Performance evaluation of ESP32 Camera Face Recognition for various projects

Article *in* Internet of Things and Artificial Intelligence Journal · February 2022

DOI: 10.31763/iota.v2i1.512

Puput Dani Prasetyo Adi
BRIN
67 PUBLICATIONS **676** CITATIONS

Article

Performance evaluation of ESP32 Camera Face Recognition for various projects

Puput Dani Prasetyo Adi, Yuyu Wahyu

[1,2]National Research and Innovation Agency (BRIN-RI), Sangkuriang street, LIPI Complex Gd. 20, Cisitu Lama street, Dago, Coblong District, Bandung City, West Java 40135, Indonesia
* Corresponding author: puput.dani.prasetyo.adi@brin.go.id

Abstract: Images Recognition and Face Recognition are a form of artificial intelligence developed on Robots and Smart CCTV, such as technology currently developing, such as YOLO Real-time Object Detection and Images Recognition using Python language. In this research, the tool trying to be used is the ESP32 camera and his approach to several types of study, such as smart door locks. To activate this ESP32 camera, use the ATmega 328 mini Microcontroller, Arduino board, and FTDI Programmer. The thing that can be analyzed is the delay or inference time (ms) on the ESP32 camera. Face recognition technology in this research is devoted to using the ES32 camera, which is expected to be an inexpensive image recognition solution. This research using the ESP32 camera was made as easy as possible to be applied by everyone, but by not leaving the analysis side, one of which is on the time side of the camera's detection of the object in front of it, how far and fast (ms), this object can be explicitly detected. Speed is seen from the resolution specifications used for face recognition and image recognition, such as CIF and QVGA, and throughput (bytes) analysis of ESP32-Cam and Server using Wireshark.

Keywords: face recognition, esp32 camera, intelligent detection, smart door lock, Artificial intelligence

check for updates

Citation: P.D.P.Adi, Yuyu.W, "Performance evaluation of ESP32 Camera Face Recognition for various projects", *Iota*, **2022**, ISSN 2774-4353, Vol.02, 01.**https://doi.org/10.31763/iota.v2i1.512**

Academic Editor : Andri.P.

Received : 16 January 2022

Accepted : 07 February 2022

Published : 18 February 2022

1. Introduction

Currently, image recognition technology continues to develop with easy and inexpensive technologies to reach and develop, one of which is the Raspberry Pi. Raspberry Pi is now the latest version 4B, which continues to create Python-based Artificial Intelligence. And also still developing some Artificial Intelligence features on the M5Sticky board that can recognize images through the camera. And recently, ESP32-Cam appeared with the cheapest version that can automatically recognize images or photos in front of it. However, this ESP32-Cam has many weaknesses, including errors when reading the webserver [2,8] and not being detected by the camera. The author changed the microcontroller, which is not using the FTDI Programmer but uses the Arduino Uno as a microcontroller programmed to read the ESP32-Cam camera. The ESP32-Cam is equipped with the UFL Antenna feature to strengthen the Wireless communication used [15], namely the WiFi module. Moreover, this paper is a complete analysis of all components used by ESP32-CAM in carrying out a particular project, including projects using solenoid valves and other projects using UFL Antenna components and propagation analysis OV2640 camera analyzing its performance including delay, resolution, and capture speed.

2. Theory

2.1. UFL Antenna

Figure 1 shows the Male UFL Antenna port of the ESP-32 Cam viewed using a digital microscope. The function of this UFL Antenna is as a signal amplifier to communicate with the receiver via its WiFi module. In detail, the specifications of the ESP32-Cam can be seen in table 1.

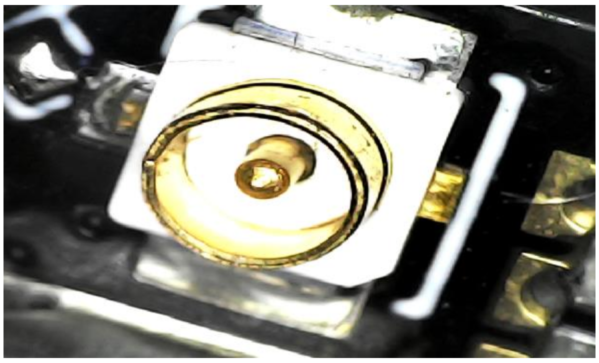


Figure 1. Male UFL antenna connector ESP32-Cam

As in table 1, the Camera ESP32-Cam [1,3,4,6,12] supports the OV2640 type, and table 1 discusses the overall specifications used in the ESP32-Cam. In more detail, the OV2640 camera is shown in Table 2. Furthermore, Figure 1 shows the position of the ESP32-Cam UFL-Antenna Port, located on ANT1, for the flexible antenna cable can be plugged in and removed from the port effortlessly. Also called an external antenna because it is located outside the ESP32-Cam component or not in the form of a PCB Antenna.

Table I. ESP32-Cam Spesification

No	ESP32-Cam Spesifications		
1	Ultra-small 802.11b/g/n Wi-Fi + BT/BLE SoC module	- Up to 240MHz, up to 600 DMIPS	Built-in 520 KB SRAM, external 4M PSRAM
2	Low-power dual-core 32-bit CPU for application processors	Supports interfaces such as UART/SPI/I2C/PWM/ADC/DAC	Support OV2640 and OV7670 cameras with built-in flash
3	Support for images WiFI upload	Support TF card	Support multiple sleep modes
4	Embedded Lwip and FreeRTOS	Support STA/AP/STA+AP working mode	Support Smart Config/AirKiss One-click distribution network
5	Support for local serial upgrade and remote firmware upgrade (FOTA)	Support secondary development	

Table II. OV2640 Spesification And Datasheets

	OV2640 Specifications & Datasheets	
No	**Parameters**	**Spesifications**
1	Array Size	1600 x 1200 (UXGA)
2	Power Supply	Core: 1.3V DC ± 5% Analog: 2.5~3.0V DC I/O: 1.7V to 3.3V
3	Power Consumption	Free running: 125mW Standby:600μA
4	Image Sensor Format	Type 1/4″

OV2640 Specifications & Datasheets

No	**Parameters**	**Spesifications**
5	Maximum Image Transfer Rate	1600×1200@15fps, SVGA@30fps, CIF@60fps
6	Sensitivity	0.6V/Lux-sec
7	S/N ratio	40dB
8	Dynamic Range	50dB
9	Pixel Size	2.2 x 2.2 µm
10	Output Format	YUV/RGB/Raw RGB Data/MJPEG
11	Shutter Type	Rolling Shutter

2.2. ESP32-Cam FTDI and Arduino Connection

The connection between ESP32 and IC Programmer such as FTDI can be seen in Figure 2. FTDI functions as USB to TTL (Transistor-Transistor Logic). So that during the transfer process, the C++ program [9,10,11,14] to identify the ESP32-Cam is required to initialize the FTDI Port and ESP32-Cam port. Until the data transfer process or Compiling data is 100% successful. The ESP32-CAM has 16 pins used for the Input and Output or I/O processes. These include 5V DC, GND, GPIO 12, GPIO 13, GPIO 15, GPIO 14, GPIO 2 and GPIO 4. On the right are 3.3 V DC, GPIO 16, GPIO 0, GND, GPIO 3, and GPIO 1. Each GPIO has functioned as Data, CLK, CMD, CSI_MCLK, RX, and TX.

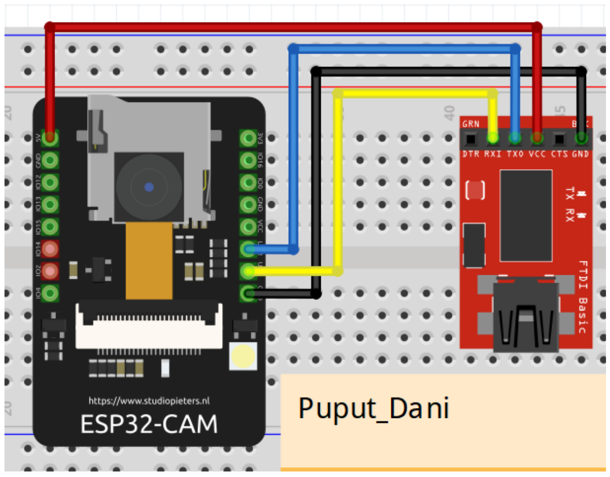


Figure 2. ESP32-Cam FTDI Connection

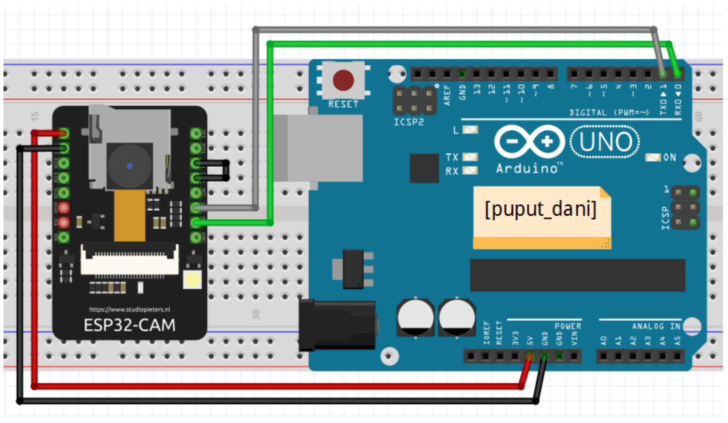


Figure 3. ESP32-Cam Arduino Connection

ESP32-CAM is used for several actuator control processes. For example, control of DC motors, Servo motors, Solenoid Valves, and other outputs. With input coming from Face or Image recognition [13], ESP32-CAM is also used as a streaming camera tool like CCTV Camera. Still, the advantage of ESP32-CAM is that it is equipped with Artificial Intelligence (AI) using specific programming languages [7] or coding programs that can set the output for the research purpose. The ESP32-CAM is combined with an Arduino UNO type microcontroller as FTDI and can directly use USB to TTL or FTDI. Some examples of research using ESP32-CAM are Face Recognition to open doors[5]. ESP32-CAM can be used as Face recognition by recording facial data as a key to open the door by moving the Solenoid Valve in the ON or OFF position.

Table III. ESP32-Cam And FTDI Wiring or Connection

	ESP32-Cam and FTDI Wiring or Connection	
Pin	**ESP32-Cam**	**FTDI**
1	GND	GND
2	-	CTS
3	5 V	VCC
4	GPIO 3 (UOR)	TX0
5	GPIO 1 (UOT)	RXI
6	-	DTR

Table IV. ESP32-Cam and Arduino Wiring or Connection

	ESP32-Cam and Arduino Wiring or Connection		
Pin	**ESP32-Cam**	**ESP32-Cam**	**Arduino**
1	5 V	-	5 V
2	GND	-	GND
3	GPIO 3 (UOR)	-	RX 0
4	GPIO 1 (UOT)	-	TX 0
5	IO0	GND	-

Figure 4 shows the Face Recognition Steps, including Capturing, Extracting, Comparing, and Matching. This Face recognition method is used to determine the Face that is the key by going through various Face Comparing and Matching processes; after the Matching process, the actuator moves to the ON position (in the case study, the automatic door opens closes using the Face).

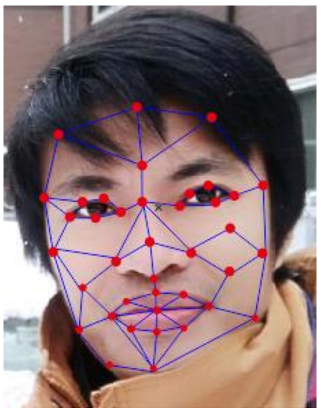


Figure 4. Face Detection use ESP32-CAM

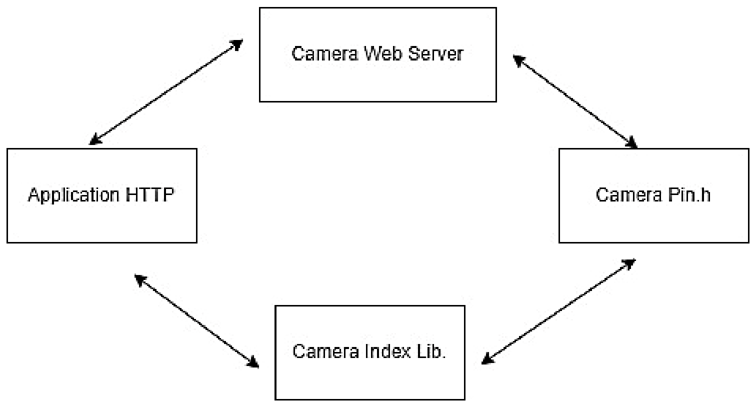


Figure 5. Programming Packet in Arduino IDE

The programming structure can be seen in the body in Figure 5. There are five core components in the Integrated Development Environment that are used to run ESP32-CAM. Camera Web Server, Application HTTP or application server, Camera Library, and Camera Pin Header. And each of these components must be a unified system. Moreover, the pseudocode used for minimal starting is shown in the following series of pseudocodes:

```
1.  Start the program
2.  Initialize of Library esp_camera.h and WiFi.h
3.  Select the Camera Model AI Thinker has PSRAM

4.  Initialize of Header of Camera Pins
5.  Initialize of your SSID and Password
6.  Initialize of Password of SSID
7.  Start of Camera Server as Void loop
8.  Start the Void Setup
9.  Specify the bitrate value 115200 bps
10. Specify of Serial.setDebugOutput(true)
11. Specify of camera and Pins configuration
12. WiFi Configuration
13. Camera Server ready
14. WiFi IP Address Status Connected
15. Create the Own program and running
16. End the program

                ---------- Pseudocode 1  ----------
```

```
1.  Define a CAMERA_MODEL_AI_THINKER
2.  Define a WiFi and Password used with a const char ssid & password
3.  Define a relay port used
4.  Define a long prevMillis & internal (int)
5.  Define a Boudrate 115200
6.  Define a Pin mode (relay as output and) & Digital Write (relay is
    low)
7.  Define active relay condition (When a High and When a Low)
8.  If matchFace is true Then relay is High
9.  If matchFace is False Then relay is Low
10. If Relay is High Then Solenoid is Unlock position
11. If Relay is Low Then Solenoid is lock position
12. Repeat the automatic door lock process with Face recognition

                ---------- Pseudocode 2  ----------
```

2.3 Images Recognition

Figure 3 is the Face Detection and Face Recognition process; Face Detection starts from the object (Grayscale object) captured by the camera (camera resolution is essential), then programming in C++ on the Arduino or Python IDE can produce object detection automatically on several objects specifically in the shape of the eyes and mouth. After the Face Detection process, the Face Recognition process starts from Pre-Processing to Face Representation followed by Face Database (here all files will be used as databases for the image recognition process from hundreds of images), then Face Classification, this is a face classification process based on the training process images, and finally the verification or identification result process.

Table V. ESP32-Cam Doorlock Pin

	ESP32-Cam Door Lock Connection		
Pin	**ESP32-Cam**	**FTDI**	**Relay**
1	5v	VCC	VCC
2	GND	GND	GND
3	UOR	Tx	-
4	UOT	Rx	-
5	IO4	-	IN

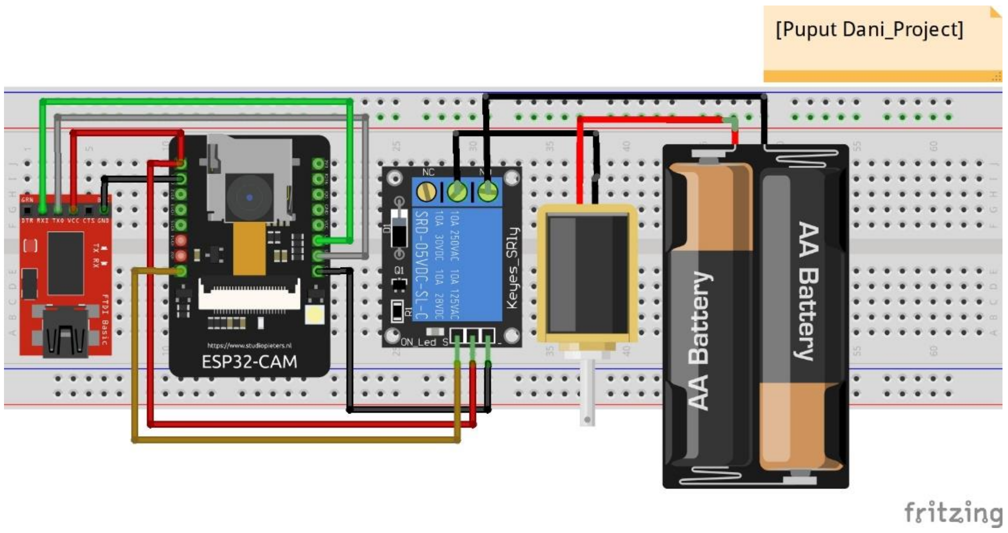


Figure 6. Automatic Door lock use ESP32 Cam [5]

The development of this Face recognition project can be applied to automatic door locks based on ESP32-Cam. Completely can be seen in Figure 6. Figure 6 is the connection between several components: FTDI, ESP32-Cam, Solenoid, Relay, and Battery 12 Volt DC. The next step is to determine the programming code to run this solenoid valve ON & OFF with input from the ESP32-Cam from the Face Recognition input. Moreover, the essential factor when uploading the program, the connection between IO0 and GND needs to be disconnected when the upload process is complete and press the reset button; this is done so that the HTTP address can appear on the serial monitor and the streaming test can be successful.

3. Method

The analytical method in this research is shown in 2 flowcharts in Figure 8 and Figure 9; the analysis is obtained from the following flowchart; in this research, it is not specific to the prototype results as shown in Figure 10, but more to the throughput analysis of the data generated by ESP32-Cam which determines the quality of data transmission, buffering process, error packet data generated from ESP32-Cam communication with the server, HTTP address in this research uses the address http://192.168.146.240/ Moreover, The software used in this research is Wireshark, which specifically analyzes the throughput generated from the communication between the ESP32-Cam and the server.

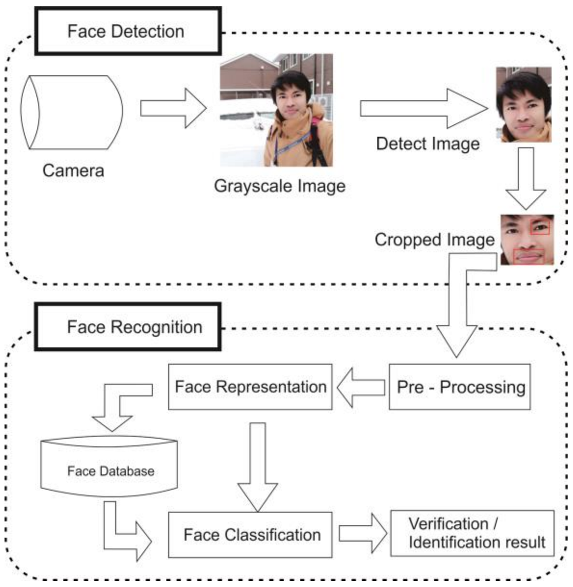


Figure 7. Face recognition system

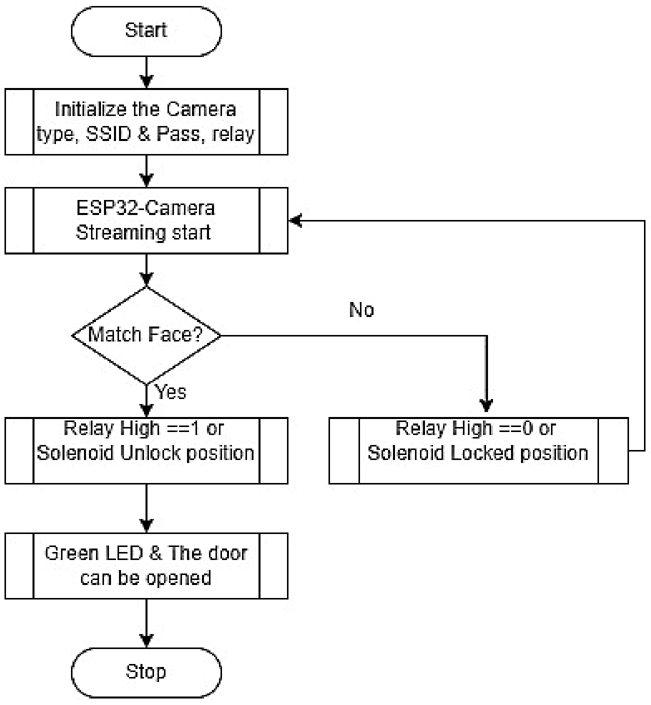

Start
Initialize the Camera type, SSID & Pass, relay
ESP32-Camera Streaming start
Match Face?
No
Yes
Relay High ==1 or Solenoid Unlock position
Relay High ==0 or Solenoid Locked position
Green LED & The door can be opened
Stop

Figure 8. Face recognition system for automatic door use ESP32-Cam

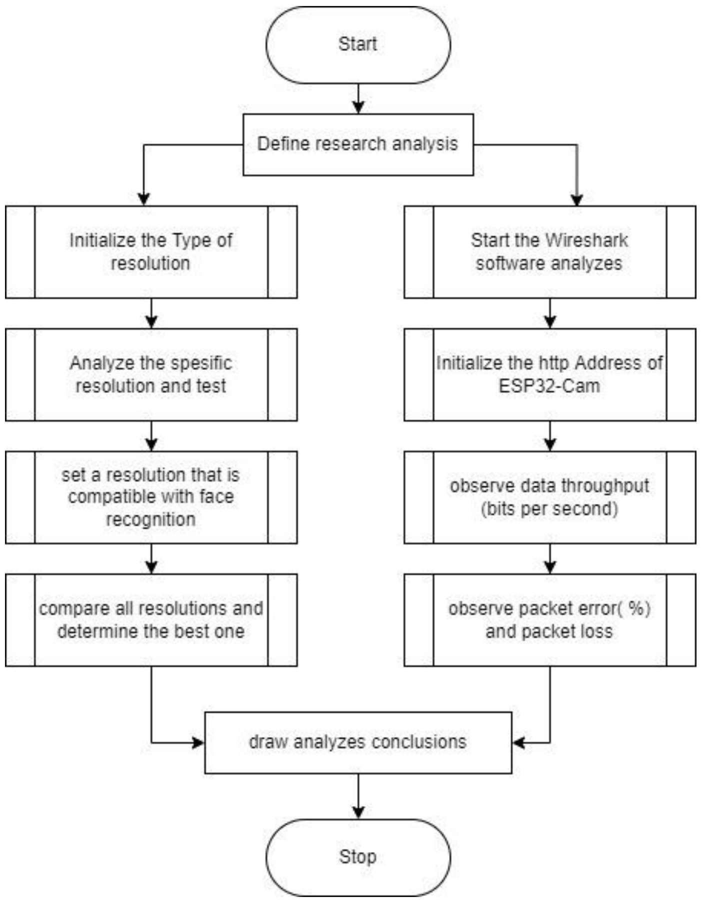
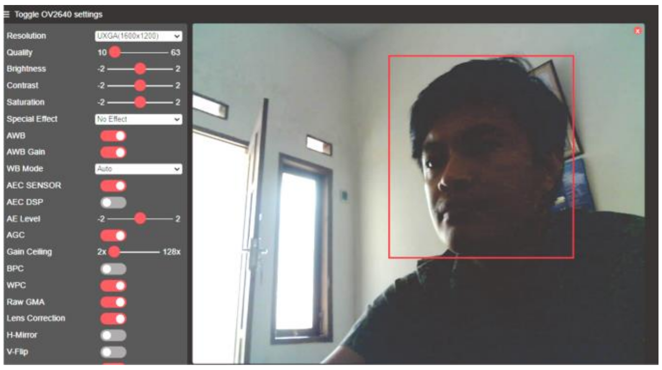

Start
Define research analysis
Initialize the Type of resolution
Analyze the spesific resolution and test
set a resolution that is compatible with face recognition
compare all resolutions and determine the best one
Start the Wireshark software analyzes
Initialize the http Address of ESP32-Cam
observe data throughput (bits per second)
observe packet error(%) and packet loss
draw analyzes conclusions
Stop

Figure 9. The analysis process in this research

4. Result and Discussion

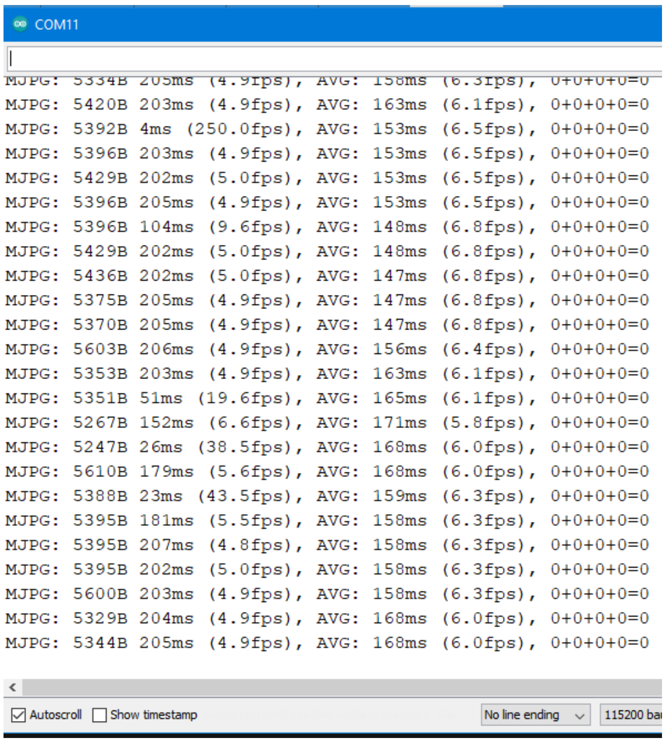
Each type of camera resolution used produces different speeds, referred to as frames per second (fps) and speeds with units of milliseconds (ms). In detail, the results of the comparison of the types of cameras used in the ESP-Cam can be seen in table 5. Table 5 shows the OV2640 speed of the camera in real-time; the comparison is seen from the resolution size, Speed of Motion (ms), and frames per second. The data in table 5 is taken in real-time when the camera detects a face. Indeed, the speed of motion (ms) varies, but it can be seen that the two resolutions of the OV2640 camera are UXVGA and SXVGA, which have the largest Average Speed of Motion (ms), namely 83 ms and 87 ms [Figure 11 & Figure 13]. As for other types of camera resolution, they are more or less the same.



Toggle OV2640 settings

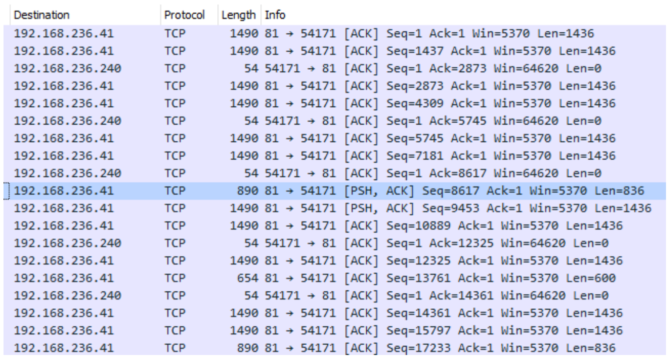
Resolution	UXGA(1600x1200)
Quality	10 — 63
Brightness	-2 — 2
Contrast	-2 — 2
Saturation	-2 — 2
Special Effect	No Effect
AWB	●
AWB Gain	●
WB Mode	Auto
AEC SENSOR	●
AEC DSP	○
AE Level	-2 — 2
AGC	●
Gain Ceiling	2x — 128x
BPC	○
WPC	●
Raw GMA	●
Lens Correction	●
H-Mirror	○
V-Flip	○

Figure 10. ESP32 Cam Streaming test



```
MJPG: 5334B 205ms (4.9fps), AVG: 158ms (6.3fps), 0+0+0+0=0
MJPG: 5420B 203ms (4.9fps), AVG: 163ms (6.1fps), 0+0+0+0=0
MJPG: 5392B 4ms (250.0fps), AVG: 153ms (6.5fps), 0+0+0+0=0
MJPG: 5396B 203ms (4.9fps), AVG: 153ms (6.5fps), 0+0+0+0=0
MJPG: 5429B 202ms (5.0fps), AVG: 153ms (6.5fps), 0+0+0+0=0
MJPG: 5396B 205ms (4.9fps), AVG: 153ms (6.5fps), 0+0+0+0=0
MJPG: 5396B 104ms (9.6fps), AVG: 148ms (6.8fps), 0+0+0+0=0
MJPG: 5429B 202ms (5.0fps), AVG: 148ms (6.8fps), 0+0+0+0=0
MJPG: 5436B 202ms (5.0fps), AVG: 147ms (6.8fps), 0+0+0+0=0
MJPG: 5375B 205ms (4.9fps), AVG: 147ms (6.8fps), 0+0+0+0=0
MJPG: 5370B 205ms (4.9fps), AVG: 147ms (6.8fps), 0+0+0+0=0
MJPG: 5603B 206ms (4.9fps), AVG: 156ms (6.4fps), 0+0+0+0=0
MJPG: 5353B 203ms (4.9fps), AVG: 163ms (6.1fps), 0+0+0+0=0
MJPG: 5351B 51ms (19.6fps), AVG: 165ms (6.1fps), 0+0+0+0=0
MJPG: 5267B 152ms (6.6fps), AVG: 171ms (5.8fps), 0+0+0+0=0
MJPG: 5247B 26ms (38.5fps), AVG: 168ms (6.0fps), 0+0+0+0=0
MJPG: 5610B 179ms (5.6fps), AVG: 168ms (6.0fps), 0+0+0+0=0
MJPG: 5388B 23ms (43.5fps), AVG: 159ms (6.3fps), 0+0+0+0=0
MJPG: 5395B 181ms (5.5fps), AVG: 158ms (6.3fps), 0+0+0+0=0
MJPG: 5395B 207ms (4.8fps), AVG: 158ms (6.3fps), 0+0+0+0=0
MJPG: 5395B 202ms (5.0fps), AVG: 158ms (6.3fps), 0+0+0+0=0
MJPG: 5600B 203ms (4.9fps), AVG: 158ms (6.3fps), 0+0+0+0=0
MJPG: 5329B 204ms (4.9fps), AVG: 168ms (6.0fps), 0+0+0+0=0
MJPG: 5344B 205ms (4.9fps), AVG: 168ms (6.0fps), 0+0+0+0=0
```

Figure 11. Real-time Serial Monitor Streaming camera



```
Destination          Protocol  Length Info
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=1 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=1437 Ack=1 Win=5370 Len=1436
192.168.236.240      TCP          54 54171 → 81 [ACK] Seq=1 Ack=2873 Win=64620 Len=0
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=2873 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=4309 Ack=1 Win=5370 Len=1436
192.168.236.240      TCP          54 54171 → 81 [ACK] Seq=1 Ack=5745 Win=64620 Len=0
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=5745 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=7181 Ack=1 Win=5370 Len=1436
192.168.236.240      TCP          54 54171 → 81 [ACK] Seq=1 Ack=8617 Win=64620 Len=0
192.168.236.41       TCP         890 81 → 54171 [PSH, ACK] Seq=8617 Ack=1 Win=5370 Len=836
192.168.236.41       TCP        1490 81 → 54171 [PSH, ACK] Seq=9453 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=10889 Ack=1 Win=5370 Len=1436
192.168.236.240      TCP          54 54171 → 81 [ACK] Seq=1 Ack=12325 Win=64620 Len=0
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=12325 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP         654 81 → 54171 [ACK] Seq=13761 Ack=1 Win=5370 Len=600
192.168.236.240      TCP          54 54171 → 81 [ACK] Seq=1 Ack=14361 Win=64620 Len=0
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=14361 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP        1490 81 → 54171 [ACK] Seq=15797 Ack=1 Win=5370 Len=1436
192.168.236.41       TCP         890 81 → 54171 [ACK] Seq=17233 Ack=1 Win=5370 Len=836
```

Figure 12. Source ESP-32 Cam Wireshark analyzes

Table VI. OV2640 Camera Resolution Comparison FPS And Speed of Motion

No	**Type of Resolution camera**	**Resolution size**	**Avg Speed of Motion (ms)**	**Avg fps**
1	UXVGA	1600x1200	111	9
2	SXVGA	1280x1024	87	11,5
3	XGA	1024x768	39	25,6
4	SVGA	800x600	39	25.6
5	VGA	640x480	43	23,3
6	QVGA	320x240	19	52,6
7	HQVGA	240x176	36	27,8
8	QQVGA	160x120	23	43,5
9	CIF	400x296	40	25

Several previous studies have shown that the performance of the ESP-32 Cam is poor. Judging by the low frames per second (fps) like 0.2-0.5 fps. by using CIF resolution during the video streaming process, then analysis using Wireshark is used to see the data bits sent to the server, the data throughput can be seen as Figure 12, Figure 14, the analysis process was carried out using Wireshark for several seconds.

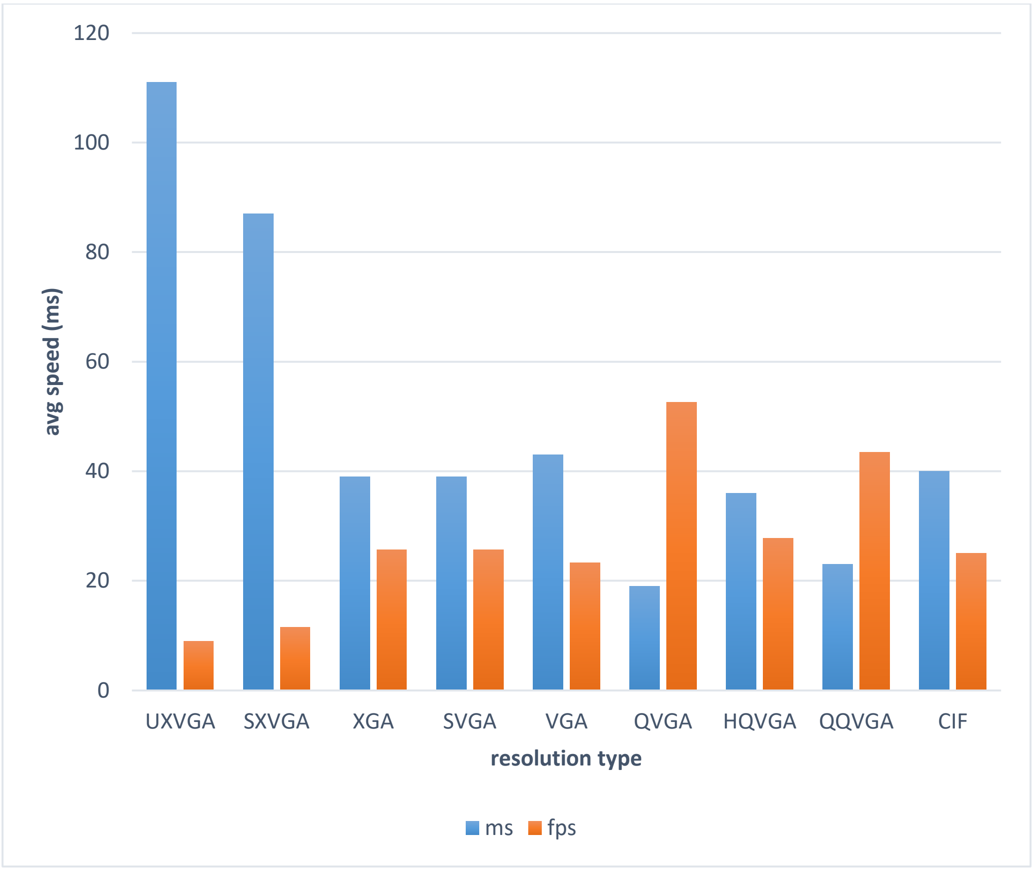


Figure 13. OV2640 Camera resolution comparison

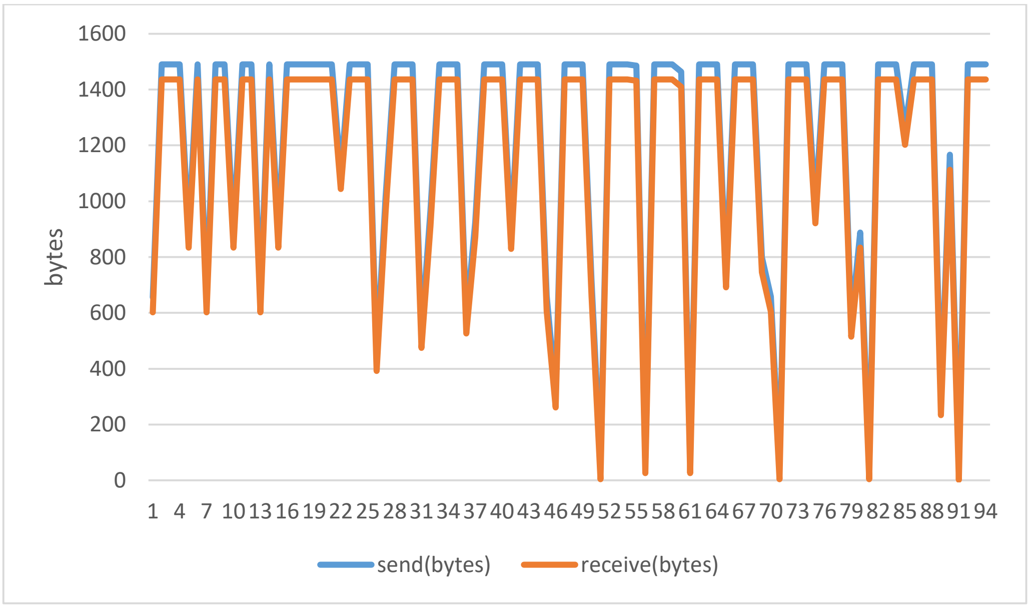


Figure 14. Throughput analyzer of ESP32-Cam in server

5. Conclusions and Suggestion

The ESP32-CAM with the camera type OV2640 can be used in streaming cameras and Face detection to control the actuator. In esp32-cam, not all types of cameras can be used, depending on the appropriate resolution and compatibility with the camera used for image recognition. UXVGA and SXVGA are the types of resolution with the highest average speed (ms) >80 ms out of the nine types of resolution that the OV2640 camera has on the ESP32-Cam. In comparison, the compatible types for Face recognition and Image recognition are CIF and QVGA, which have a speed of 40 ms and 19 ms.

Author Contributions: Conceptualization; Puput Dani Prasetyo Adi (P.D.P.A); methodology; (P.D.P.A), validation; (P.D.P.A); formal analysis; (P.D.P.A); investigation; (P.D.P.A); data curation; (P.D.P.A); writing—original draft preparation; (P.D.P.A); writing—review and editing; (P.D.P.A); visualization; (P.D.P.A); supervision; (Yuyu.W); project administration (P.D.P.A)**;** funding acquisition; (P.D.P.A), have read and agreed to the published version of the manuscript.

Acknowledgments: Thank you to fellow researchers who have helped complete the research process on this ESP32 Camera test. Thanks for BRIN helping complete this research to develop knowledge about Face Recognition.

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

References

1. A. Kaur, A. Jadli, A. Sadhu, S. Goyal, A. Mehra and Rahul, "Cloud Based Surveillance using ESP32 CAM," 2021 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE), 2021, pp. 1-5, doi: 10.1109/ITSS-IoE53029.2021.9615334.
2. Fransiska Sisilia Mukti, Puput Dani Prasetyo Adi, Dwi Arman Prasetya, Volvo Sihombing, Nicodemus Rahanra, Kristia Yuliawan, and Julianto Simatupang, "Integrating Cost-231 Multiwall Propagation and Adaptive Data Rate Method for Access Point Placement Recommendation" International Journal of Advanced Computer Science and Applications(IJACSA), 12(4), 2021, DOI: 10.14569/IJACSA.2021.0120494
3. Hemanth Kumar MS, ESP32-CAM for Face Mask Detection, February 2022, International Journal of Advanced Research in Science, Communication, and Technology (IJARSCT), DOI: 10.48175/IJARSCT-2509
4. I. Allafi and T. Iqbal, "Design and implementation of a low-cost web server using ESP32 for real-time photovoltaic system monitoring," 2017 IEEE Electrical Power and Energy Conference (EPEC), 2017, pp. 1-5, DOI: 10.1109/EPEC.2017.8286184.
5. N. Y. L. Venkata, C. Rupa, B. Dharmika, T. G. Nithin, and N. Vineela, "Intelligent Secure Smart Locking System using Face Biometrics," 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2021, pp. 268-273, DOI: 10.1109/RTEICT52294.2021.9573869.
6. O. Barybin, E. Zaitseva and V. Brazhnyi, "Testing the Security ESP32 Internet of Things Devices," 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 2019, pp. 143-146, DOI: 10.1109/PICST47496.2019.9061269.
7. Puput Dani Prasetyo Adi, et.al., "A Study of Programmable System on Chip (PSoC) Technology for Engineering Education", October 2020, Journal of Physics: Conference Series, Volume 1899, 2nd Workshop on Engineering, Education, Applied Sciences and Technology (WEAST) 2020 5 October 2020, Makassar, Indonesia, doi:10.1088/1742-6596/1899/1/012163

8. P. D. P. Adi, A. Kitagawa, and J. Akita, "Finger Robotic control use M5Stack board and MQTT Protocol-based," 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2020, pp. 1-6, DOI: 10.1109/ICITACEE50144.2020.9239170.
9. P. D. P. Adi and A. Kitagawa, "Performance Evaluation of Low Power Wide Area (LPWA) LoRa 920 MHz Sensor Node to Medical Monitoring IoT Based," 2020 10th Electrical Power, Electronics, Communications, Controls, and Informatics Seminar (EECCIS), 2020, pp. 278-283, DOI: 10.1109/EECCIS49483.2020.9263418.
10. P. D. P. Adi, A. Kitagawa, D. A. Prasetya and A. B. Setiawan, "A Performance of ES920LR LoRa for the Internet of Things: A Technology Review," 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT), 2021, pp. 1-7, DOI: 10.1109/EIConCIT50028.2021.9431912.
11. P.D.P. Adi, A.Kitagawa, D.A. Prasetya, R.Arifuddin, S.Yoseph, "LoRaWAN Technology in Irrigation Channels in Batu Indonesia", Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, Vol. 7, No. 3, December 2021, pp. 522-538, ISSN: 2338-3070, DOI: 10.26555/jiteki.v7i3.22258, doi:10.26555/jiteki.v7i3.22258
12. R. Chauhan et al., "Design of Robotic Snake with ESP 32 CAM and Arduino," 2021 19th OITS International Conference on Information Technology (OCIT), 2021, pp. 6-9, doi: 10.1109/OCIT53463.2021.00013.
13. Umara Zafar, Mubeen Ghafoor,Tehseen Zia,Abdullahi Mohamud Sharif, Face recognition with Bayesian convolutional networks for robust surveillance systems, January 2019, EURASIP Journal on Image and Video Processing 2019(1), DOI: 10.1186/s13640-019-0406-y
14. Y. A. Liani et al., "The Broiler Chicken Coop Temperature Monitoring Use Fuzzy Logic and LoRAWAN," 2021 3rd International Conference on Electronics Representation and Algorithm (ICERA), 2021, pp. 161-166, doi: 10.1109/ICERA53111.2021.9538771.
15. M. Niswar et al., "Performance evaluation of ZigBee-based wireless sensor network for monitoring patients' pulse status," 2013 International Conference on Information Technology and Electrical Engineering (ICITEE), 2013, pp. 291-294, doi: 10.1109/ICITEED.2013.6676255.