

*Machine Learning, Embedded System*

Chicken Coop Monitoring System using You Only Look Once (YOLO) Algorithm

Omar Fauzi Rahman ¹, Tati Erlina ^{*2}

^{1,2} Department of Computer Engineering, Faculty of Information Technology, Universitas Andalas, Padang, West Sumatra, 25163, Indonesia

ARTICLE INFORMATION

Received: September 1, 2025
Revised: January 12, 2026
Available online: April 30, 2026

KEYWORDS

Monitoring System, Chicken Coop, You Only Look Once (YOLO), Raspberry Pi, Security.

CORRESPONDENCE

E-mail: tatierlina@it.unand.ac.id

A B S T R A C T

As chicken meat production increases in Padang City, farmers face serious threats from predators and intruders that cause significant losses. Current solutions, such as PIR motion detectors, have the disadvantage of not being able to recognize the objects they detect. This study designs a chicken coop monitoring system capable of identifying objects using the You Only Look Once (YOLO) algorithm. The system uses a Raspberry Pi 4 Model B as the processing center, connected to two Arducam OV5647 cameras mounted on MG996R servo motors. The system is also integrated with a Telegram application for real-time notifications. Based on testing, the system is capable of detecting and classifying objects with an average accuracy of 83.26% and an average notification response time of 10.8 seconds. Thus, the system is effective in addressing security issues in chicken farms

INTRODUCTION

Chicken meat production, especially free-range chickens and laying hens in Padang City, continues to increase. This is not surprising, considering that chicken is one of the most easily accessible sources of protein for various segments of society.

However, chicken farmers face serious challenges. Threats from intruders and predators are two major factors that have the potential to cause significant losses. Chicken coops are generally built far away from residential areas, although the aim is to reduce the spread of disease and environmental pollution, this makes them vulnerable to attack [1]. The issue of “the threat of predators and intruders to chicken coops located far from residential areas” is important to address in order to mitigate security vulnerabilities in chicken coops. Without an appropriate solution, farmers risk suffering significant losses that could have a major impact on supply and demand in the market.

Predators that threaten chickens in coops include weasels and snakes, which often hunt chickens indiscriminately. These predators and intruders can disrupt chicken meat production, especially predators that can also endanger residents living near chicken coops [2].

The purpose of solving the problem with the proposed solution is to protect poultry from predators and theft, thereby minimizing losses for chicken farmers. This problem needs to be solved with

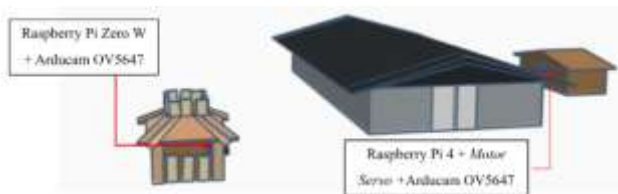
an efficient algorithm system to detect the presence of predators and intruders, so that threats approaching the chicken coop can be detected. This solution will be integrated with software to facilitate real-time remote monitoring.

Once objects are identified and predators or intruders are detected, the system will send real-time alerts to the farmer's mobile phone. In developing this solution, special attention must be given to energy efficiency to ensure the system can operate long-term. Additionally, the system must be able to process data quickly and accurately to ensure the information provided is relevant and timely.

METHOD

General System Design

The system is divided into two points, namely camera 1 and camera 2. Both cameras are placed slightly away from the cage with the cameras facing the cage, so that the camera system can monitor the conditions around the cage without having to turn the camera too far. The general system design can be seen in Figure 1.



The placement of cameras at two different points in the cage is intended so that the cameras can cover every corner of the cage that cannot be reached by the camera.

Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B is a single-board computer that is the size of a credit card and has high performance, enabling it to perform the same functions as a computer or laptop.



Arducam OV5647 NoIR Camera

The camera used in this system is the Arducam OV5647 NoIR Camera, which can capture images in real time with a resolution of up to 1920 x 1080 at a frame rate of 30 FPS or 1280 x 720 at 60 FPS [4].



Raspberry Pi Zero W

The role of the Raspberry Pi Zero W in this system is to run the YOLO (You Only Look Once) model on camera 2, then send the captured images to be processed by the main Raspberry Pi 4 processor.



YOLO Computational Method

The YOLO (You Only Look Once) algorithm is one of the deep learning algorithms that can identify objects in real time. The YOLO algorithm can detect objects quickly and efficiently by combining predictions of bounding boxes, often referred to as grids, and object class probabilities in a single evaluation. With the YOLO algorithm, the system can analyze the entire image and generate object detection in a short amount of time [6].

The YOLO algorithm can process image input at up to 45 FPS, even with a smaller neural network, such as Fast YOLO, which can process up to 155 FPS, making it the fastest algorithm for real-time detection. This is because YOLO uses the single-shot detection method, where the Convolutional Neural Network

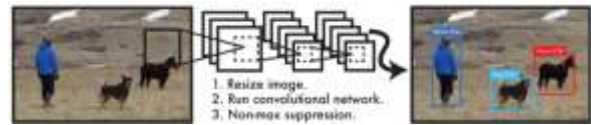


Figure 5. YOLO Detection Process[20]

As shown in Figure 4, this is the process of how the YOLO algorithm works. This algorithm has three stages before finally identifying objects, namely resizing the image, running the convolutional network, and non-max suppression. In the first stage, resizing the image, the image received from the camera or other input sources will be resized to a certain dimension so that it matches the input size expected by the CNN. The purpose of this stage is to simplify computations without losing important information in the image. Next is the run convolutional network stage, where after the image size is adjusted to the required specifications, the YOLO algorithm executes the convolutional network process to detect objects in a single step.

The image used will be divided into a grid (vertical and horizontal lines used to arrange objects or images), for example a 5x5 grid, where each grid cell will be assigned to detect the presence of objects within its area. Each grid cell will predict the bounding box for the object and calculate the confidence score, and this process is performed very quickly as it is done in a single step. Then, in the final stage, non-max suppression, this stage will reduce duplication in the bounding boxes that detect the same object. For example, if several boxes detect a “person,” the algorithm will select the box with the highest confidence score to mark the object and delete the other boxes.

Process Design

A system flowchart is a representation of the stages of each process performed by the system. The purpose of this flowchart is to provide an overview of the process and steps from start to finish in the designed system. The general workflow or flowchart of the chicken coop monitoring system illustrates how the entire system works from start to finish, as shown in Figure 6 below.

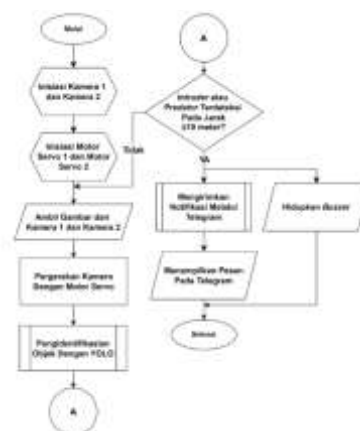


Figure 6. Main Flowchart of the System

The flowchart in Figure 6 explains the stages of system design, which begins with initializing the camera to capture images of the environment around the chicken coop and the servo motor to rotate the camera. The image capture process is carried out

simultaneously with the rotation of the camera using a servo motor at a rotation angle of 60° every 30 seconds. The captured images are processed using the YOLO algorithm to identify objects such as predators and intruders. Based on the analysis results using the YOLO algorithm, the system verifies whether the detected objects are within a radius of ≤ 10 meters from the coop or not. The YOLO algorithm calculates the distance using Stereo Vision in OpenCV.

After that, the system will decide whether the detected image contains a predator or intruder. If not, the system will return to the initial stage to continue taking pictures. However, if it does, the system will send a notification to the farmer's mobile phone via the Telegram app in the form of a 30-second video showing the conditions around the cage and a message about the threat, while simultaneously activating the buzzer as a warning to the environment around the cage. After that, the system will display the message sent to the Telegram app, and the system is considered complete, then returns to the initial stage.

Detailed Design of System Components

The hardware design of the system can also be drawn in the form of a schematic diagram. The schematic diagram also illustrates the relationship between each component in the system in more detail in terms of the interconnection between the pins of each component. The schematic diagram of the chicken coop monitoring system can be seen in Figure 7 below.

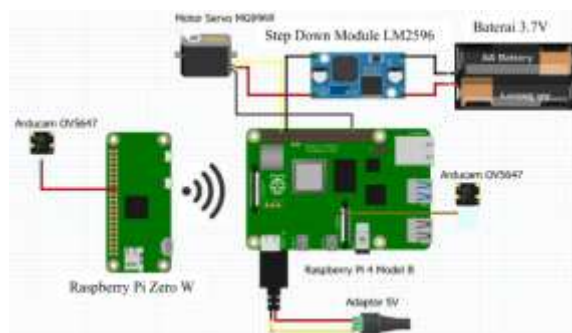


Figure 7. System Schematic Diagram

As can be seen in Figure 7, the components used in the system are: Raspberry Pi 4 Model B, Raspberry Pi Zero W, Arducam camera, LM2596 step-down module and 3.7V AA battery. The Raspberry Pi Zero W is connected to the Raspberry Pi 4 Model B wirelessly to send images captured by the Arducam OV5647 camera. The battery is used to power the MG996R servo motor and the step-down module is used to reduce the battery voltage from 7.4V to 5V.

RESULTS AND DISCUSSION

Hardware Implementation

The implementation of the hardware for the chicken coop monitoring system was carried out by storing most of the components in a 15 x 10 cm rectangular casing. This casing contains a Raspberry Pi 4 Model B, an Arducam OV5647 camera and a MG996R motor servo. The implementation of the hardware system can be seen in Figure 8.



(a) Camera 1



(b) Camera 2

Figure 8. Hardware Implementation of System

Figure 4.1(a) shows the implementation of camera 1 in the system, and Figure 4.1(b) shows the implementation of camera 2. The two cameras are positioned slightly away from the enclosure, facing towards it. This allows the camera system to monitor the surrounding area without having to turn the cameras too far. For details of the hardware implementation, see Figures 9 to 12.

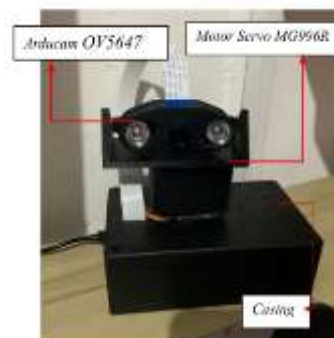


Figure 9. Exterior view of the camera and casing 1

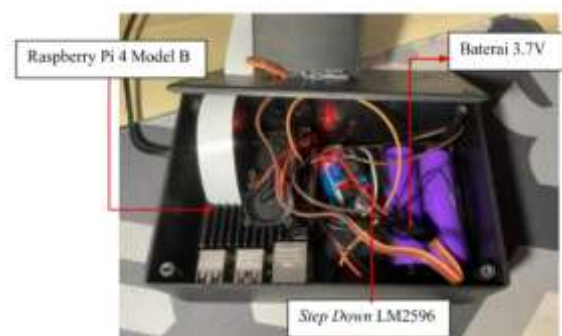


Figure 10. Hardware details inside case 1

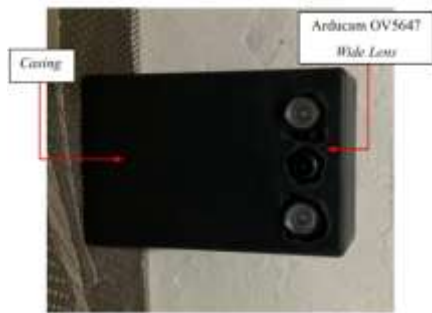


Figure 11. Exterior view of camera 2 and casing

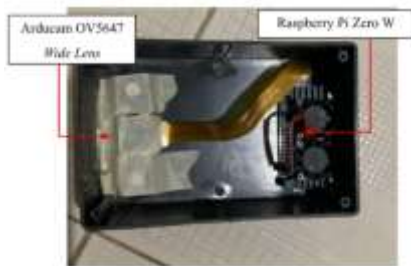


Figure 12. Hardware details in case 2

The hardware details implemented in Figures 9 to 12, show the components of the designed system. These components are stored in a casing that has been specially designed to store each device safely and securely. The following is an explanation of each component used and its function:

1. The MG996R servo motor is used to support the camera so that it can cover every side of the enclosure and avoid blind spots. A camera that will later be rotated is placed on top of the motor servo.
2. The Arducam OV5647 Normal Lens is placed above the motor servo as an input device for capturing images of intruders and predators approaching the enclosure.
3. The Raspberry Pi 4 Model B is placed inside the casing, parallel to the camera, to minimise damage to the FFC (Flat Flexible Cable) cable. The Raspberry Pi is used as the central processing unit for the images received from the Arducam OV5647 camera and the images sent by the Raspberry Pi Zero W, running the YOLO (You Only Look Once) model.
4. The LM2596 step-down module is used to reduce the voltage generated by the battery from 7.4 V to 5 V.
5. The 3.7 V lithium-ion battery is used as the power source to drive the servo motor.
6. The Arducam OV5647 Wide Lens is placed in casing 2 and is used to capture images of intruders and predators approaching the enclosure.
7. The Raspberry Pi Zero W is placed in casing 2 and is used to send images wirelessly to the Raspberry Pi 4.

Software Implementation

The implementation of the software in this system includes creating a machine learning model using the YOLO algorithm, programming the Raspberry Pi to process the YOLO model, running the actuator and programming the Telegram application. The first stage of this implementation involves creating a dataset and designing a machine learning model with the YOLOv5 algorithm. The model will then be run on a Raspberry Pi using Python programming language. The results of the processing carried out by the Raspberry Pi can be viewed in the Telegram

application, which has a feature for displaying videos captured by cameras connected to the Raspberry Pi.

Implementation of Machine Learning Model Training

The training process is carried out on Google Colaboratory, which offers free access to GPU (Graphics Processing Unit) and TPU (Tensor Processing Unit), helping to speed up the training process for the model compared to using a CPU. The combination of GPU and TPU is highly efficient in executing the parallel calculations required during the training of deep neural networks [8].

Training on Google Colaboratory involves creating code to ensure the directory structure on Google Colab. At this stage, the model to be built is YOLOv5, which was designed by Ultralytics, so the process of cloning the official Ultralytics repository is required. Then the necessary libraries, such as PyTorch and OpenCV, are installed from requirements.txt.

Once the installation of all the modules is complete, the dataset prepared in Roboflow is uploaded to Google Colab using the Roboflow API. Once the dataset has been imported into Google Colab, the machine learning model is ready for training. The machine learning model used in this system is YOLOv5n, or YOLOv5 Nano, which is the lightest YOLOv5 architecture compared to other YOLOv5 models. The advantage of the YOLOv5n model lies in its detection capabilities despite its limited resource usage [9].

The training process uses 80 epochs or iterations, whereby the model reads the dataset 80 times to understand the repeated patterns. The use of '--device 0' ensures that during training, the hardware used is the GPU and TPU, enabling the training process to run more quickly and efficiently than using a CPU. Once the training process is complete, the results can be viewed as shown in Figure 13 below.

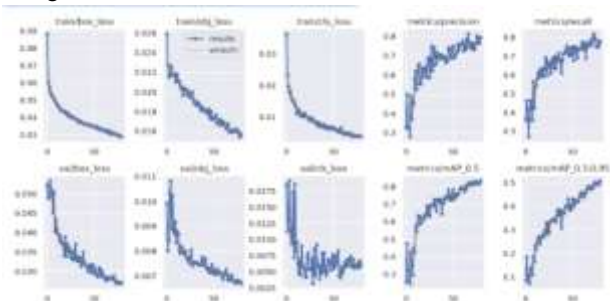


Figure 13. Training Dataset Results

Implementation of the Telegram application

The implementation of the Telegram application is intended to provide a communication interface between the farmer and the system, enabling the sending of real-time threat notifications, system control and monitoring of the conditions in the pen. The implementation on Telegram involves creating a Telegram bot that receives alerts when the system detects intruders and predators, and displays the results of video capture by the two cameras. The Telegram bot is called Jagayam.

The implemented Telegram bot is able to notify the farmer of objects detected by the camera within a radius of ≤ 10 metres from the pen. The notification message sent is in the form of a video of the camera's capture, for example 'Human detected on Camera 1!', as can be seen in Figure 14.



Figure 14. Notification of Telegram messages to users

In addition to sending notification messages to the farmer, the Telegram bot designed for this purpose enables the farmer to switch the system on or off and view the last 10 seconds of video from the camera. If the farmer selects the option to turn on the system, the system will initialise the camera and the servo motor to start monitoring the environment in the pen. Conversely, when the option to turn off the system is selected, all monitoring activities in the pen will be stopped. When the farmer selects the option to display the last 10 seconds of video from each camera, the Telegram bot will display the latest footage from the environment monitored by the camera and send it together with a message explaining whether or not any threats have been detected. This display can be seen in Figure 15.

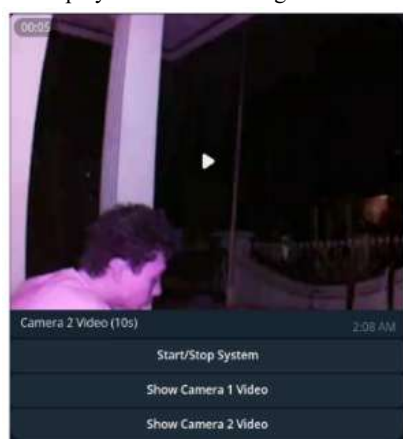


Figure 15. System Control Menu Display via Telegram

System Testing

The system was tested in its entirety at the chicken farm where it was integrated, i.e. at the close house chicken farm at the Faculty of Veterinary Science, University of Andalas, and at the pet shop. The test was carried out to prove and evaluate the overall performance of the system's inputs, processes and outputs.

However, testing of the musang and snake objects was carried out in a pet shop, due to limited access to bringing the musang and snake directly to the chicken coop. Additionally, musang and snake detection at night was carried out using a replica, as there was no access to live musang and snakes for detection.

Testing was conducted over three days in a chicken coop and a pet store. Testing was conducted under two conditions: daylight (light) and nighttime (dark). Testing in the chicken coop involved displaying an object in the area detected by the camera. Meanwhile, testing in the pet store involved pointing the camera directly at the object.

The results of the overall chicken coop monitoring system testing can be seen in Table 1 – Table 7.

Table 1. Human subject testing in chicken coop

Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
15.000	<i>Human</i>	17:09:05	17:09:18	2.88 m
14.000	<i>Weasel</i>	17:11:09	17:11:21	2.22 m
14.000	<i>Human</i>	17:11:03	17:11:18	6.12 m
12.000	<i>Human</i>	17:50:08	17:50:18	5.36 m
12.000	<i>Human</i>	17:51:14	17:51:24	0.53 m
11.000	<i>Human</i>	17:57:22	17:57:39	5.19 m
10.000	<i>Human</i>	17:59:55	18:00:01	4.37 m

Table 2. Human subject testing in pet shop

Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
12.000	<i>Human</i>	16:38:04	16:38:07	2.88 m
12.000	<i>Human</i>	16:38:38	16:38:45	2.22 m
11.750	<i>Weasel</i>	16:45:46	16:45:50	6.12 m
11.650	<i>Weasel</i>	16:48:22	16:48:29	5.36 m

Table 3. Weasel subject testing in pet shop

Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
13.000	<i>Weasel</i>	16:05:25	16:05:35	2.34 m
13.000	<i>Weasel</i>	16:16:08	16:16:14	1.28 m
12.700	<i>Weasel</i>	16:17:13	16:17:25	1.25 m
12.700	<i>Weasel</i>	16:18:26	16:18:31	2.45 m
12.000	<i>Human</i>	16:48:24	16:48:32	0.72 m
11.500	<i>Weasel</i>	16:51:11	16:51:16	0.60 m
11.500	<i>Weasel</i>	16:53:05	16:53:10	0.49 m

Table 4. Snake subject testing in pet shop

Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
12.500	<i>Snake</i>	16:11:45	16:11:54	0.27 m

11.000	<i>Snake</i>	16:40:53	16:40:57	0.70 m
11.750	<i>Snake</i>	16:42:28	16:42:32	1.20 m
11.500	<i>Weasel</i>	16:43:31	16:43:36	1.01 m
11.500	<i>Snake</i>	16:45:24	16:45:34	0.43 m
11.500	<i>Snake</i>	16:46:48	16:48:57	0.30 m
11.100	<i>Snake</i>	16:51:27	16:51:32	0.30 m

Table 5. Human subject testing around the house

Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
40.000	<i>Human</i>	13:45:36	13:45:39	2.91 m
0,01	<i>Weasel</i>	2:47:48	2:47:52	0.52 m
0,01	<i>Human</i>	3:07:33	3:07:35	2.63 m
0,01	<i>Human</i>	3:07:49	3:07:53	2.14 m
19.000	<i>Human</i>	10:59:32	10:59:38	5.66 m
19.000	<i>Human</i>	11:00:06	11:00:16	4.29 m
19.000	<i>Human</i>	11:00:45	11:00:51	6.82 m

Table 6. Weasel subject testing around the house

Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
0,01	<i>Human</i>	2:28:49	2:28:50	0.84 m
0,01	<i>Weasel</i>	2:29:19	2:29:27	0.69 m
15	<i>Snake</i>	3:16:22	3:16:27	0.78 m
15	<i>Weasel</i>	3:17:21	3:17:27	1.03 m
15	<i>Weasel</i>	3:18:47	3:18:54	1.01 m
20	<i>Weasel</i>	3:25:38	3:25:34	0.43 m
0,01	<i>Human</i>	2:28:49	2:28:50	0.84 m

Table 7. Snake subject testing around the house

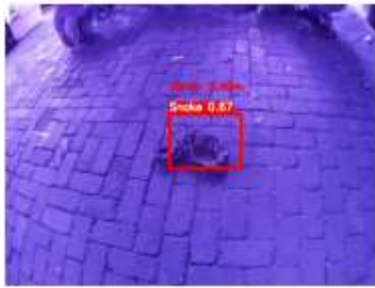
Intensitas Cahaya (lux)	Output Telegram	Waktu Objek Muncul	Waktu Objek Terdeteksi	Jarak
0,01	<i>Snake</i>	3:10:28	3:10:33	0.34 m
0,01	<i>Snake</i>	3:11:48	3:11:58	0.35 m

0,01	<i>Snake</i>	3:12:39	3:12:47	0.38 m
0,01	<i>Snake</i>	3:14:46	3:14:54	0.54 m
0,01	<i>Snake</i>	3:15:55	3:15:58	0.50 m

Documentation of the conditions of the test location for dark and light conditions can be seen in Figures 16 and 17.

**(a) Human****(b) Snake****(c) Weasel****Figure 16. Testing in dark conditions****(a) Human**

(b) Weasel



(c) Snake

Figure 17. Testing in bright conditions

Based on testing of the entire system, results showed optimal performance in detecting intruders (humans) and predators (monkeys and snakes) in both dark and bright lighting conditions. Dark lighting refers to nighttime, and bright lighting refers to daytime. This was done because the chicken coop monitoring system is especially important at night when there is minimal activity around the coop.

The Arducam OV5647 camera used allows for object detection in low-light conditions due to its infrared capability, ensuring effective detection at night. Detection times for message notifications on the Telegram app ranged from 6 to 17 seconds, with an average delivery time of 10.8 seconds. In bright conditions, detection times reached 6 to 15 seconds, such as for a human object from 5:11:03 to 5:11:18 (15 seconds). This demonstrates that good lighting can improve the processing efficiency of the YOLOv5 Nano model in detecting objects.

The Arducam OV5647 camera's capabilities proved effective in maintaining system reliability in dark coop environments. The analysis of the detected time speed shows that the system is able to send notifications via Telegram within a sufficient timeframe for real-time purposes, with the longest time being 17 seconds for human detection in dark conditions (1:45:22 – 1:45:39), possibly affected by lighting challenges.

Test results also revealed some identification difficulties, with objects such as snakes and civets sometimes being difficult to detect or misclassified by the YOLO model. This occurred because some tests used images from mobile phones or paper, leading to confusion in detection. Furthermore, Telegram delivery also encountered several errors, such as when a human object was detected but the message sent indicated a civet or snake. This was caused by the detection of more than one class within a given timeframe, resulting in the system only sending messages for one object.

Therefore, it can be concluded that the system is quite functionally optimized for chicken coop security monitoring, with a wide range of servo motors integrating a 0-180° range for wide coverage, as well as consistent Telegram output for user communication. Overall, the system effectively detects intruders and predators around chicken coops, especially with IR support for nighttime detection. The average notification speed of 10.8 seconds is responsive enough for real-time detection needs of farmers.

CONCLUSIONS

The conclusion from the chicken coop monitoring system is that it has successfully developed a system that can assist chicken farmers in monitoring their surrounding environment and identifying intruders and predators attempting to approach the coop area. Therefore, the complex engineering problem of threats to chicken coops located far from residential areas can be addressed with the designed chicken coop monitoring system.

ACKNOWLEDGMENT

For further development of the chicken coop monitoring system, components with higher computing power and greater cost efficiency can be used in future research, thereby improving the system. Furthermore, it is recommended to improve the machine learning algorithm model to better identify objects..

REFERENCES

- [1] E. Wahyuni dan D. Santoso, "Dampak Lingkungan dan Keberlanjutan Peternakan Ayam Ras Pedaging Pola Kemitraan," *Jurnal Agrikultura*, vol. 34, no. 2, hlm. 237–254, 2023, doi: [dx.doi.org/10.24198/agrikultura.v34i2.46783](https://doi.org/10.24198/agrikultura.v34i2.46783).
- [2] D. T. Pangestu dan S. Azizah, "Dampak Sosial Ekonomi Peternakan Ayam Kampung Berskala Mikro Di Desa Payaman, Nganjuk," *Jurnal Pendidikan Ilmu Pengetahuan Sosial (JPIPS)*, vol. 14, hlm. 31–39, Jun 2022.
- [3] "Raspberry Pi 4 Model B," Apr 2024.
- [4] Arducam, "Arducam OV5647 NoIR Camera Module for Raspberry Pi," 2013.
- [5] "OV5647 Raspberry pi camera 5MP Lensa Wide Angle IR Night Vision Light - 130degree."
- [6] O. Alfiano dan dan Santi Rahayu, "Implementasi Algoritma Deep Learning YOLO (You Only Look Once) Untuk Deteksi Kualitas Kentang Segar dan Busuk Secara Real Time," *JORAPI: Journal of Research and Publication Innovation*, vol. 2, no. 3, hlm. 2470–2478, Jul 2024.
- [7] J. Redmon, S. Divvala, R. Girshick, dan A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection."
- [8] G. I. E. Soen, M. Marlina, dan R. Renny, "Implementasi Cloud Computing dengan Google Colaboratory pada Aplikasi Pengolah Data Zoom Participants," *JITU: Journal Informatic Technology And Communication*, vol. 6, no. 1, hlm. 24–30, Jun 2022, [Daring]. Tersedia pada: <https://ejournal.uby.ac.id/index.php/jitu/article/view/78>

- [9] G. Jocher, "YOLOv8 vs YOLOv5: A Detailed Comparison." Diakses: 17 Juli 2025. [Daring]. Tersedia pada: <https://docs.ultralytics.com/compare/yolov8-vs-yolov5/#conclusion-which-model-should-you-choose>

AUTHOR(S) BIOGRAPHY



Omar Fauzi Rahman

Born in Tangerang, Ciputat, on January 9, 2003. He attended MI Pembangunan UIN Jakarta, MTS Pembangunan UIN Jakarta, and SMAN 6 Jakarta. He then continued his studies at the Department of Computer Engineering, Faculty of Information Technology, Andalas University.