Pembangunan Sistem Pengawasan Cerdas Dengan Visualisasi 3D

Development Of Smart Surveillance System With 3D Visualization

1st Arsil Qodryantha
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
arsillite@student.telkomuniversity.ac.id

2nd Rikman Aherliwan Rudawan
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
rikman@telkomuniversity.ac.id

3rd Anang Sularsa
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
ananks@telkomuniversity.ac.id

Abstrak—Industri 4.0 adalah evolusi teknologi dimana setiap perangkat dapat saling berkomunikasi menggunakan sebuah jaringan internet tanpa dibatasi oleh jarak. Salah satu contohnya adalah *CCTV*. *CCTV (Closed Circuit Television)* adalah kamera yang memiliki kegunaan untuk mengintai atau merekam situasi suatu tempat demi kepentingan keamanan. Di era ini banyak kamera *CCTV* yang terintegrasi dengan jaringan internet, memungkinkan pengguna dapat mengaksesnya dari mana saja, ini tentu merupakan suatu hal yang baik. Namun di sisi lain, hal ini dapat menyebabkan orang lain juga memiliki peluang untuk mendapatkan akses yang sama karena faktor seperti keamanan pada *router* dan *malicious code* yang mungkin ada pada sebuah perangkat. Untungnya pada era tersebut masyarakat memiliki akses terhadap perangkat yang memungkinkan semua orang dapat merancang suatu sistem tanpa ada campur tangan dari pihak lain. *ESP32Camera* merupakan sebuah modul berbasis *ESP32* yang memiliki *onboard* camera *2MP* yang berguna untuk berbagai macam aplikasi *IOT* yang membutuhkan adanya fitur *vision.* Selain modul tersebut, digunakan juga *MAX9814*, sebuah modul mikrofon *low noise* yang berguna sebagai *trigger* dalam melakukan proses penangkapan gambar yang selanjutnya dikirim ke komputer ketika ada kenaikan suara. Fitur trigger ini dapat mengeliminasi penggunaan *doorbell* pada rumah. Sistem ini juga terintegrasi dengan *frontend* berbasis *3D WebGL* yang menampilkan visualisasi rumah pengguna.

Kata Kunci—*ESP32Camera, MAX9814, AI, 3D WebGL*

Abstract—Industry 4.0 is a technological evolution where every device can communicate with each other using an internet network without being limited by distance. One example is CCTV. CCTV (Closed Circuit Television) is a camera that has a function to spy or record the situation of a place for security purposes. In this era, many CCTV cameras are integrated with the internet network, allowing users to access them from anywhere, this is certainly a good thing. But on the other hand, this can cause other people to also have the opportunity to get the same access due to factors such as security on the router and malicious code that may exist on a device. Fortunately, in that era, people had access to tools that allowed everyone to design a system without any intervention from other parties. ESP32Camera is an ESP32-based module that has a 2MP onboard camera that is useful for various IoT applications that require vision features. In addition to these modules, the MAX9814 is also used, a low noise microphone module that is useful as a trigger in the process of capturing images which are then sent to the computer when there is an increase in sound. This trigger feature can eliminate the use of doorbells at home. The system is also integrated with a 3D WebGL-based frontend that displays visualizations of the user's home.

Keywords—*ESP32Camera, MAX9814, AI, 3D WebGL*

I. PENDAHULUAN

CCTV (Closed Circuit Television) adalah kamera yang memiliki kegunaan untuk mengintai, mengawasi maupun merekam situasi suatu tempat untuk kepentingan keamanan. Kamera tersebut akan memberikan sebuah gambar dari suatu tempat melalui sebuah transmisi sinyal baik kabel ataupun secara *wireless* menggunakan jaringan *WiFi*. Penggunaan kamera pengintai ini bisa sangat penting untuk kebutuhan secara umum baik di lingkungan publik, pribadi, dan bisnis. Kamera pengintai ini juga merupakan perangkat yang penting untuk dipasang di area-area yang membutuhkan pengawasan

lebih [1]. Namun, karena harganya yang cukup mahal, tidak semua kalangan masyarakat dapat membeli dan memanfaatkannya walaupun sangat dibutuhkan. Selain itu, kamera yang beredar di pasaran tentunya tidak dapat dimodifikasi sesuai dengan keinginan dan kebutuhan meskipun sebagai pemilik perangkat, dikarenakan adanya faktor syarat dan ketentuan penggunaan.

ESP32 Camera merupakan *ESP32-based-module* yang memiliki *onboard camera* dengan resolusi *2MP* [2]. Modul tersebut cocok digunakan untuk sebuah sistem *embedded* yang dimana memerlukan fitur "*vision*", contohnya adalah pada alat pengawasan rumah seperti proyek ini. Karena modul ini menggunakan *ESP32* sebagai *MCU (Microcontroller)*, transmisi data dapat dengan mudah dilakukan menggunakan fitur *WiFi* dari *MCU* tersebut sebagai jalur komunikasi ke sebuah *server* pada komputer menggunakan protokol seperti *HTTP*, *TCP*, dsb. *MAX9814* adalah modul mikrofon *low-noise* yang memiliki *built-in AGC (Automatic Gain Control)* [3]. Modul ini dapat digunakan pada sistem *embedded* yang merupakan *sound-reactive* seperti proyek ini, sebagai *trigger* dalam melakukan proses penangkapan gambar yang selanjutnya akan dikirim ke komputer jika modul ini menangkap adanya suara yang melebihi *volume-threshold* yang telah ditentukan pada program. Kedua Alat *IOT* tersebut merupakan alat yang *budget-friendly,* menggunakan *resource* rendah dan memiliki dimensi yang kecil.

Terdapat kekurangan pada kamera *cctv* pada umumnya seperti, tidak mudah diintegrasikan dengan aplikasi *custom*, cukup mahal dan memiliki dimensi yang cukup besar. Maka dari itu penulis membuat sistem pengawasan rumah yang memiliki kapabilitas seperti *CCTV* menggunakan alat *IOT* yang dapat diintegrasikan dengan sistem lain seperti kecerdasan buatan dan *visualisasi* yang berbasis *3D WebGL* agar memiliki nilai estetis dan *entertaining*, tentunya dapat dikustomisasi sesuai keinginan. Sistem dikembangkan menggunakan metode *modified waterfall* yang memberikan pengembang dapat kembali ke fase-fase sebelumnya untuk melakukan sebuah perbaikan atau perubahan.

II. KAJIAN TEORI

A. Kajian Produk Sejenis

Adapun produk sejenis yang ditemukan ada 2 yakni dijelaskan pada tabel berikut:

TABEL 2.1
KOMPARASI PRODUK SEJENIS

No	Pembanding	Sistem pertama	Sistem kedua	Sistem Usulan
1	Metode Transmisi	*ESP32Cam* memberikan akses *server HTTP* yang dapat diakses semua orang	*ESP32Cam* memberikan akses *server HTTP* yang dapat diakses semua orang	*ESP32Cam* melakukan akses koneksi terhadap *server TCP* pada komputer
2	*Image Capturing*	Manual via *web*	Deteksi inframerah (*PIR)*	Deteksi *threshold* suara
3	*Object Detection*	-	-	Ada
4	*Gate Detection*	-	-	Ada
5	*Face Recognition*	Ada	-	-
6	*GUI/Frontend*	Konvensional	Konvensional	Visualisasi *WebGL*

Sistem pertama yang dibandingkan merupakan *example sketch* yang disediakan

oleh *espressif*, *sketch* tersebut dapat didapatkan pada *source code github espressif* dan juga akan tersedia apabila telah menambahkan *board ESP32* pada *Arduino IDE* (jika menggunakan *Arduino IDE*). *Sketch* tersebut juga digunakan pada proyek akhir dari Aprylia, seorang mahasiswi Universitas Sumatera Utara dengan judul *SMART HOUSE* BERBASIS *WEB SERVER* MENGGUNAKAN *ESP 32* SEBAGAI *DOOR LOCK* MENGGUNAKAN *FACE LOCK* pada tahun 2020 [4]. Aprylia mengintegrasikan fitur *sketch* tersebut yang berupa *Face Recognition* untuk membuka atau menutup pintu menggunakan *Selenoid Door Lock*. Sistem serupa juga dikembangkan oleh Andrew, seorang berkebangsaan spanyol yang mem-*posting* proyeknya itu di *platform youtube* [5]. Sistem kedua didapatkan dari jurnal yang ditulis oleh Andi Setiawan dan Ade Irma Purnamasari dari IKATAN AHLI INFORMATIKA INDONESIA (IAII) dengan judul "Pengembangan *Passive Infrared Sensor (PIR) HC-SR501* dengan *Microcontrollers ESP32-CAM* Berbasiskan *Internet of Things (IoT)* dan *Smart Home* sebagai Deteksi Gerak untuk Keamanan Perumahan" dimana menggunakan sensor *(PIR)* sebagai *trigger* dalam melakukan penangkapan gambar yang selanjutnya dikirim ke komputer [6].

B. *ESP32 Camera*

ESP32 Camera adalah sebuah modul berbasis *ESP32* yang memiliki *onboard-camera* yang berguna untuk berbagai macam aplikasi *IoT*, prototipe, dan proyek *DIY*. Modul ini memiliki *PSRAM* sebanyak *4MB* yang dapat digunakan sebagai penyimpanan gambar secara sementara yang nantinya dapat dikirim ke sebuah *server*. Selain itu, modul ini juga dilengkapi dengan *slot microSD* yang dapat digunakan untuk menyimpan gambar secara permanen yang dimana juga berfungsi sebagai alternatif mengirimkan data ke *server* menggunakan koneksi *WiFi*. Modul ini mengintegrasikan *WiFi*, *Bluetooth* tradisional, dan *BLE (Bluetooth Low Energy)* dengan *dual-core CPU LX6 32-bit* dengan kecepatan *80MHz – 240MHz*. Kamera pada modul ini adalah *OV2640* yakni sensor kamera bertegangan rendah buatan *OmniVision* yang memiliki resolusi maksimum *2MP (UXGA)* serta *Image processor* dalam satu *chip* (*SOC*) [7].

C. *MAX9814*

MAX9814 adalah sebuah modul mikrofon *electret* yang memiliki *amplifier* untuk meningkatkan suara dalam situasi di mana kenyaringan *audio* tidak dapat diprediksi. Modul ini merupakan modul yang memiliki *AGC (Automatic Gain Control)* yang dimana suara keras di sekitarnya akan diheningkan dan suara yang rendah dan jauh akan diperkuat. Dengan adanya fitur ini tentu dapat mengurangi *noise floor* dan membuat *sound detection* lebih akurat [8].

D. *PlatformIO*

PlatformIO adalah sebuah *framework* yang dapat membantu melakukan pembangunan aplikasi pada *embedded system. PlatformIO* dapat digunakan pada banyak *text editor* seperti *VS Code, Atom,* dan *Sublime Text* yang memungkinkan pengembang dapat menggunakan fitur-fitur yang ada pada masing-masing *text editor* [9]. Contohnya adalah *autocomplete*, *breakpoint* dan *debugging, goto reference*, *linting*, dll.

E. *WebGL*

WebGL adalah sebuah *Wrapper OpenGL* yang berbasiskan *javascript. WebGL* memberikan kemudahan dalam melakukan *rendering 3D* dan *2D graphics* langsung pada browser. *WebGL* pada dasarnya adalah *Rasterization Engine* dan bukan *3D library* yang dimana ia membuat atau menggambar titik, garis, dan segitiga berdasarkan kode atau instruksi yang diberikan. Sedangkan *3D library* memberikan kemudahan dalam hal 3D baik itu data posisi suatu obyek, material obyek, pencahayaan, dsb. *WebGL* berjalan pada *GPU* di komputer. Oleh karena itu, kode atau instruksi yang diberikan harus dapat dimengerti oleh *GPU* yakni berupa 2 fungsi yang disebut *vertex shader* dan *fragment shader* dan keduanya biasanya ditulis dalam bahasa *GLSL (GL Shader Language)*. Ada banyak *3D framework* dan *library* yang telah menggunakan *WebGL* termasuk namun tidak terbatas pada *Unity*, *Unreal*, *Three.js*, dan *Babylon.js*. Pada waktu penulisan PA ini, beberapa browser yang telah memiliki *official full support* untuk *WebGL 2.0* (versi terbaru)

yakni *Chrome* 56, *Firefox* 51, *Safari* 10, *Edge* 16, dan *Opera* 44[10]-[12].

F. *PyTorch*

PyTorch merupakan sebuah *library* yang dapat digunakan dalam *machine learning*, *deep learning*, dan bahkan *natural language processing* yang dikembangkan oleh *Facebook AI* (sekarang *Meta AI*) pada tahun 2016 [13]. *PyTorch* pada dasarnya adalah *compiled library /shared object* yakni sebuah *file* yang isinya adalah fungsi-fungsi yang dapat di eksekusi pada masing-masing sistem operasi atau secara *native*. Format *file* berbeda-beda pada tiap sistem operasi, yaitu *.dll* pada windows dan *.so* pada linux. *PyTorch* dikembangkan menggunakan *c++*, namun *PyTorch* memiliki *frontend API* berbasis *python* yang sangat mempermudah penggunaan *library* tersebut. Selain itu, *PyTorch* juga memiliki *frontend API c++* bernama *LibTorch* yang dimana digunakan pada proyek ini, namun pada bahasa pemrograman *Rust* menggunakan *wrapper* bernama *tch-rs* (https://github.com/LaurentMazare/tch-rs).
Model yang dapat di-*load* pada *LibTorch* adalah *Torchscript* yang pada dasarnya adalah kumpulan fungsi yang dapat dieksekusi tanpa harus ada ketergantungan pada *python* atau *python environment* contohnya pada aplikasi *c++* atau *rust* (seperti proyek ini).

G. *Roboflow*

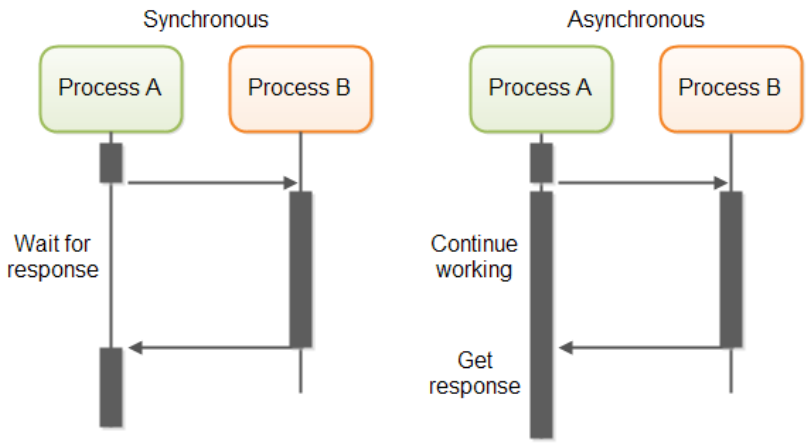
Roboflow merupakan sebuah *platform* yang dapat digunakan dalam melakukan sebuah pembuatan *dataset* pada *computer vision* yakni dengan melakukan sebuah *image preprocessing* seperti orientasi gambar, ukuran gambar, kontras, dan augmentasi data [14]. Dalam membuat suatu *dataset* pada *Roboflow* dapat dilakukan dengan beberapa langkah seperti:

1. *Upload* gambar
2. *Labeling* gambar
3. *Train/Test split*
4. *Preprocessing*
5. Augmentasi
6. *Generate*
7. *Train (opsional)*

H. *Rust*

Rust merupakan *Statically Typed Programming Language* yang artinya tipe variabel diketahui pada waktu kompilasi, dalam kata lain *developer* harus menentukan sendiri tipe dari tiap-tiap variabel yang ada, selain itu tipe variabel tidak dapat diubah sepanjang program berjalan, kecuali dengan melakukan *variable shadowing* sama seperti *c*, *c++*, dan *java*. *Rust* juga tidak memiliki *garbage collection* yang berarti *developer* harus melakukan manajemen memori secara manual, namun yang membuat *Rust* berbeda adalah, *Rust* menggunakan *borrow checker* yang dimana secara desain *Rust* akan memastikan tidak ada *reference* yang dapat hidup lebih lama dari data yang dituju. Hal itu dilakukan pada saat kompilasi, tentunya dapat mengurangi *bug* yang disebabkan oleh kesalahan dalam melakukan manajemen memori [15]-[16]. Salah satu *bug* yang disebabkan oleh manajemen memori yang buruk adalah *double free error,* yang dimana melakukan pelepasan alokasi memori dua kali. *Rust* juga memiliki *built-in async/await* seperti yang dapat digunakan pada *javascript.*

I. *Asynchronus Programming*



GAMBAR 2.1
KOMPARASI TEKNIK EKSEKUSI PROSES PADA PROGRAM

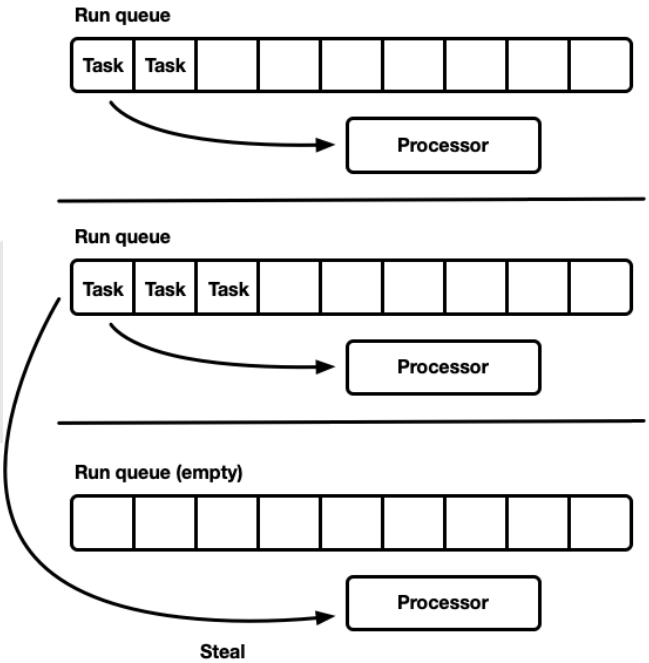
Asynchronus programming adalah sebuah teknik yang dapat membuat eksekusi kode dapat dilakukan secara bersamaan (*concurrent).* Perhatikan gambar 2.1. Dalam pemrograman komputer, operasi asinkron adalah suatu proses yang dapat beroperasi secara independen dari proses lain, sedangkan operasi sinkron berarti bahwa proses akan dieksekusi hanya jika beberapa proses lain sudah diselesaikan atau diserahkan [17]. Penggunaan asinkron umumnya ada pada proses *networking* dan juga *GUI* aplikasi yang dimana pada *networking* melakukan penerimaan data dari dua perangkat atau lebih secara bersamaan. Setiap transmisi yang diterima akan dilakukan proses yang dilakukan secara independen agar dapat menerima transmisi lain meskipun proses pada transmisi pertama belum selesai. Sedangkan pada *GUI* aplikasi contohnya adalah, *GUI* yang menampilkan *loading animation* saat terjadi sebuah proses *data loading* seperti aset audio, gambar, dsb.

J. *Tokio*

Tokio merupakan sebuah *async library* yang ditulis menggunakan bahasa pemrograman *Rust*. *Tokio* memberikan sebuah runtime yang menjadi sebuah komponen utama dalam menjalankan kode secara asinkronus. Selain itu Tokio juga memberikan beberapa komponen seperti:

1. *Runtime (multi thread).*
2. Nama *API* yang mirip seperti *std library.*
3. Ekosistem *library* besar.

Inti dari semua *runtime* asinkronus (baik itu pada bahasa *Javascript*, *Rust,* dll) adalah *event loop* yang juga disebut prosesor. Setiap *event-loop* memiliki *task queue* atau antrian tugas yang menunggu untuk diselesaikan. Perhatikan gambar 2.2, *Runtime* pada *Tokio* merupakan *work-stealing runtime* yang artinya setiap prosesor dapat mencuri tugas/*task* dalam *queue* prosesor lain jika sedang kosong (tidak memiliki tugas/*task*).

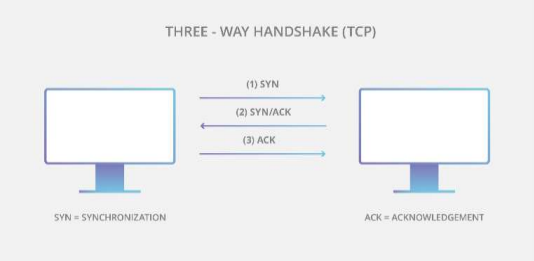


GAMBAR 2.2
RUNTIME TOKIO

Dengan fitur *built-in async/await Rust*, penulisan atau pembuatan aplikasi secara asinkron dapat dilakukan dengan sangat mudah karena *Library Tokio* mengikuti

penamaan *API* yang mirip dengan *std library* yang dimana memungkinkan mengonversi kode yang ditulis secara sinkronus menggunakan *std library* menjadi kode yang ditulis dengan *Tokio* [18]-[19].

K. *TCP*



GAMBAR 2.3
HANDSHAKE PADA *TCP*

TCP adalah salah satu protokol yang digunakan dalam melakukan pertukaran data pada internet. *TCP* bekerja sama dengan *IP (Internet Protocol)* dan menyediakan layanan transmisi yang dapat diandalkan untuk antar proses dengan menggunakan *network layer* yang disediakan oleh *IP*. *TCP* berfungsi sebagai komponen yang menangani permintaan paket dan memeriksa adanya kesalahan pada paket. Protokol ini mengirim paket dengan menggunakan format *TCP Segment* yang dimana ia memiliki *header* dan data. Perhatikan gambar 2.3, *TCP/IP* mirip dengan orang mengirim pesan yang ditulis pada sebuah *puzzle.* Ada proses yang harus dilakukan saat memulai transmisi data pada *TCP* yakni *Handshake*. Pertama, klien akan mengirimkan SYN (sinkronisasi) / permintaan inisialisasi ke *server* untuk memulai transmisi. Kemudian *server* akan mengirimkan *SYN-ACK* untuk mengonfirmasi proses. Setelah itu klien akan mengirimkan *ACK* lalu pesan akan dikirim. Pesan tersebut dipecah menjadi beberapa bagian seperti halnya sebuah *puzzle,* yang nantinya dikirim melalui rute yang berbeda hingga tiba di tujuan yang sama, setelah itu *TCP* akan merakit *puzzle* tersebut sehingga berada dalam urutan yang benar dan menjadi pesan yang sebenarnya [20].

L. *WebSocket*

WebSocket adalah protokol yang dibangun di atas *TCP*, dirancang untuk dapat melakukan komunikasi *duplex* atau dua arah antara *web server* dan *browser*, protokol ini memberikan koneksi *persistent* yang menyebabkan *latency overhead* yang rendah dikarenakan tidak menggunakan *pattern request-response* seperti *HTTP*. Protokol ini memungkinkan mengirim pesan, baik dari *server* ataupun klien di waktu kapan saja tanpa harus menunggu *server* membalas. Penggunaan *WebSocket* umumnya ada pada *real-time application* seperti aplikasi *chatting* ataupun *game*. *WebSocket* tidak dianjurkan untuk digunakan jika hanya melakukan transmisi data satu kali [21].

III. HASIL DAN PEMBAHASAN

Proyek akhir ini memiliki 3 komponen utama yang terdiri dari Kamera, *CLI*, dan *GUI.* Pada tiap komponen memiliki kebutuhan yang juga berbeda, yakni sebagai berikut:

A. Kamera

Kamera pada sistem ini merupakan *ESP32 Camera* yang merupakan sebuah modul *ESP32* dengan *built-in* kamera. Pada sistem ini, komponen tersebut bertugas dalam mengirim gambar ke sebuah *server TCP* (transmisi data) jika alat ini membaca kenaikan level volume dari *MAX9814* yang melebihi *threshold* yang ditentukan. *ESP32 Camera* ini tidak akan selalu terkoneksi ke jaringan *WiFi*, tujuannya adalah agar *power consumption* dari *ESP32 Camera* tetap rendah ketika *idle* dan memungkinkan penggunaan baterai jika menginginkan sedikit portabilitas dalam penempatan kamera. Kebutuhan lengkapnya dapat dilihat pada tabel 3.1 di bawah.

TABEL 3.1
KEBUTUHAN FUNGSIONAL KAMERA

No	Kebutuhan Fungsional
1	Dapat menginisialisasi modul kamera
2	Mendeteksi adanya *psram* atau tidak

3	Dapat membaca level audio dari *MAX9814*
3	Dapat melakukan *capturing* sesuai *frame buffer*
4	Melakukan koneksi ke *server TCP*
5	Mengirim data ke *server TCP*

B. CLI

CLI (Command Line Interface) pada sistem ini merupakan komponen yang paling utama pada sistem ini, karena memiliki fungsi yaitu menjalankan sebuah *server* untuk kamera agar dapat mengirim gambar, melakukan analisa kecerdasan buatan seperti deteksi objek dan deteksi keadaan pagar, dan memberikan *GUI* yang di-*host* sebagai *web application.* *CLI* dikembangkan menggunakan bahasa pemrograman *Rust* agar mendapatkan keuntungan seperti *memory safety*, performanya cepat, dan *concurrency.* Kebutuhan lengkapnya dapat dilihat pada tabel 3.2 di bawah.

TABEL 3.2
KEBUTUHAN FUNGSIONAL *CLI*

No	Kebutuhan Fungsional
1	Menjalankan *server TCP*
2	Dapat Menjalankan *server HTTP* dan *WebSocket* untuk *GUI*
3	Dapat menerima data dari klien
4	Dapat meng-*encode* data *ascii* menjadi *hex*
5	Dapat membedakan data gambar atau bukan
6	Deteksi objek dan deteksi pagar
7	Dapat menyimpan gambar
8	Dapat Membuat notifikasi pada sistem
9	Dapat Mengirim hasil analisa ke *GUI* menggunakan *WebSocket*

C. GUI

GUI pada sistem ini merupakan sebuah visualisasi yang merefleksikan analisa yang didapatkan dari kecerdasan buatan pada aplikasi *CLI*. *GUI* ini menggunakan *library* berbasis *WebGL* yakni *Three.Js* dalam memberikan visualisasi yang dapat diakses melalui *browser*. Visualisasi yang ditampilkan adalah sebuah aset tiga dimensi dari rumah klien. Kebutuhan lengkapnya dapat dilihat pada tabel 3.3 di bawah.

TABEL 3.3
KEBUTUHAN FUNGSIONAl *GUI*

No	Kebutuhan Fungsional
1	Dapat menampilkan *asset*
2	Dapat melakukan navigasi
3	Melakukan Koneksi ke *server WebSocket*
4	Dapat merubah visual sesuai dengan analisa AI

D. Pengembangan Sistem

Adapun beberapa perangkat keras dan juga perangkat lunak dalam mengembangkan

sistem ini terdapat pada tabel 3.4 - 3.6 di bawah ini.

TABEL 3.4
KEBUTUHAN PERANGKAT KERAS UNTUK PENGEMBANGAN

No	Perangkat Keras	Spesifikasi Minimum
1	*CPU*	*Any x86_64 @ 3.10 GHz*
2	*RAM*	*8 GB*
3	*GPU*	*Intel HD 3000*
4	*Storage*	*8 GB*

TABEL 3.5
KEBUTUHAN PERANGKAT LUNAK UNTUK PENGEMBANGAN

No	Perangkat Lunak	Keterangan
1	*VSCode*	*Text editor* yang dipakai untuk mengembangkan sistem secara keseluruhan
2	*PlatformIO*	*Plugin VSCode* untuk pengembangan *embedded system* yakni kamera pada proyek ini
3	*Rust Analyzer*	*Plugin VSCode* untuk membantu pengembangan aplikasi *rust* yakni *CLI* pada proyek ini
4	*GitHub*	*Version Control*
5	*Blender*	Aplikasi untuk pembuatan *3D model*

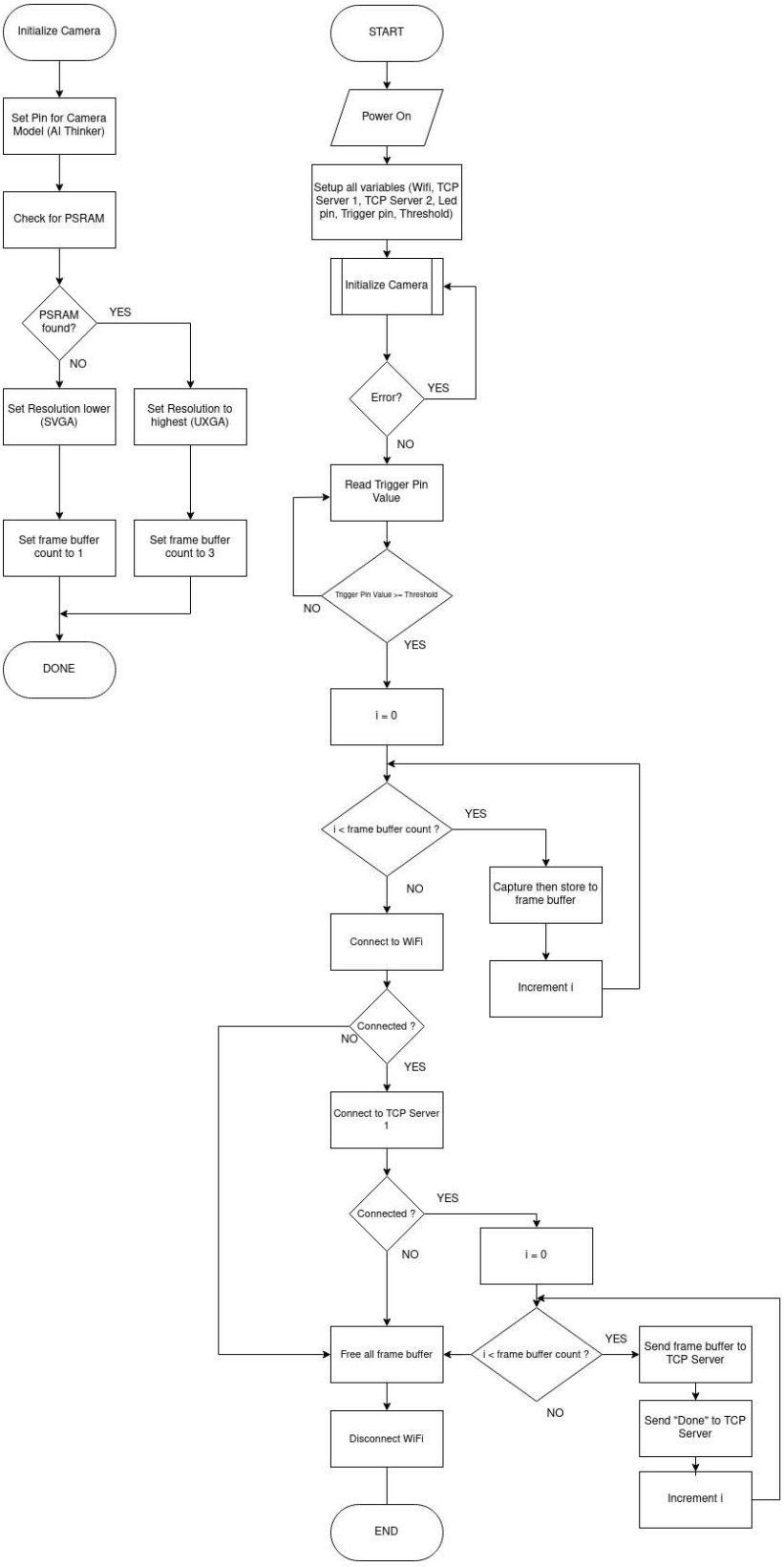
E. Implementasi Sistem

TABEL 3.6
KEBUTUHAN PERANGKAT KERAS UNTUK IMPLEMENTASI

No	Perangkat Keras	Keterangan
1	*ESP32 Camera*	*MCU (Microcontroller)* sekaligus kamera
2	*MAX9814*	Modul pendeteksi level suara
3	*Adapter / Charger 5v 1a*	*Power supply* untuk *MCU*
4	Kabel	Alternatif antena eksternal yang dapat di *solder* ke antena bawaan *ESP32*

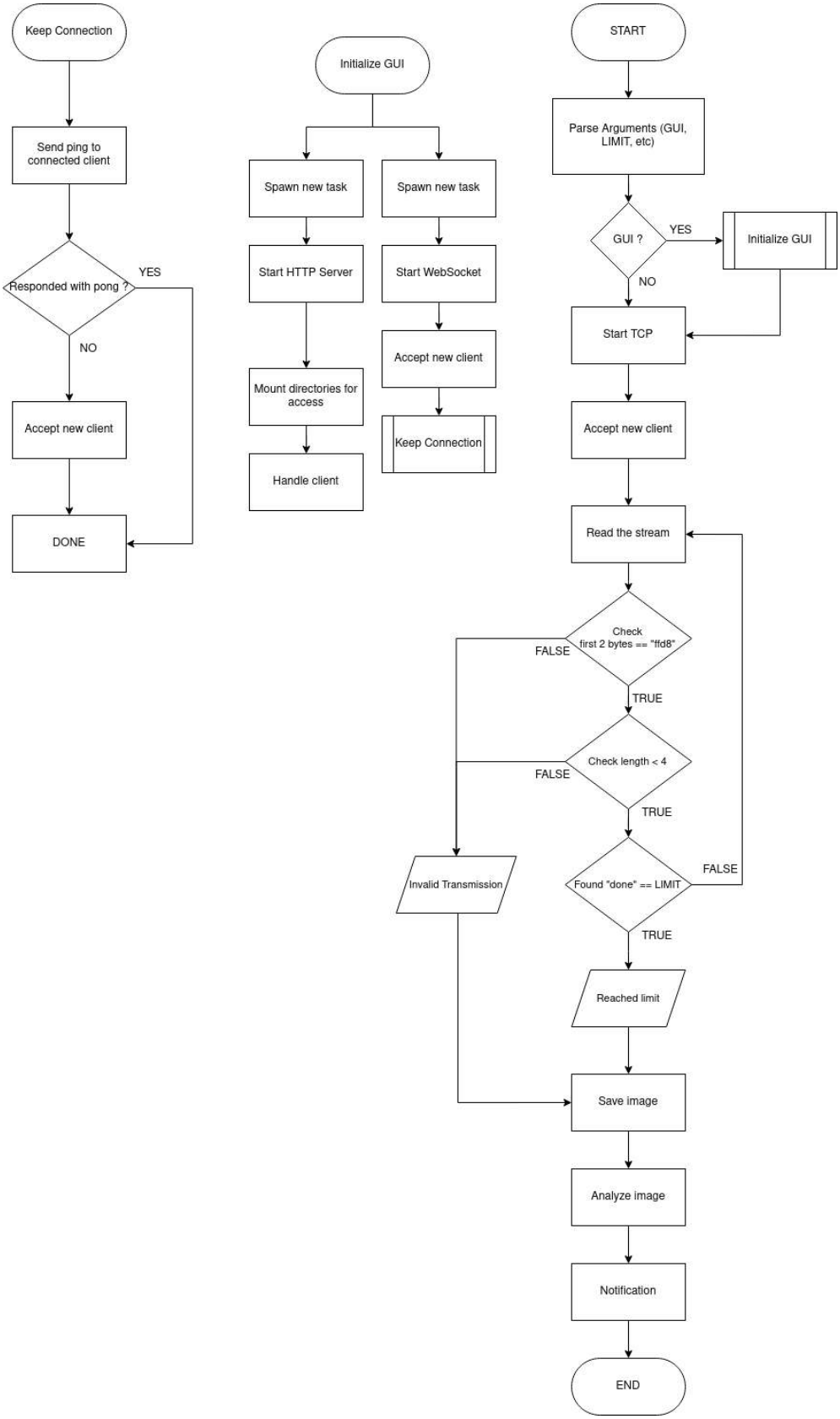
3.6 Perancangan Sistem

Berikut ini merupakan *flowchart* dari komponen kamera yang terdapat pada gambar 3.1 di bawah.



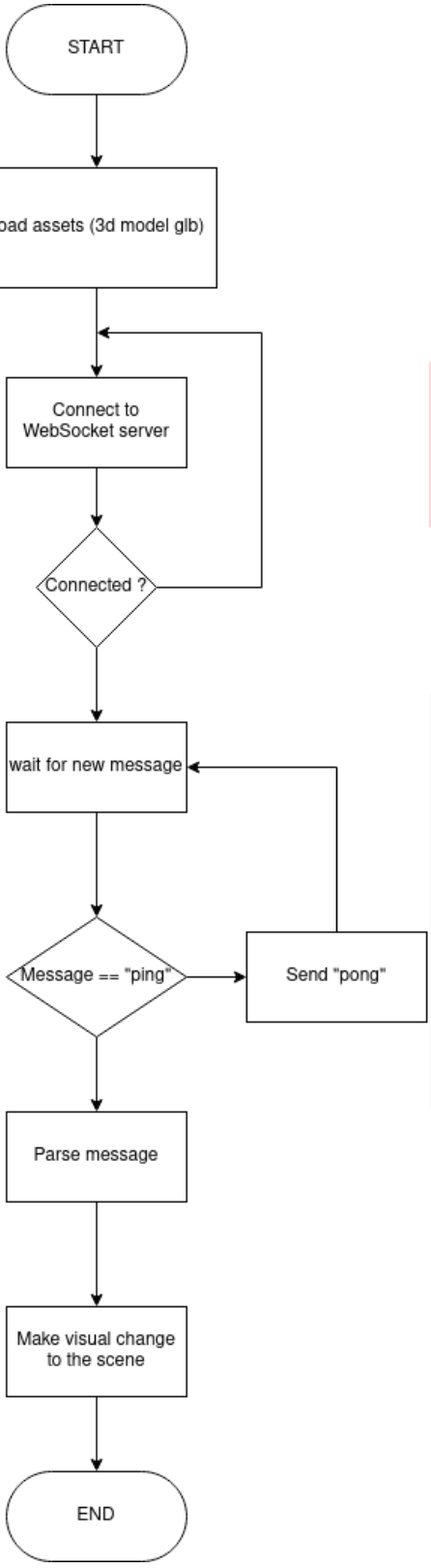
GAMBAR 3.1
FLOWCHART KAMERA

Berikut ini merupakan *flowchart* dari komponen *CLI* yang terdapat pada gambar 3.2 di bawah ini.



GAMBAR 3.2
FLOWCHART CLI

Berikut ini merupakan *flowchart* dari komponen *GUI* yang terdapat pada gambar 3.3 di bawah ini.



GAMBAR 3.3
FLOWCHART GUI

G. Pengujian Sistem

Berikut ini merupakan tabel pengujian pada setiap sistem.

TABEL 3.7
PENGUJIAN KAMERA

No	Parameter	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Inisialisasi kamera	Kamera dapat diinisialisasikan tanpa *crash*.	Kamera pada *ESP32* berhasil diinisialisasi dengan benar, karena terdapat *error handling* pada proses spesifik	Valid
2	Membaca sound level dari *MAX9814*	Memberikan *value* dari 0 sampai 3300	Program berhasil eksekusi baris pada kondisi *"if threshold"*	Valid
3	Menangkap gambar	Program dapat menangkap gambar sesuai banyak *frame buffer*	Program berhasil menangkap gambar sesuai banyak *frame buffer*	Valid
4	Koneksi *server*	Program dapat melakukan koneksi ke *server TCP* melalui *WiFi* saat *threshold* terlampaui	Program berhasil melakukan koneksi ke *server TCP* pada *CLI* melalui *WiFi* saat *threshold* terlampaui	Valid
5	Mengirim gambar	Program dapat mengirim semua gambar melalui *server TCP*	Program berhasil mengirim semua gambar melalui *server TCP*	Valid
6	*Loop*	Program dapat mengosongkan *framebuffer* untuk penagkapan gambar selanjutnya	Program berhasil mengosongkan *framebuffer* lalu diskonek *WiFi* lalu memulai membaca *sound level* lagi	Valid

TABEL 3.8
PENGUJIAN *CLI*

No	Parameter	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
1	*Parsing argument*	Program dapat mengetahui input yang user berikan ketika menjalankan program	*Argument* berhasil di *parse*	Valid
2	*Server HTTP*	Program dapat menjalankan *server HTTP*	Program berhasil menjalankan *server HTTP* jika menemukan *argument --gui* yang diberikan *user*	Valid
3	*Server WebSocket*	Program dapat menjalankan *server* WebSocket sebagai *IPC* antara *GUI*	Program berhasil menjalankan *server WebSocket* sebagai *IPC* antara *GUI* jika menemukan *argument --gui* yang diberikan *user*	Valid
4	*Server TCP*	Program dapat menjalankan *server*	Program berhasil menjalankan *server TCP* untuk *ESP32 Camera*	Valid
5	Konversi *byte TCP*	Program dapat mengetahui *index* gambar menggunakan *hex* pada *byte TCP*	Program berhasil melakukan konversi *binary ascii* ke *hex* lalu	Valid
6	Menerima gambar	Program dapat menerima gambar sesuai limit yang ditetapkan	Program berhasil menerima gambar sesuai limit yang ditetapkan	Valid
7	Validasi transmisi	Program dapat mendeteksi transmisi *TCP* yang tidak valid	Program berhasil mendeteksi transmisi yang tidak valid dan melakukan proses *fallback*	Valid
8	Analisa *AI*	Program dapat mendeteksi objek dan kondisi pagar	Program berhasil mendeteksi objek dan kondisi pagar	Valid

9	Notifikasi hasil analisa	Program dapat memberikan notifikasi	Program berhasil memberikan notifikasi pada sistem operasi menggunakan *notify-send* atau pada *GUI* menggunakan *WebSocket*	Valid

TABEL 3.9
PENGUJIAN *GUI*

No	Parameter	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Aset ditampilkan	Program dapat menampilkan semua aset visualisasi	Program berhasil menampilkan semua aset visualisasi	Valid
2	Koneksi *IPC*	Program dapat melakukan koneksi ke *server WebSocket* pada aplikasi *CLI*	Program berhasil melakukan koneksi ke *server WebSocket* pada aplikasi *CLI*	Valid
3	*Message parsing*	Program dapat memproses pesan dari *server*	Program berhasil memproses pesan dari *server*	Valid
4	*Visual change*	Program dapat menampilkan perubahan visual sesuai pesan dari *server*	Program berhasil menampilkan perubahan visual sesuai pesan yang diterima	Valid

IV. KESIMPULAN

Berdasarkan hasil dari Pembangunan Sistem Pengawasan Cerdas Dengan Visualisasi *3D*, dapat disimpulkan:

1. Data yang didapatkan melalui transmisi *TCP* yakni *binary* dapat di konversikan ke *HEX* untuk lebih mudah dimanipulasi.
2. Notifikasi pada sistem dapat dilakukan dengan memanggil instruksi yang tersedia pada sistem tersebut.
3. Analisa kecerdasan buatan membutuhkan *resource* komputasi yang besar, namun sekarang dapat dilakukan pada komputer yang kita gunakan sehari-hari.
4. Penggunaan *MAX9814* sebagai *trigger* dapat memberikan kemudahan dalam mendeteksi adanya aktivitas yang terjadi di lingkungan rumah terutama adanya tamu yang memanggil kita.
5. Penggunaan *AI* untuk mendeteksi kondisi pagar menghilangkan kerumitan (*hassle*) memasang sebuah sensor ke pagar.

REFERENSI

[1]"Pengertian CCTV: Sejarah, Fungsi, Gambar, Jenis dan Cara Kerja Kamera", Ames Boston, 2022. [Online]. Available: https://www.amesbostonhotel.com/pengertian-cctv/. [Accessed: 01- Apr- 2022].

[2]D. Workshop and D. Workshop, "ESP32-CAM - Getting Started & Solving Common Problems", DroneBot Workshop, 2022. [Online]. Available: https://dronebotworkshop.com/esp32-cam-intro/. [Accessed: 30- Mar- 2022].

[3]Datasheets.maximintegrated.com, 2022. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf. [Accessed: 01- Apr- 2022].

[4]Aprylia, Repositori.usu.ac.id, 2022. [Online]. Available: https://repositori.usu.ac.id/bitstream/handle/123456789/29501/172411074.pdf?sequence=1&isAllowed=y. [Accessed: 02- Apr- 2022].

[5]Youtube.com, 2022. [Online]. Available: https://www.youtube.com/watch?v=mu3-Sff0B9w. [Accessed: 02- Apr- 2022].

[6] A. I. Purnamasari and A. Setiawan, "Pengembangan Passive Infrared Sensor (PIR) HC-SR501 dengan Microcontrollers ESP32-CAM Berbasiskan Internet of Things (IoT) dan Smart Home sebagai Deteksi Gerak untuk Keamanan Perumahan," Prosiding SISFOTEK, vol. 3, no. 1, pp. 148–154, Oct. 2019.

[7]Uctronics.com, 2022. [Online]. Available: https://www.uctronics.com/download/OV2640_DS.pdf. [Accessed: 30- Mar- 2022].

[8]A. Industries, "Electret Microphone Amplifier - MAX9814 with Auto Gain Control." [Online]. Available: https://www.adafruit.com/product/1713. [Accessed: Jul. 05, 2022].

[9] "What is PlatformIO? — PlatformIO latest documentation." [Online]. Available: https://docs.platformio.org/en/latest/what-is-platformio.html. [Accessed: Jul. 06, 2022].

[10]"WebGL: 2D and 3D graphics for the web - Web APIs | MDN", Developer.mozilla.org, 2022. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API. [Accessed: 31- Mar- 2022].

[11]"WebGL Fundamentals", Webglfundamentals.org, 2022. [Online]. Available: https://webglfundamentals.org/webgl/lessons/webgl-fundamentals.html. [Accessed: 31- Mar- 2022].

[12]"Browser Compatibility Testing of WebGL 2.0", Lambdatest.com, 2022. [Online]. Available: https://www.lambdatest.com/webGL-2.0. [Accessed: 31- Mar- 2022].

[13]"What is PyTorch? | Applications, Advantages & Diadvantage of PyTorch," EDUCBA, Jul. 04, 2021. [Online]. Available: https://www.educba.com/what-is-pytorch/. [Accessed: Jul. 05, 2022].

[14] B. D. APR 29 and 2021 8 Min Read, "Getting Started with Roboflow," Roboflow Blog, Apr. 29, 2021. [Online]. Available: https://blog.roboflow.com/getting-started-with-roboflow/. [Accessed: Jul. 06, 2022].

[15]J. Goulding, "What is Rust and why is it so popular?," Stack Overflow Blog, Jan. 20, 2020. [Online]. Available: https://stackoverflow.blog/2020/01/20/what-is-rust-and-why-is-it-so-popular/. [Accessed: Jul. 05, 2022].

[16]"What is Ownership? - The Rust Programming Language." [Online]. Available: https://doc.rust-lang.org/book/ch04-01-what-is-ownership.html. [Accessed: Jul. 05, 2022].

[17]"What is Asynchronous and What Does it Mean?," SearchNetworking. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/asynchronous. [Accessed: Jul. 05, 2022].

[18]S. Kerkour, "Async Rust: What is a runtime? Here is how tokio works under the hood," Sylvain Kerkour, Feb. 01, 2022. [Online]. Available: https://kerkour.com/rust-async-await-what-is-a-runtime. [Accessed: Jul. 05, 2022].

[19]“Tutorial | Tokio - An asynchronous Rust runtime.” [Online]. Available: https://tokio.rs/tokio/tutorial. [Accessed: Jul. 05, 2022].

[20] “What is TCP/IP? | Cloudflare.” [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/. [Accessed: Jul. 06, 2022].

[21] “Introduction to WebSockets Programming | www.Developer.com,” Developer.com, Nov. 19, 2021. [Online]. Available: https://www.developer.com/web-services/intro-web-sockets/. [Accessed: Jul. 06, 2022].