had learned previously. Before executing the model's learning process, it is necessary to open the configuration file and modify certain parameters such as the number of batches, size of the image, number of channels, and learning rate to optimise the learning process according to the situation. The configuration file can be found in the Darknet folder, and because this study employs the tiny YOLO version 4, its file name is yolov4-tiny.cfg.

The learning process of the model was executed in a Google Colab environment. It is recommended to run 2000 batches per class of the category; however, in this study, 9000 batches were run per class, to achieve better results, with the total learning time being over 20 h. After the learning process is completed, the Python file is transferred along with the model to a micro-SD card to be run on the Raspberry Pi. When the Python file is executed, it automatically opens a new window with live video input from the webcam. When a certain object is placed in front of the camera, it creates a boundary box along

with a label that indicates the category of that object. It is confirmed that the YOLO object detection in this study is implemented in Raspberry Pi and is performed at approximately 2 FPS in an actual operation scenario, resulting in an accuracy of 97–99%. The operation result of YOLO object detection is illustrated in Figure 8.



(**a**)



(**b**)

Figure 8. YOLO object detection results from camera point-of-view (**a**) and detailed information in terminal (**b**).

4.2. Hardware Implementation

After YOLO has been developed successfully to detect the categories to which the recyclables belong, the next step is to develop the hardware for the capacity monitoring feature. The ultrasonic sensors, which are attached to the top of each compartment, must be converted to a percentage value to achieve easy reading. Therefore, the equation for converting capacity to percentage is as follows:

$$\text{Capacity} = 100 - (((\text{ultrasonic duration} \div 29.155 \div 2.0)) \div \text{trashtrash bin height}) \times 100 \quad (1)$$

The "ultrasonic duration" indicates the duration of the ultrasonic travel time, and because sound travels at 343 m/s, it requires 29.155 μs/cm. Therefore, the ultrasonic duration is divided by 29.155 and then by 2 because the sound has to travel the distance

twice (to the object and then back to the sensor). The "trash bin height" in the equation indicates the height of the trash bin. The trash bin height in this system was 26 cm; therefore, this number was used to measure capacity. Once the capacity has been converted into a percentage, configuring the GPS to obtain the location coordinates and ESP8266 is important for uploading all data to the Firebase Database. For ESP8266, the required configurations are Wi-Fi SSID, password, Firebase Database API key, Firebase Database URL, user email, and password. Without these configurations, the ESP8266 cannot upload data.

The final hardware assembly in this study is presented in Figure 9, which illustrates a webcam, Raspberry Pi, ultrasonic sensor, GPS module, ESP8266 Wi-Fi module, and two Arduino Unos. The Raspberry Pi sends the results of the YOLO object detection to the Arduino, which is connected by wired communication using U-start cables. A separate Arduino Uno was adopted because the servo requires up to 1 A of instantaneous power when operating; therefore, the servo's direct connection to the Raspberry Pi can cause damage to the Raspberry Pi. To prevent this, separate auxiliary power from the four AA batteries was added and connected to the servo motor.

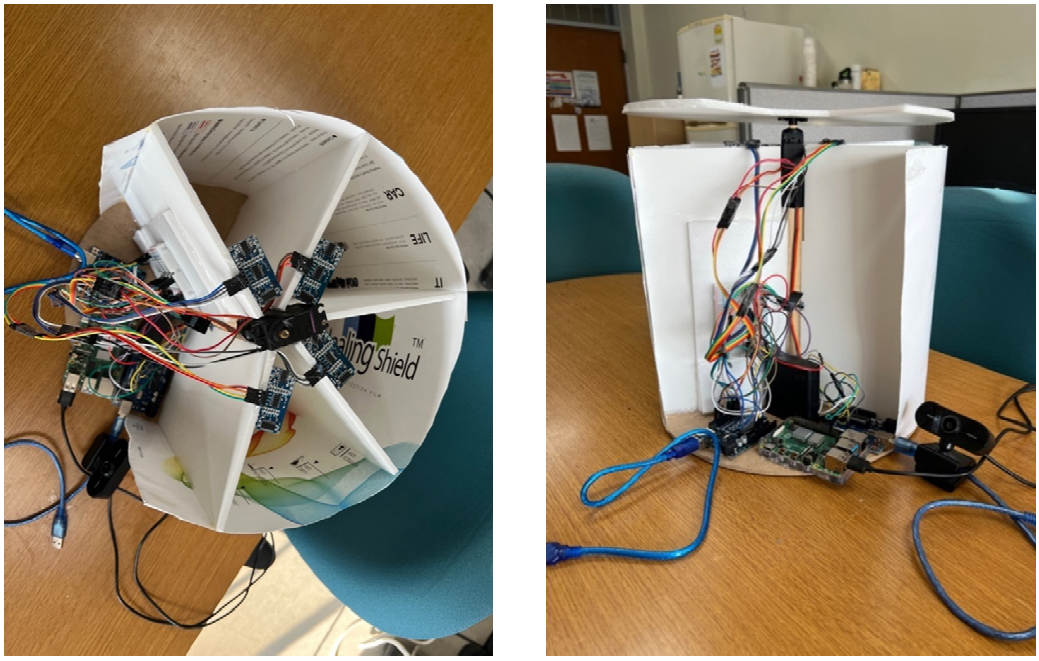
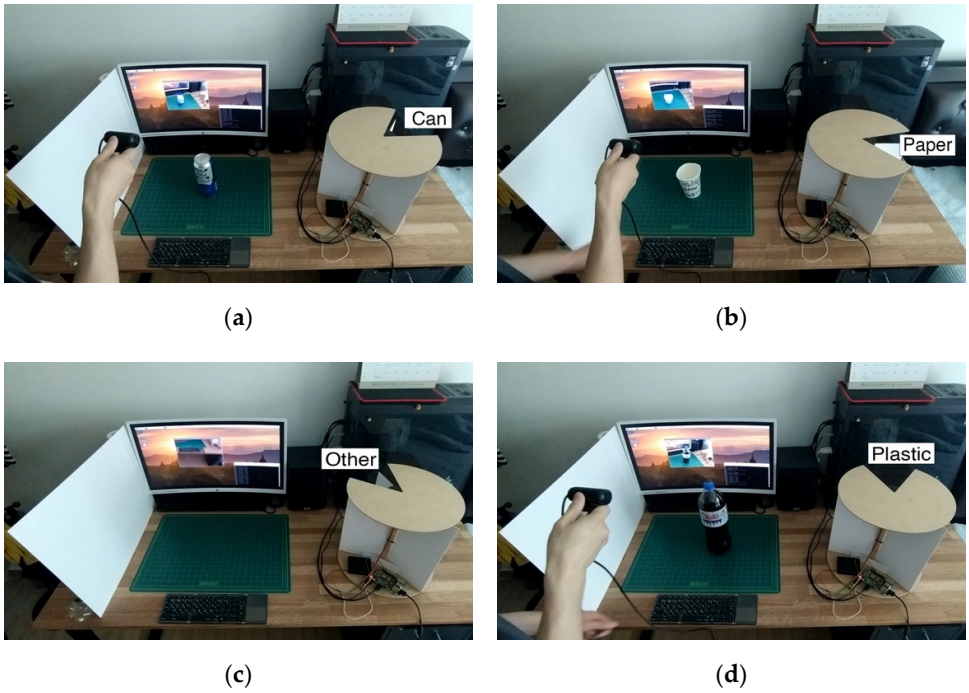


Figure 9. Final hardware assembly.

In addition, four ultrasonic sensors were attached to the top of each compartment to measure the capacity of the trash bin. Each ultrasonic sensor, the ESP8266 Wi-Fi module, and GPS module, were attached to an Arduino Uno. The object detection time interval was adjusted to a 2-s interval considering the rotating time of the servo, and the user throws away the recyclable trash.

The movement of the servo is illustrated in Figure 10, where Figure 10a depicts the case in which YOLO detects the object as a can, Figure 10b illustrates when YOLO detects the object as paper, Figure 10c is when YOLO detects the object as "other", and Figure 10d depicts the case in which YOLO detects the object as plastic.

As illustrated in Figure 10, when the YOLO model was deployed in Raspberry Pi and tested on actual recyclables, the accuracy ranged from 96 to 99% and the servo rotated the trash bin lid correctly to reveal the correct compartment according to the classified recyclables.



(**a**) (**b**)

(**c**) (**d**)

Figure 10. System integration testing with can (**a**), paper (**b**), other (**c**), and plastic (**d**) as recyclables.

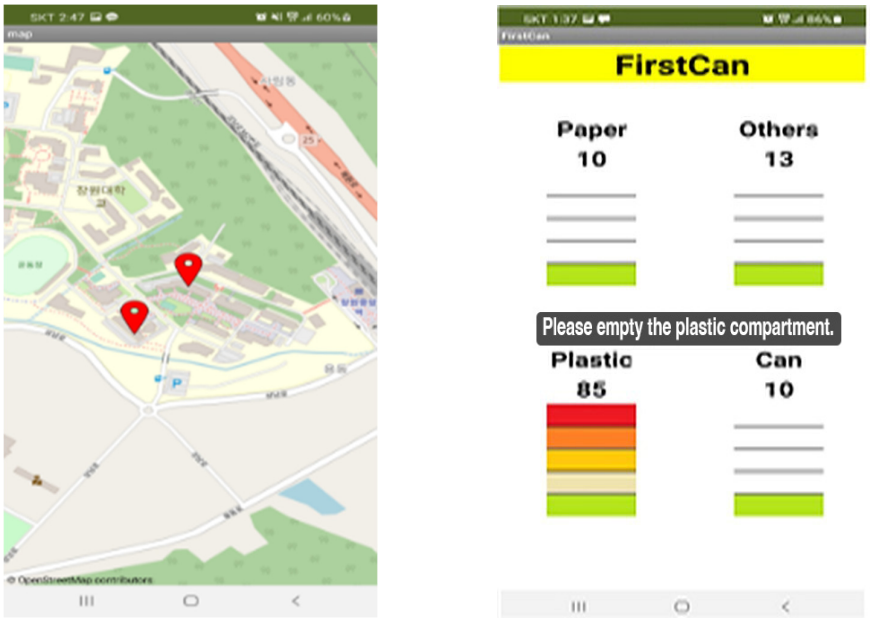
4.3. Mobile Application Implementation

A mobile application was developed to monitor the trash bin capacity using the MIT App Inventor, which is an intuitive visual programming environment that adopts a block-based coding method, as opposed to writing the code that supports Android, iOS, and tablet application development, thereby making it a user-friendly approach to developing a mobile application [19].

The application comprises two major user interfaces: a map that displays the location of the trash bin based on the GPS module attached and the trash bin capacity. Whenever a value change in Firebase Database is detected, it sets the newly read value. To display the status of the capacity in each compartment, there are live bars stacked on top of each other that show a colour that depends on the capacity. If the capacity is below 20%, the first bar is filled with green colour; if the capacity is between 20% and 40%, the second bar is filled with beige colour; if the capacity is between 40% and 60%, the third bar is filled with yellow colour; if the capacity is between 60% and 80%, the fourth bar is filled with orange colour; and finally, if the capacity is over 80%, the fifth bar is filled with red colour and it will display a notification to empty to the corresponding compartment. The development puzzle for the MIT App Inventor is presented in Figure 11, and a screenshot of the application is shown in Figure 12.

After the entire system is ready to run, when the user positions the recyclable trash in front of the camera, YOLO automatically detects the category of the recyclables, and then the servo rotates to reveal the correct compartment. The ultrasonic sensors continuously measure the distance from the top of the trash bin to the topmost object inside each compartment. The ultrasonic sensor raw data are captured by the Arduino Uno, which is then converted into a percentage value of the trash bin capacity. The results are then uploaded to the Firebase Database using the ESP8266 Wi-Fi module. Once the data are successfully uploaded, whenever the value changes, it is indicated in the Firebase Database, as illustrated in Figure 13. When the value changes, it also changes the value presented in the mobile application in real time as a capacity-monitoring functionality.

Figure 11. Puzzle code for mobile application in MIT App Inventor.



(**a**) (**b**)

Figure 12. Screenshot of the mobile application for showing the trash bin location on the map (**a**) and the trash bin capacity (**b**).

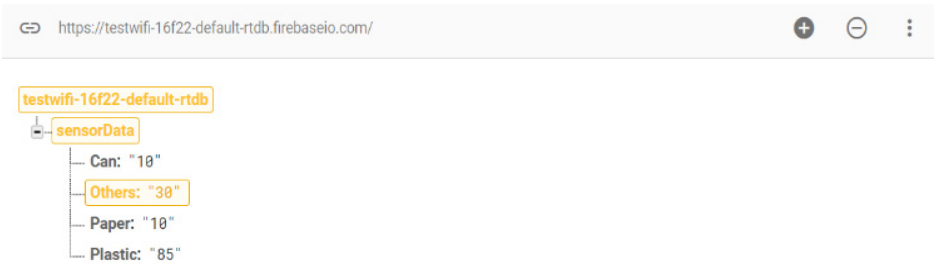


Figure 13. Capacity value in Firebase Database.

5. Performance Evaluation

There are various methods for determining the performance of a YOLO model; however, to evaluate the system, this study adopts the most common measurement method called mAP, which refers to the average accuracy (precision) of the object classification, box drawing for the detected object, and the model's confidence in generating a prediction. For this evaluation, the image to be trained and the image to be tested for performance evaluation should be constructed independently.

The mAP value of the YOLO model implemented in this study is presented in Figure 14, where the horizontal axis of the graph indicates how many times the YOLO model has been trained, and the vertical axis of the graph indicates the loss value. The blue line represents the loss value, which decreases as the number of training increases (the lower the value, the better). The red line indicates the mAP value, which increases as the amount of training increases until it converges to a constant value and becomes flat (the higher the value, the better). The mAP value of the YOLOv4-tiny model used for the implementation in this study is 75%, which exhibits an optimal result, considering that it is deployed in Raspberry Pi.

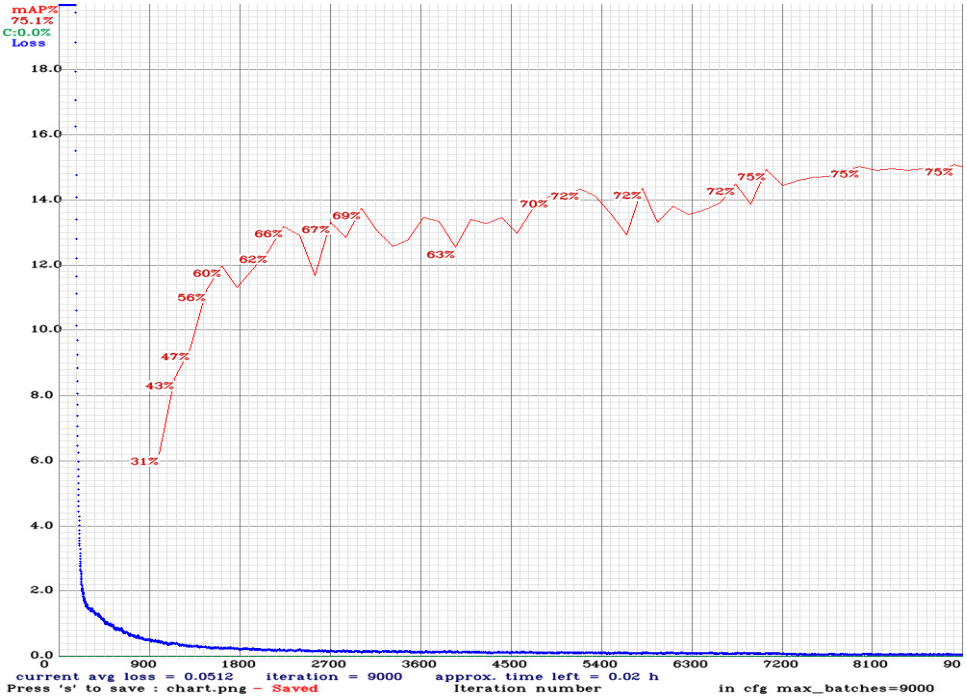


Figure 14. Loss and accuracy graph of the YOLOv4-Tiny.

For comparison, the same method was adopted to evaluate the performance of a full-size YOLOv4 model deployed on a Windows PC, as illustrated in Figure 15. After 6000 iterations, the mAP value was 91%, which is a 14% increase to the YOLOv4-tiny model accuracy. In this study, YOLOv4-tiny was adopted owing to the performance limitations and miniaturisation of Raspberry Pi; however, if the performance of the computing environment to implement the YOLO model is sufficiently supported to run a full-size YOLOv4 properly, the recyclables can be detected and classified more accurately and precisely.

To further evaluate the performance results, a comparison with other classification models is presented in Table 4. The classification model used for comparison comprises DenseNet121, DenseNet169, InceptionResnetV2, MobileNet from [9], WasteNet from [10], Faster R-CNN from [11], and R-CNN from [12].

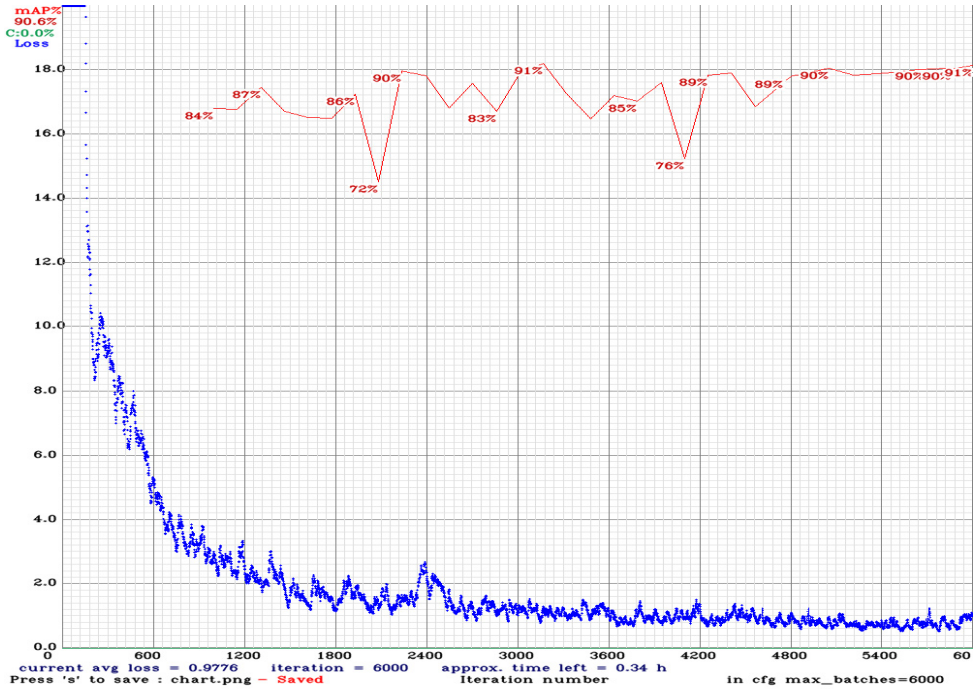


Figure 15. Loss and accuracy graph of the full-size YOLOv4.

Table 4. Performance comparison.

Classification Model	Dataset Number	Number of Epoch	Accuracy Results (%)
InceptionResnetV2	2527 images	210	89
DenseNet169	2527 images	127	95
DenseNet121	2527 images	110	95
MobileNet	2527 images	210	84
WasteNet	2527 images	1000	97
Faster R-CNN	2527 images	50	81
R-CNN	1,453,514,535 images	-	64%
YOLO	1000 images	9000	75%

As presented in Table 4, the YOLO classification model used in this study was not as accurate as the other classification models, especially WasteNet. However, some of the factors that contribute to this accuracy result are the dataset and Raspberry Pi. Although WasteNet adopts the TrashNet dataset with over 2000 images, the model in this study used the dataset from [18], with a smaller number of images. The usage of Raspberry Pi could contribute to the accuracy results because it exhibits a lower performance in terms of CPU and GPU than NVIDIA Jetson, where the WasteNet model was deployed.

Although, based on synthetic measurement, the accuracy of the model is relatively low, in a real-life scenario, YOLO can accurately predict and classify recyclables with an accuracy of over 95%, as illustrated in Figure 7, and can also rotate the trash bin properly to reveal the correct compartment according to the recyclables.

6. Challenges and Issues

The system implemented in this study presents several challenges. First, although the system can detect recyclables, the design only allows users to manually throw away the trash. The second issue is that users can only throw away one recyclable at a time. This is difficult if the user wants to throw away a bunch of recyclables with different categories because it takes a considerable amount of time to dispose of all of them. This is because the system cannot detect multiple objects, as the servo can rotate only once to reveal the correct compartment before re-scanning the object to rotate again to reveal another compartment.

The third challenge is that, with the current design, there is no proper way to empty the compartment simultaneously. The only access to the trash bin is from the top, making it difficult to empty the trash when it is full, because it implies that the officer needs to empty the trash from above.

7. Conclusions and Future Studies

The system implemented in this study is a trash bin that can automatically separate and collect recyclable trash using YOLO object detection, which runs in Raspberry Pi 4 B+ and classifies four recyclable categories (can, paper, plastic, and other). Here, the webcam retrieves a video image of the recyclables and functions as the input for classification using YOLO object detection. Once the object has been detected, a servo rotates the trash bin lid to reveal the correct compartment according to the detected recyclables.

In addition, ultrasonic sensors were attached to the top of each compartment, which measured the distance to the topmost object inside the compartment, to calculate the trash bin capacity and a GPS module to locate the trash bin coordinates. The trash bin capacity and location coordinates were gathered by Arduino Uno, sent to the ESP8266 Wi-Fi module, and then uploaded to the Firebase Database. A mobile application was also developed using the MIT App Inventor to retrieve the data in the Firebase Database and display the current trash bin capacity as a monitoring feature in real time.

After measuring the performance of the YOLOv4-tiny model used as the object detector trained with 1000 images in 9000 epochs, it was determined that the model achieved 75% accuracy. When the full-size YOLOv4 model was trained with a smaller number of epochs, while still using the same dataset, and deployed on a full-size personal computer (PC), the accuracy measurement increased to 91%. However, although a lighter model was used and there was a 14% decrease in accuracy when deployed in Raspberry Pi, the system was still considered satisfactory when detecting recyclables by scoring above 90% accuracy in a real-life scenario, as illustrated in Figure 8. After comparing the model used in this study with other similar classification models, it was demonstrated that when trained with a smaller number of images and deployed in Raspberry Pi, it achieved a lower accuracy than DenseNet121, DenseNet169, InceptionResnetV2, MobileNet, WasteNet, and Faster R-CNN, although it was significantly higher than R-CNN.

Furthermore, for capacity monitoring, the hardware can successfully upload the information to the Firebase Database, and the information can be displayed on the mobile application in real time. Therefore, although the YOLO classification model does not achieve the highest accuracy score, it is concluded that the development of the trash bin achieved the primary objective of this study.

The system proposed in this paper is part of a large project plan comprising three stages of development, of which this paper presents the first stage. In the first stage, the smart bin can only detect and classify recyclables; however, separating them still requires human interference. The second stage attempts to provide an improvement in terms of the classification model performance, while the recyclables' separation is fully automatic without any human interference. The third stage attempts to provide further improvement in classifying the recyclables and enables the user to throw away multiple recyclables at once, as opposed to throwing the recyclables away successively, as in the two previous stages. Therefore, in further studies, there will be a consideration of alternatives for classification models other than YOLO, as well as alternatives for single-board computers other than Raspberry Pi.

Author Contributions: Supervision, M.H.; writing—original draft, A.B.W.; M.H. conceived and presented the idea for the device to several undergraduate students at Changwon National University as a graduation project and encouraged A.B.W. as a graduate student to help in the development process. A.B.W. communicated and interviewed the project leader extensively to gather all the information regarding the development of the device to write the paper. A.B.W. created the hardware design and system architecture based on the developed hardware. A.B.W. translated and reinterpreted the undergraduate students' project reports and wrote the entire content. M.H. supported

by providing supervision of the project, reviewing and editing suggestions of the paper, as well as addressing several administration tasks. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Gyeongnam SW Convergence Cluster 2.0, under contract.

Institutional Review Board Statement: Ethical review and approval were waived because they did not involve identifiable personal or sensitive data.

Informed Consent Statement: Students' consent was waived because they did not involve identifiable personal or sensitive data.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Acknowledgments: The authors thank Seongsoo Han and Jeongwon Lee from the Department of Information and Communication Engineering, Changwon National University for providing the initial research data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwon, J. South Korea's Plastic Problem Is a Literal Trash Fire. Available online: https://www.cnn.com/2019/03/02/asia/south-korea-trash-ships-intl/index.html (accessed on 8 March 2022).
2. OECD Circular Economy—Waste and Materials. *Environment at a Glance Indicator*; Organisation for Economic Co-Operation and Development Publishing: Paris, France, 2022.
3. Chen, B.; Lee, J. Household Waste Separation Intention and the Importance of Public Policy. *Int. Trade Politics Dev.* **2020**, *4*, 61–79. [CrossRef]
4. Abdul-Rahman, F. Reduce, Reuse, Recycle: Alternatives for Waste Management. Available online: https://aces.nmsu.edu/pubs/_g/G314.pdf (accessed on 8 March 2022).
5. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
6. Ahmad, T.; Ma, Y.; Yahya, M.; Ahmad, B.; Nazir, S.; Ul Haq, A. Object Detection through Modified YOLO Neural Network. *Sci. Program.* **2020**, *2020*, 8403262. [CrossRef]
7. Lee, I.; Lee, K. The Internet of Things (IoT): Applications, Investments, and Challenges for Enterprises. *Bus. Horiz.* **2015**, *58*, 431–440. [CrossRef]
8. Samann, F. The Design and Implementation of Smart Trash Bin. *Acad. J. Nawroz Univ.* **2017**, *6*, 141–148. [CrossRef]
9. Aral, R.A.; Keskin, Ş.R.; Kaya, M.; Hacıömeroğlu, M. Classification of TrashNet Dataset Based on Deep Learning Models. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 2058–2062.
10. White, G.; Cabrera, C.; Palade, A.; Li, F.; Clarke, S. WasteNet: Waste Classification at the Edge for Smart Bins. *arXiv* **2020**, arXiv:2006.05873.
11. Kulkarni, H.N.; Raman, N.K.S. Waste Object Detection and Classification. Available online: https://cs230.stanford.edu/projects_fall_2019/reports/26262187.pdf (accessed on 8 March 2022).
12. Seredkin, A.V.; Tokarev, M.P.; Plohih, I.A.; Gobyzov, O.A.; Markovich, D.M. Development of a Method of Detection and Classification of Waste Objects on a Conveyor for a Robotic Sorting System. *J. Phys. Conf. Ser.* **2019**, *1359*, 012127. [CrossRef]
13. Sottile, M.J.; Minnich, R.G. Supermon: A High-Speed Cluster Monitoring System. In Proceedings of the IEEE International Conference on Cluster Computing, Chicago, IL, USA, 23–26 September 2002; pp. 39–46.
14. Li, W.-J.; Yen, C.; Lin, Y.-S.; Tung, S.-C.; Huang, S. JustIoT Internet of Things Based on the Firebase Real-Time Database. In Proceedings of the 2018 IEEE International Conference on Smart Manufacturing, Industrial Logistics Engineering (SMILE), Hsinchu, Taiwan, 8–9 February 2018; pp. 43–47.
15. Firebase Realtime Database. Firebase Documentation. Available online: https://firebase.google.com/docs/database (accessed on 8 March 2022).
16. Mikołajczyk, A.; Grochowski, M. Data Augmentation for Improving Deep Learning in Image Classification Problem. In Proceedings of the 2018 International Interdisciplinary PhD Workshop (IIPhDW), Świnoujście, Poland, 9–12 May 2018; pp. 117–122.
17. The Selenium Browser Automation Project. Available online: https://www.selenium.dev/documentation/ (accessed on 8 March 2022).
18. Garbage Classification. Available online: https://www.kaggle.com/asdasdasasdas/garbage-classification (accessed on 8 March 2022).
19. About US. Available online: https://appinventor.mit.edu/about-us (accessed on 8 March 2022).