



REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA

Barrier Gate

GROUP B-5

Eldisja Hadasa	2106640133
Andikha Wisanggeni	2106731503
Muhammad Aqil Muzakky	2106731604
Laode Alif Ma'sum Sidrajat Raja Ika	2106731213

PREFACE

Puji dan syukur kita panjatkan kehadirat Tuhan Yang Maha Esa yang telah memberikan rahmatnya sehingga kami sebagai kelompok B-5 dapat menyelesaikan tugas akhir Praktikum Sistem Waktu Nyata dan IOT yang berjudul “Barrier Gate”.

Seiring perkembangan zaman, teknologi pun juga akan semakin berkembang dan tidak bisa terlepas dari kehidupan sehari-hari. Hal ini dapat dilihat mulai dari kita bangun di pagi hari hingga nanti kita tidur di malam hari, kita tidak akan bisa lepas dari teknologi. Dengan membuat proyek ini, kami berharap kami dapat mengimplementasikan modul-modul yang telah kami pelajari pada modul-modul sebelumnya, dan juga proyek ini dapat bermanfaat bagi masyarakat sekitar.

Kami sebagai kelompok B-5 ingin mengucapkan terima kasih kepada seluruh rekan kelompok kami yang telah bekerja sama dengan baik dalam mencari ide, membuat rangkaian, menulis kode, hingga menyusun laporan. Berkat kerja sama ini, proyek kami dapat diselesaikan tepat waktu dan memberikan hasil yang memuaskan. Kami juga ingin menyampaikan rasa terima kasih kami kepada asisten laboratorium yang telah membimbing kami selama praktikum. Kami juga ingin berterima kasih kepada bang M. Raihan Azhari yang telah memberikan saran untuk proyek kami.

Depok, December 08, 2023

Group B-5

TABLE OF CONTENTS

CHAPTER 1.....	4
INTRODUCTION.....	4
1.1 PROBLEM STATEMENT.....	4
1.3 ACCEPTANCE CRITERIA.....	5
1.4 ROLES AND RESPONSIBILITIES.....	6
1.5 TIMELINE AND MILESTONES.....	7
CHAPTER 2.....	8
IMPLEMENTATION.....	8
2.1 HARDWARE DESIGN AND SCHEMATIC.....	8
2.2 SOFTWARE DEVELOPMENT.....	10
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	38
CHAPTER 3.....	40
TESTING AND EVALUATION.....	40
3.1 TESTING.....	40
3.2 RESULT.....	41
3.3 EVALUATION.....	42
CHAPTER 4.....	43
CONCLUSION.....	43

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Dalam perkembangan masyarakat urban yang pesat, kebutuhan akan sistem keamanan perumahan semakin menjadi prioritas utama. Kendaraan bermotor menjadi aspek penting dalam kehidupan sehari-hari, namun kendali terhadap akses dan keamanannya seringkali kurang optimal. Sistem tradisional yang mengandalkan kunci fisik atau remote control sering kali rentan terhadap risiko keamanan, seperti hilangnya kunci atau penggunaan remote control yang tidak sah. Oleh karena itu, tim IoT kami merasa perlu untuk menciptakan solusi inovatif yang dapat meningkatkan keamanan perumahan dengan memanfaatkan teknologi Internet of Things (IoT).

Proyek ini didasarkan pada pemahaman bahwa metode otentikasi yang canggih dan terkini diperlukan untuk memastikan akses kendaraan yang aman dan efisien. Dengan menggunakan teknologi Bluetooth dan WiFi, kami bertujuan untuk menciptakan sistem keamanan perumahan yang terhubung secara pintar. Tujuan utama kami adalah mengimplementasikan sistem yang dapat secara otomatis mengidentifikasi dan mengautentikasi kendaraan berdasarkan credential Bluetooth, sehingga meminimalkan risiko akses yang tidak sah. Keputusan ini didasarkan pada keinginan kami untuk memberikan solusi yang tidak hanya efektif dalam meningkatkan keamanan, tetapi juga praktis dalam penggunaannya.

Melalui proyek ini, kami berharap dapat memberikan kontribusi positif terhadap solusi keamanan perumahan yang modern dan berkelanjutan. Solusi ini diharapkan dapat menjadi alternatif yang lebih efisien dan andal, meninggalkan metode tradisional yang rentan terhadap risiko keamanan. Dengan memadukan keahlian dalam IoT, keamanan, dan teknologi nirkabel, tim kami memiliki keyakinan bahwa proyek ini dapat menjadi langkah inovatif dalam menciptakan lingkungan perumahan yang lebih aman, cerdas, dan terkoneksi.

1.2 PROPOSED SOLUTION

Dalam merespons tantangan keamanan perumahan yang semakin kompleks di era urbanisasi yang pesat, tim IoT kami merancang sebuah solusi yang menggabungkan

kecerdasan buatan dan Internet of Things (IoT) untuk meningkatkan keamanan akses kendaraan di lingkungan perumahan. Terinspirasi oleh keterbatasan sistem keamanan konvensional yang cenderung rentan terhadap risiko pencurian atau akses yang tidak sah, kami merasa perlunya sebuah solusi inovatif yang mampu memberikan tingkat keamanan yang lebih tinggi dan kemudahan penggunaan.

Konsep utama dari proyek ini adalah menggunakan dua modul ESP (Embedded Systems Platform) yang berkomunikasi melalui WiFi. Modul pertama, yang dilengkapi dengan kemampuan Bluetooth, bertanggung jawab untuk mendeteksi keberadaan kendaraan di sekitar dan mengidentifikasi mereka berdasarkan credential Bluetooth yang terdaftar. Melalui scanning MAC address perangkat Bluetooth yang terdekat, sistem secara otomatis memvalidasi akses dan memberikan izin hanya kepada kendaraan yang terdaftar dalam database sistem.

Modul kedua berfungsi sebagai pengendali aktuator, seperti pintu otomatis, yang dikendalikan melalui sinyal WiFi dari modul pertama. Jika kendaraan terdeteksi dan credential Bluetooth valid, modul kedua akan menerima instruksi untuk membuka pintu. Sebaliknya, jika kendaraan tidak terdaftar atau terjadi ketidaksesuaian credential, pintu akan tetap terkunci dan sistem akan memberikan peringatan.

Melalui integrasi teknologi Bluetooth dan WiFi, proyek ini menjanjikan keamanan akses yang efisien dan terkoneksi, serta memberikan kemudahan manajemen akses kendaraan bagi penghuni perumahan. Solusi ini tidak hanya mengatasi tantangan keamanan, tetapi juga membawa perubahan positif dalam pengelolaan keamanan perumahan menuju arah yang lebih cerdas dan terintegrasi. Dengan fokus pada pemanfaatan teknologi terkini, tim kami berkomitmen untuk menciptakan solusi yang dapat memenuhi tuntutan keamanan modern dan meningkatkan kualitas hidup penghuni perumahan.

1.3 ACCEPTANCE CRITERIA

The acceptance criteria of this project are as follows:

1. Ketika kendaraan hendak memasuki area, terutama pusat, prosesnya akan melibatkan koneksi dengan perangkat Bluetooth. Tujuan utama dari koneksi ini adalah untuk memverifikasi apakah kredensial Bluetooth dari mobil yang hendak masuk telah terdaftar.

2. Jika kendaraan terdaftar, sistem akan mengirimkan pesan ke perangkat lainnya untuk membuka pintu.
3. Namun, apabila kendaraan tidak terdaftar, sistem akan mengunci pintu dan memberikan peringatan sesuai.
4. Pengecekan kredensial Bluetooth dengan menggunakan alamat MAC dari hasil pemindaian perangkat terdekat, dimana nantinya alamat MAC tersebut akan dibandingkan dengan daftar alamat MAC yang telah terdaftar.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Programmer	Membuat kode untuk dijalankan di alat	Laode Alif Ma'sum Sidrajat Raja Ika, Andikha Wisanggeni, Eldisja Hadasa, Muhammad Aqil Muzakky
Hardware Designer	Membuat desain dan merangkai desain untuk hardware	Laode Alif Ma'sum Sidrajat Raja Ika, Andikha Wisanggeni, Eldisja Hadasa
Pembuat laporan	Membuat dan mengedit laporan	Andikha Wisanggeni, Eldisja Hadasa, Laode Alif Ma'sum Sidrajat Raja Ika

Table 1. Roles and Responsibilities

1.5 TIMELINE AND MILESTONES

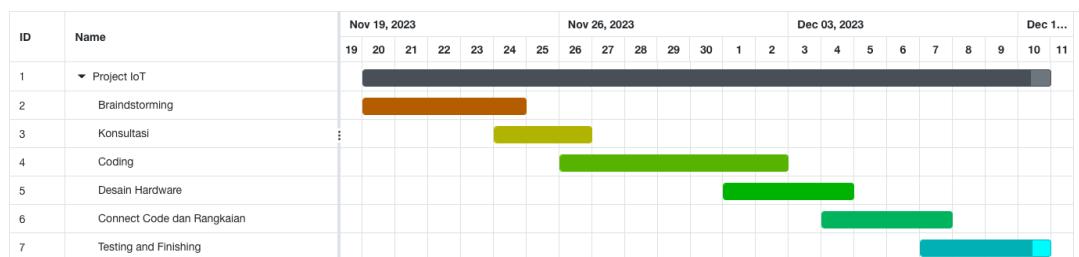


Fig 1. Gantt Chart

- a. Brainstorming : Mencari ide untuk judul, deskripsi dari proyek akhir, dan menentukan modul-modul yang akan digunakan.
- b. Konsultasi : Bertanya kepada asisten laboratorium tentang ide yang dimiliki.
- c. Coding : Menulis kode untuk diimplementasikan ke rangkaian nantinya, dan kode akan dalam bahasa C++ dan Python.
- d. Desain Hardware : Menentukan bahan-bahan yang diperlukan dan merangkainya sesuai dengan ide yang telah ada.
- e. Connect Code dan Rangkaian : Menyambungkan kode yang telah dibuat dengan rangkaian yang telah dirangkai.
- f. Testing dan Finishing : Melakukan testing dan menyempurnakan alat yang dibuat, dan juga menyusun laporan.

CHAPTER 2

IMPLEMENTATION

2.1 HARDWARE DESIGN AND SCHEMATIC



Fig 2. Rangkaian terbuka

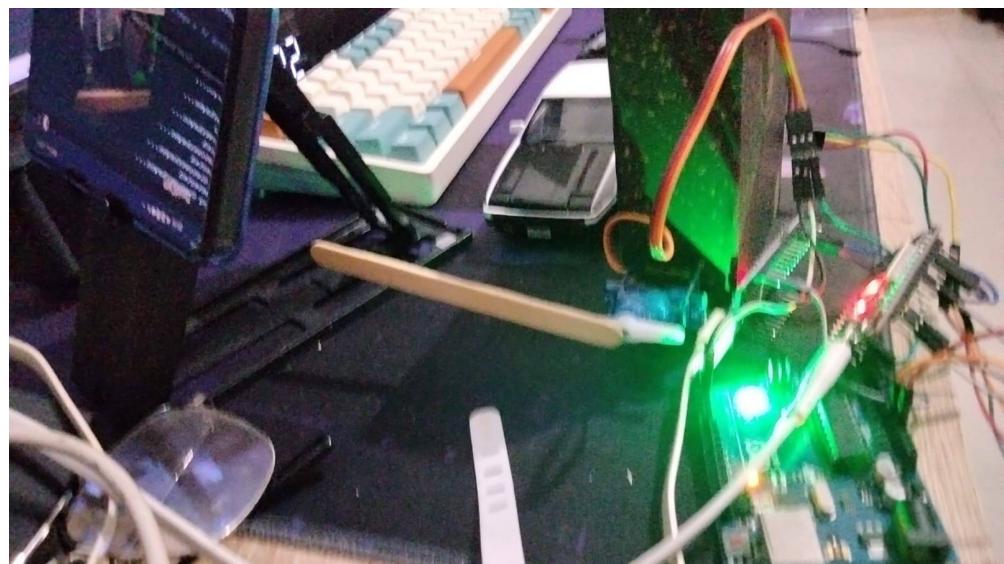


Fig 3. Rangkaian Terbuka sisi lain

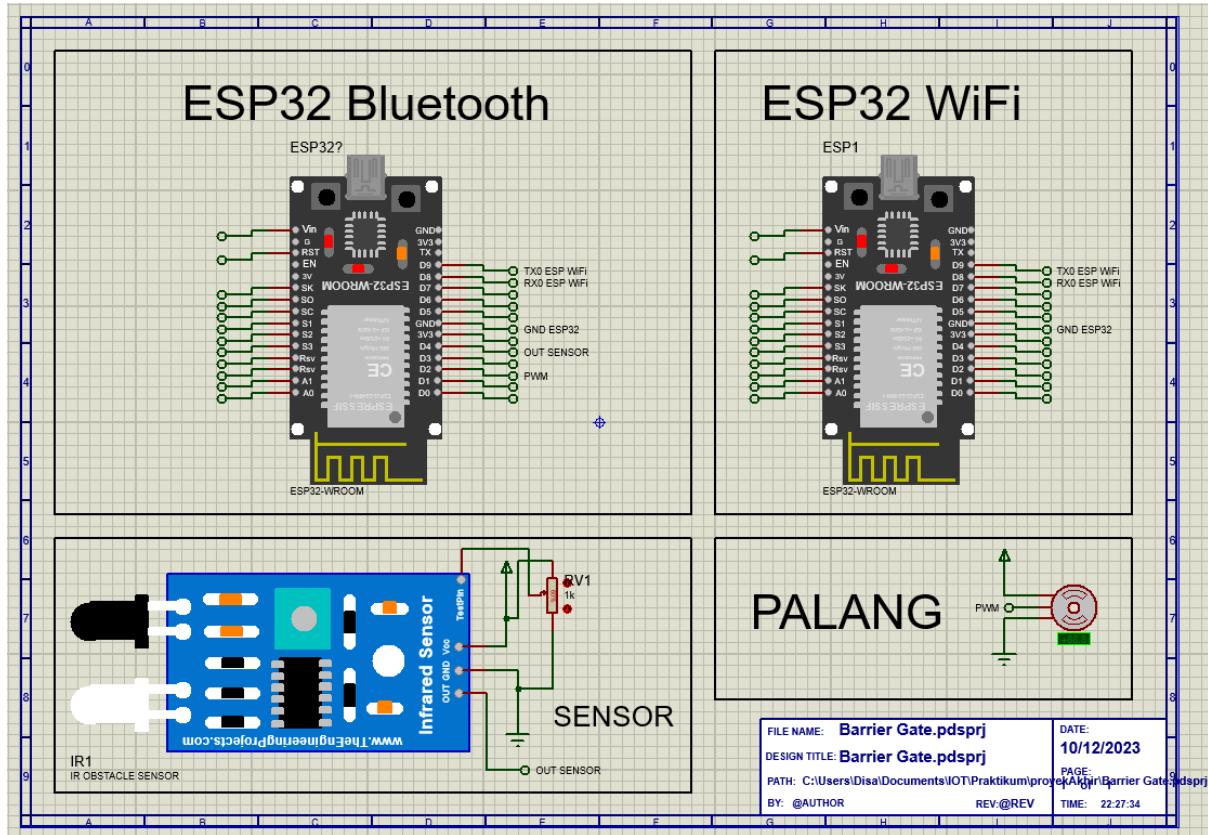


Fig 4. Rangkaian Proteus

Terdapat beberapa komponen pada proyek ini, seperti ESP32, servomotor, infrared sensor, Bluetooth module, Arduino, dan kabel jumper [3][4]. Berikut penjelasan untuk setiap komponen:

a. ESP32

Merupakan mikrokontroler yang kuat dan serbaguna dengan kemampuan WiFi dan Bluetooth. Di dalam proyek ini, ESP32 berfungsi sebagai otak dari sistem, mengelola komunikasi WiFi dengan perangkat pengontrol, serta mengendalikan servomotor dan sensor infrared.

b. Servomotor

Servomotor digunakan untuk menggerakkan atau mengendalikan palang sesuai dengan instruksi yang diterima dari ESP32. Dalam proyek ini, servomotor digunakan untuk mengontrol palang berdasarkan informasi yang diterima dari sensor infrared atau perintah dari perangkat pengontrol melalui WiFi.

c. Infrared sensor

Infrared sensor digunakan untuk mendeteksi objek atau kejadian tertentu. Di proyek ini, sensor ini digunakan untuk mendeteksi kendaraan atau objek yang mendekati palang. Informasi dari sensor ini kemudian dapat digunakan oleh ESP32 untuk mengambil keputusan terkait status dari palang, apakah harus dibuka atau ditutup.

d. Bluetooth module

Bluetooth module dapat berfungsi sebagai antarmuka alternatif untuk mengontrol sistem, meskipun pada kode tersebut hanya terlihat penggunaan WiFi untuk komunikasi. Bluetooth ini akan digunakan untuk memindai dan membandingkan MAC address.

e. Arduino

Arduino disini digunakan untuk mengantikan power supply pada rangkaian.

f. Kabel jumper

Kabel jumper digunakan untuk menghubungkan pin-pin antar komponen. Dalam proyek ini, kabel jumper akan menghubungkan pin ESP32 ke servomotor, sensor infrared, Bluetooth module, dan Arduino.

2.2 SOFTWARE DEVELOPMENT

Source Code:

a. main.py

```
# main.py
import time

from WiFiCommunicator import WiFiCommunicator

# Buat objek WiFiCommunicator
wifi_communicator = WiFiCommunicator(max_buffer_sz=1024)

# Jalankan program
try:
    while True:
        # Dapatkan pesan dari antrian pesan masuk
```

```

message = wifi_communicator.get_message()
print(message.data)
wifi_communicator.send_message("test")
if message is not None:
    # Periksa apakah plat nomor sesuai dengan yang
    diinginkan
    plat_nomor_terdeteksi = message.data
    plat_nomor_diinginkan = "ABC123"

    is_plat_nomor_sesuai = plat_nomor_terdeteksi ==
    plat_nomor_diinginkan

    # Cetak apakah palang bedas naik atau turun
    berdasarkan hasil komparasi
    #if is_plat_nomor_sesuai:
        # print("Palang Bedas Naik")
    #else:
        # print("Palang Bedas Turun")
    time.sleep(0.5)
except KeyboardInterrupt:
    # Tangkap sinyal keyboard interrupt dan hentikan program
    wifi_communicator.destroy()

```

Penjelasan:

Program 'main.py' yang diberikan berfungsi sebagai penghubung untuk komunikasi melalui WiFi menggunakan objek 'WiFiCommunicator'. Pertama-tama, modul 'time' dan 'WiFiCommunicator' diimpor untuk mengatur waktu dan mengakses fungsionalitas komunikasi WiFi.

Selanjutnya, objek 'wifi_communicator' dibuat dari kelas 'WiFiCommunicator' dengan ukuran maksimum buffer sebesar 1024. Ini akan digunakan sebagai sarana untuk mengelola proses komunikasi WiFi. Program memasuki loop utama yang berjalan tak terbatas ('while True:'). Pada setiap iterasi, program mencoba untuk mengambil pesan dari antrian pesan masuk melalui metode 'get_message()' pada objek 'wifi_communicator'. Pesan yang diterima kemudian dicetak ke konsol, dan program mengirim pesan dengan data "test" menggunakan metode 'send_message("test")'.

Selanjutnya, program melakukan pemeriksaan terhadap pesan yang diterima. Jika pesan tidak kosong, program menyimpan data plat nomor terdeteksi dan membandingkannya dengan plat nomor yang diinginkan ("ABC123"). Hasil perbandingan disimpan dalam variabel 'is_plat_nomor_sesuai', yang kemudian digunakan untuk mencetak status apakah plat nomor sesuai atau tidak. Program juga menyisipkan jeda selama 0.5 detik dengan menggunakan 'time.sleep(0.5)' sebelum memulai iterasi berikutnya. Dengan demikian, program tetap berjalan dan memproses pesan secara berkala.

Terakhir, program menangani pengecualian 'KeyboardInterrupt', yang menangkap sinyal saat pengguna menekan Ctrl+C. Dalam kasus ini, program memanggil metode 'destroy()' pada objek 'wifi_communicator' untuk membersihkan atau menutup koneksi WiFi sebelum keluar dari program. Ini memastikan penghentian program dengan aman dan pembebasan sumber daya yang diperlukan.

b. WiFiCommunicator.py

```
import time
import socket
import threading
from queue import Queue

class InMessage:
    """
        Incoming Message definition
        NOTE: This is created so that you can add as many flags as
        you want,
        without changing the interface, and you'd only need to
        change the decoding method
    """

    def __init__(self, data: str, require_ack: bool,
                 client_addr: str) -> None:
        self.data = data
        self.require_acknowledgment = require_ack
        self.client_addr = client_addr
```

```

class OutMessage:
    """
        Outgoing Message definition
    NOTE: This is created so that you can add as many flags as
you want,
        without changing the interface, and you'd only need to
change the encoding method
    """

    def __init__(self, data: str, require_ack: bool = False) ->
None:
        self.data = data


class WiFiCommunicator:
    """
    """

    ACKNOWLEDGMENT_FLAG = 'A'
    CPU_RELEASE_SLEEP = 0.000_001

    def __init__(self, max_buffer_sz: int, port: int = 11111,
in_queue_sz: int = 0, out_queue_sz: int = 0) -> None:
        """
            @param max_buffer_sz: The maximum amount of bytes to be
received at once
            @param port: The port on which we shall communicate
            @param in_queue_sz: The incoming messages' queue size,
if 0 -> infinite
            @param out_queue_sz: The outgoing messages' queue size,
if 0 -> infinite
        """

        assert max_buffer_sz > 0, f"Buffer size must be > 0
[ {max_buffer_sz = } ]"
        assert in_queue_sz >= 0, f"Queue size can't be negative
[ {in_queue_sz = } ]"
        assert out_queue_sz >= 0, f"Queue size can't be negative
[ {out_queue_sz = } ]"

        # Signal to the communicator to "Rest In Peace"
        self._rip = False
        self._have_client = False

```

```

        self._max_buffer_size = max_buffer_sz
        self._incoming_messages_queue =
Queue(maxsize=in_queue_sz)
        self._outgoing_messages_queue =
Queue(maxsize=out_queue_sz)

        # Client info
        self._client = None
        self._client_address = None

        # Socket creation
        soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        soc.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)

        # To allow the ESP through the firewall (if u had
firewall problems, and u're on Linux):
        # $ sudo iptables -I INPUT -s ESP.IP.GOES.HERE -p tcp
--dport 15000 -j ACCEPT
        soc.bind(('0.0.0.0', port))
        soc.listen(0)

        # Start the show
        self._threads = [
            threading.Thread(target=self.__listener_thread,
daemon=True),
            threading.Thread(target=self.__sender_thread,
daemon=True),

threading.Thread(target=self.__wait_for_connection_thread,
daemon=True, args=[soc]),
        ]
        for thread in self._threads:
            thread.start()

    def get_message(self) -> InMessage:
        """
        Returns (if exists) a message from the incoming messages
queue
        """

```

```
        return self._incoming_messages_queue.get()

    def send_message(self, message: OutMessage) -> None:
        """
        Adds a message to the sending queue to be sent
        """
        self._outgoing_messages_queue.put(message)

    def destroy(self):
        """
        Destroy the communicator
        """
        if self._client is not None:
            self._client.close()

        self._rip = True
        for thread in self._threads:
            thread.join(0.1)

    def __wait_for_connection_thread(self, soc: socket.socket)
-> None:
        """
        Establish a connection with a client, and die
        """
        self._client, self._client_address = soc.accept()
        self._have_client = True

    def __decode(self, in_bytes: bytes) -> 'None|InMessage':
        """
        Decodes the incoming message to the required format
        """
        message = in_bytes.decode()
        if not len(message):
            return None

        ack = message[0] == self.ACKNOWLEDGMENT_FLAG
        data = message[1 * ack:]
        return InMessage(data=data, require_ack=ack,
client_addr=self._client_address)

    def __listener_thread(self):
        while not self._rip:
```

```

        if not self._have_client:
            time.sleep(self.CPU_RELEASE_SLEEP)
            continue

        message = self._client.recv(self._max_buffer_size)
        decoded_msg = self.__decode(message)

        if decoded_msg is not None:
            # Tambahkan langkah-langkah komparasi plat nomor
            # di sini
            plat_nomor_terdeteksi = decoded_msg.data
            plat_nomor_diinginkan = "ABC123" # Hardcode
            nilai plat nomor yang diinginkan

            is_plat_nomor_sesuai = plat_nomor_terdeteksi ==
            plat_nomor_diinginkan

            # Kirim sinyal berdasarkan hasil komparasi
            if is_plat_nomor_sesuai:
                # Kirim sinyal untuk palang bedas menyalal
                sinyal = OutMessage(data="Palang Bedas
                Menyalal")
                self._outgoing_messages_queue.put(sinyal)
            else:
                # Kirim sinyal untuk palang bedas mati
                sinyal = OutMessage(data="Palang Bedas
                Mati")
                self._outgoing_messages_queue.put(sinyal)

            # Tambahkan pesan yang telah didekod ke dalam
            antrian pesan masuk
            self._incoming_messages_queue.put(decoded_msg)

    def __encode(self, message: OutMessage) -> bytes:
        '''
        Encodes the outgoing message into the required sendable
        format
        '''
        return message.data.encode()

    def __sender_thread(self):
        '''

```

```
'''  
        while not self._rip:  
            if not self._have_client:  
                time.sleep(self.CPU_RELEASE_SLEEP)  
                continue  
  
                # This is blocking on purpose, if not, we'd have to  
                handle getting no-data when timing-out  
                msg = self._outgoing_messages_queue.get()  
                self._client.send(self.__encode(msg))
```

Penjelasan:

Kelas 'WiFiCommunicator' dalam file 'WiFiCommunicator.py' bertanggung jawab atas implementasi komunikasi melalui WiFi dengan menggunakan antarmuka yang terstruktur. Dalam konstruktor, kelas menerima parameter seperti 'max_buffer_sz', 'port', 'in_queue_sz', dan 'out_queue_sz', yang mengatur ukuran buffer maksimum, nomor port, serta ukuran antrian pesan masuk dan keluar. Objek 'Queue' digunakan untuk mengelola antrian pesan masuk dan keluar.

Tiga thread dimulai pada saat inisialisasi, yaitu thread pemantau ('__listener_thread'), thread pengirim ('__sender_thread'), dan thread menunggu koneksi ('__wait_for_connection_thread'). Thread menunggu koneksi bertugas untuk menerima koneksi dari klien dan menandai bahwa koneksi telah terbentuk dengan mengubah status '_have_client' menjadi True. Thread pemantau adalah bagian inti dari kelas ini. Thread ini terus mendengarkan pesan dari klien, menggunakan socket untuk menerima pesan dan kemudian mendekode pesan menggunakan metode '__decode'. Hasil dekoding, yang berupa objek 'InMessage', dimasukkan ke dalam antrian pesan masuk. Thread pengirim mengambil pesan dari antrian pesan keluar dan mengirimnya ke klien setelah melakukan encoding menggunakan metode '__encode'.

Kelas ini memberikan antarmuka yang bersih dan mudah digunakan untuk komunikasi melalui WiFi. Selain itu, struktur kodennya memungkinkan penambahan fitur tanpa memerlukan perubahan pada antarmuka komunikasi yang sudah ada. Seluruh sistem ini memastikan pengelolaan pesan yang efisien dan handal selama proses komunikasi berlangsung.

c. number_plate.py

```
import cv2
import time
import matplotlib.pyplot as plt
import easyocr
from IPython.display import Image
from wifi_communicator import WiFiCommunicator
import re

harcascade = "model/haarcascade_russian_plate_number.xml"
plate = None
plateMatch = False
registeredPlate = "13954"

def keep_alphanumeric(input_string):
    if input_string is not None:
        # Use a regular expression to keep only alphanumeric
        characters
        cleaned_string = re.sub(r'[^\w\d]', ' ', input_string)
        return cleaned_string
    else:
        return None

# Jalankan program
def main():
    global plate
    wifi_communicator = WiFiCommunicator(max_buffer_sz=1024,
    plate_match_ref=plateMatch)
    while True:
        try:
            correctCount = 0
            # Dapatkan pesan dari antrian pesan masuk
            message = wifi_communicator.get_message().data
            print(message)
            if (message == "Denied!"):
                print("Running plate recognition")
                if(wifi_communicator._plate_match_ref != True):
                    plate = plateDetection()
                    plate = keep_alphanumeric(plate)
                    print("Parse : " + plate)
```

```

        if message is not None:
            # Periksa apakah plat nomor sesuai dengan
            yang diinginkan
            for i in range(len(registeredPlate)):
                try:
                    if plate[i] == registeredPlate[i]:
                        correctCount += 1
                except IndexError:
                    # Handle the IndexError (e.g., print
                    a message or take appropriate action)
                    print("Index out of bound. Skipping
comparison for index:", i)
                    break # Break out of the for loop
and reloop the entire main function
                # Calculate correct percentage only if there
                are elements in both plates
                if len(plate) > 0 and len(registeredPlate) >
0:
                    correctPercentage = correctCount /
len(plate)
                    print("Correct percent:",
correctPercentage)
                else:
                    print("No elements to compare.")
                    if correctCount / len(plate) >= 0.6:
                        print("Correct percent: " +
str(correctCount / len(plate)))
                        wifi_communicator._plate_match_ref =
True
                        print("Plate Match: " +
str(wifi_communicator._plate_match_ref))
                except KeyboardInterrupt:
                    # Tangkap sinyal keyboard interrupt dan hentikan
                    program
                    wifi_communicator.destroy()
                    time.sleep(1)

def plateDetection():
    cap = cv2.VideoCapture(0)

    cap.set(3, 640) # width
    cap.set(4, 480) # height

```

```
min_area = 500
plateFound = False
img_roi = None

while not plateFound:
    success, img = cap.read()

    plate_cascade = cv2.CascadeClassifier(harcascade)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    plates = plate_cascade.detectMultiScale(img_gray, 1.1,
4)

    for (x, y, w, h) in plates:
        area = w * h

        if area > min_area:
            cv2.rectangle(img, (x, y), (x + w, y + h), (0,
255, 0), 2)
            cv2.putText(img, "Number Plate", (x, y - 5),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255, 0, 255), 2)

            img_roi = img[y: y + h, x:x + w]
            cv2.imshow("ROI", img_roi)

            timestamp = time.time()
            print(timestamp)
            time.sleep(0.2)
            cv2.imwrite("plates/" + str(timestamp) + ".jpg",
img_roi)
            cv2.rectangle(img, (0, 200), (640, 300), (0,
255, 0), cv2.FILLED)
            cv2.putText(img, "Plate Saved", (150, 265),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 2, (0, 0, 255), 2)
            cv2.imshow("Results", img)
            cv2.waitKey(500)
            Image('plates/' + str(timestamp) + '.jpg')
            time.sleep(0.5)
            reader = easyocr.Reader(['en'])
            output = reader.readtext('plates/' +
str(timestamp) + '.jpg')
```

```

        if output:
            plate = output[0][1]
            print("Plate found:", plate)
            plateFound = True
            break # Break the inner loop

cv2.imshow("Result", img)

key = cv2.waitKey(1) & 0xFF # Check for key press (wait
for 1 ms)

if key == ord('q'): # Quit if 'q' is pressed
    break

cap.release() # Release the video capture
cv2.destroyAllWindows() # Close all OpenCV windows
return plate

if __name__ == '__main__':
    main()

```

Penjelasan:

'number_plate.py' dirancang untuk mengintegrasikan fungsi pengenalan plat nomor dalam sistem kontrol gerbang. Saat dieksekusi, code ini memanfaatkan OpenCV untuk mendeteksi nomor plat dalam video dari kamera. Setelah deteksi, gambar plat disimpan dan dianalisis menggunakan EasyOCR untuk mengambil karakter alfanumerik.

Melalui modul kustom 'WiFiCommunicator', code berkomunikasi dengan sistem utama melalui jaringan Wi-Fi. Pesan yang diterima, khususnya "Denied!", memicu proses pengenalan nomor plat. Jika nomor plat terdeteksi dan sesuai dengan yang terdaftar, sistem mengenali kredensial dan memperbarui status plat yang sesuai pada 'WiFiCommunicator'. Code ini memberikan kontribusi penting terhadap manajemen akses area terbatas dengan memanfaatkan teknologi pengenalan plat nomor secara efisien.

d. config.h

```
/*
 *
-----*
-----*
 * Example: Two way communication between ESP32 and Python using
WIFI
 *
-----*
-----*
 * Author: Radhi SGHAIER: https://github.com/Rad-hi
 *
-----*
-----*
 * Date: 07-05-2023 (7th of May, 2023)
 *
-----*
-----*
 * License: Do whatever you want with the code ...
 *
 * If this was ever useful to you, and we happened to
meet on
 *
 * the street, I'll appreciate a cup of dark coffee, no
sugar please.
 *
-----*
-----*
 */
#ifndef __CONFIG_H__
#define __CONFIG_H__


/* Pins definitions */

#define IRSensor 18 //IR Sensor pin
#define ServoOut 14
#define ServoVcc 12
#define ServoGnd 13
```

```

/* Communication params */
#define BT_DISCOVER_TIME 10000
#define ACK "A" // acknowledgment packet
#define QUEUE_LEN 5
#define MAX_BUFFER_LEN 32

/* WiFi params */
#define WIFI_SSID "Cappuccino"
#define WIFI_PASSWORD "milkshake5623"

/* Socket */
#define SERVER_ADDRESS "192.168.18.47"
#define SERVER_PORT 11111

#endif // __CONFIG_H__

```

Penjelasan:

'config.h' adalah file konfigurasi pada proyek ESP32 yang mengatur parameter untuk perangkat keras dan komunikasi. Pada bagian pin definisi, terdapat penentuan pin untuk sensor inframerah dan servo motor, memberikan fleksibilitas dalam konfigurasi perangkat keras. Parameter komunikasi termasuk waktu pencarian perangkat Bluetooth ('BT_DISCOVER_TIME'), pesan pengakuan ('ACK'), dan panjang serta maksimum buffer antrian pesan ('QUEUE_LEN' dan 'MAX_BUFFER_LEN'). Informasi kredensial Wi-Fi, seperti nama jaringan dan kata sandi, ditentukan dalam bagian Wi-Fi params, sedangkan alamat dan port server WiFi diatur dalam bagian socket params. Dengan menyediakan nilai-nilai ini dalam file terpisah, 'config.h' memfasilitasi penyesuaian parameter tanpa perlu mengubah kode inti proyek, meningkatkan fleksibilitas dan kemudahan konfigurasi.

e. my_wifi.h

```

/*
 *
-----
```

```
-----  
* Example: Two way communication between ESP32 and Python using  
WIFI  
*  
-----  
* Author: Radhi SGHAIER: https://github.com/Rad-hi  
*  
-----  
* Date: 07-05-2023 (7th of May, 2023)  
*  
-----  
* License: Do whatever you want with the code ...  
* If this was ever useful to you, and we happened to  
meet on  
* the street, I'll appreciate a cup of dark coffee, no  
sugar please.  
*  
-----  
*/  
  
#ifndef __MY_WIFI__  
#define __MY_WIFI__  
  
#include <WiFi.h>  
#include "config.h"  
  
bool is_wifi_connected(){  
    return (WiFi.status() == WL_CONNECTED);  
}  
  
/*  
 * Attempt connection once  
 */  
static bool connect_wifi_once(){  
    WiFi.mode(WIFI_STA);  
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);  
    return (WiFi.status() == WL_CONNECTED);  
}
```

```

/*
    Task to keep on trying to connect to the WiFi forever, then
die once connected
*/
static void connect_to_wifi(void*) {
    if(!connect_wifi_once())
        while(!is_wifi_connected()) { delay(500); }
    // No longer need this task
    vTaskDelete(NULL);
}

/*
    Non-blocking way of connecting to the WiFi network
*/
void setup_wifi() {
    xTaskCreatePinnedToCore(connect_to_wifi, "WiFiTask", 2048,
NULL, 3, NULL, 0);
    delay(10);
}

#endif // __MY_WIFI__

```

Penjelasan:

'my_wifi.h' adalah bagian dari proyek ESP32 yang bertujuan untuk menyederhanakan dan mengelola koneksi WiFi. Fungsi 'is_wifi_connected()' memberikan informasi tentang status koneksi WiFi saat ini, sementara 'connect_wifi_once()' mencoba menghubungkan perangkat ke jaringan WiFi yang ditentukan dalam file 'config.h'. Ada juga fungsi 'connect_to_wifi()' yang berperan dalam membuat tugas yang berusaha terhubung ke WiFi secara berulang hingga berhasil. Proses koneksi WiFi dilakukan secara non-blok, dan tugasnya akan dihapus setelah koneksi tercapai.

File ini juga menyediakan fungsi 'setup_wifi()' yang memulai tugas koneksi WiFi pada inti 0 (core 0) dari ESP32. Dengan demikian, 'my_wifi.h' memberikan solusi yang efisien untuk manajemen koneksi WiFi pada perangkat ESP32, memastikan keterhubungan yang andal dengan menyederhanakan proses koneksi dan memberikan otomatisasi dalam percobaan ulang koneksi.

f. wifi_communicator.h

```
/*
 *
-----
-----
 * Example: Two way communication between ESP32 and Python using
WIFI
 *
-----
-----
 * Author: Radhi SGHAIER: https://github.com/Rad-hi
 *
-----
-----
 * Date: 07-05-2023 (7th of May, 2023)
 *
-----
-----
 * License: Do whatever you want with the code ...
 *           If this was ever useful to you, and we happened to
meet on
 *           the street, I'll appreciate a cup of dark coffee, no
sugar please.
 *
-----
*/
#ifndef __WIFI_COMMUNICATOR_H__
#define __WIFI_COMMUNICATOR_H__

#include "my_wifi.h"
#include "config.h"

static SemaphoreHandle_t _send_tsk_mutex;
static SemaphoreHandle_t _recv_tsk_mutex;

static QueueHandle_t _send_q;
static QueueHandle_t _recv_q;
```

```

static TaskHandle_t _socket_reporter_task_h = NULL;

// The sockets client
static WiFiClient _client;

/*
    Attempt to connect the client
*/
void connect_client(){
    // We have to connect, no other options
    while(!_client.connect(SERVER_ADDRESS, SERVER_PORT)){
        delay(1000);
    }

    bool is_client_connected(){
        return _client.connected();
    }
}

void send_message(char *msg){
    xQueueSend(_send_q, (void*)msg, 5);
}

bool get_message(char *msg){
    return xQueueReceive(_recv_q, (void*)msg, 5) == pdTRUE;
}

/*
    This task would wait for the send signal and send whatever in
the sending queue
*/
static void sender_task(void*){
    // Unless connected to the client, no need for this to run
    xSemaphoreTake(_send_tsk_mutex, portMAX_DELAY);

    char buff[MAX_BUFFER_LEN] = {0};
    while(1){
        // Wait for a notification to do anything
        if(xQueueReceive(_send_q, (void*)&buff, 5) == pdTRUE){
            _client.print(buff);
        }
    }
}

```

```

}

/*
    This task would wait for data to be present, and signal that
something has been received
*/
static void receiver_task(void*) {
    // Unless connected to the client, no need for this to run
    xSemaphoreTake(_recv_tsk_mutex, portMAX_DELAY);

    char buf[MAX_BUFFER_LEN] = { 0 };
    while(1) {
        // wait until data is available
        while(_client.available() <= 0){ delay(100); };

        // if here, then we have data to read
        for(uint8_t i = 0; i < _client.available(); i++){ buf[i] =
(char)_client.read(); };

        // Add message to the received messages queue
        xQueueSend(_recv_q, (void*)&buf, 5);
    }
}

/*
    This function would initialize the communicator and setup
everything
*/
void setup_wifi_communicator() {
    _send_q = xQueueCreate(QUEUE_LEN, MAX_BUFFER_LEN);
    _recv_q = xQueueCreate(QUEUE_LEN, MAX_BUFFER_LEN);
    _recv_tsk_mutex = xSemaphoreCreateMutex();
    _send_tsk_mutex = xSemaphoreCreateMutex();
    Serial.println("Masuk sini");
    // block both tasks once created for them to wait for the
client to connect
    xSemaphoreTake(_send_tsk_mutex, portMAX_DELAY);
    xSemaphoreTake(_recv_tsk_mutex, portMAX_DELAY);

    // Create tasks
    xTaskCreatePinnedToCore(sender_task, "sendTask", 2048, NULL,
3, NULL, 1);
}

```

```

xTaskCreatePinnedToCore(receiver_task, "receiveTask", 2048,
NULL, 3, NULL, 1);

connect_client();

// release tasks
xSemaphoreGive(_send_tsk_mutex);
xSemaphoreGive(_recv_tsk_mutex);
}

#endif // __WIFI_COMMUNICATOR_H__

```

Penjelasan:

'wifi_communicator.h' adalah bagian integral dari proyek ESP32 yang memungkinkan komunikasi dua arah antara ESP32 dan perangkat Python melalui jaringan WiFi. Dengan memanfaatkan konsep FreeRTOS, file ini menciptakan dua tugas terpisah, yaitu 'sender_task' dan 'receiver_task', yang bertanggung jawab atas pengiriman dan penerimaan pesan.

Tugas 'sender_task' didesain untuk mengirim pesan dari antrian pengiriman ('_send_q') ke perangkat Python melalui koneksi WiFi. Di sisi lain, tugas 'receiver_task' menangkap data dari saluran WiFi dan memasukkannya ke dalam antrian penerimaan ('_recv_q'). Melalui fungsi 'setup_wifi_communicator()', antrian dan mutex diinisialisasi, dan tugas-tugas ini diluncurkan, kemudian mencoba terhubung ke server dengan menggunakan 'connect_client()'.

Dengan struktur ini, 'wifi_communicator.h' menyediakan kerangka kerja yang efektif dan terorganisir untuk mencapai komunikasi yang andal antara ESP32 dan perangkat Python melalui jaringan WiFi.

g. wifiSlave.ino

```

/*
 *
-----
```

```
-----  
 * Example: Two way communication between ESP32 and Python using  
 WIFI  
 *  
-----  
 * Author: Radhi SGHAIER: https://github.com/Rad-hi  
 *  
-----  
 * Date: 07-05-2023 (7th of May, 2023)  
 *  
-----  
 * License: Do whatever you want with the code ...  
 * If this was ever useful to you, and we happened to  
 meet on  
 * the street, I'll appreciate a cup of dark coffee, no  
 sugar please.  
 *  
-----  
 */  
  
#include "config.h"  
#include "my_wifi.h"  
#include "wifi_communicator.h"  
  
#include <MyButton.h>  
  
#define ENABLE_DEBUG /* <-- Commenting this line will remove any  
 trace of debug printing */  
#include <MacroDebugger.h>  
  
// Task Handle  
TaskHandle_t UARThandle = NULL;  
TaskHandle_t wifiHandle = NULL;  
  
// Communication messages  
char incoming_msg[MAX_BUFFER_LEN] = {0};  
char response[MAX_BUFFER_LEN] = {0};  
String input;
```

```

bool accessDeniedSent = false; // Flag to track whether the
"Access Denied" response has been sent

/* A collection of random responses to send when the button is
clicked */
#define NUM_RANDOM_RESPONSES      1
char *responses[NUM_RANDOM_RESPONSES] = {
    "Denied!"
};

void receiveUART(void *pvParameters) {
    while(1) {
        input = Serial.readStringUntil('\n');
        Serial.println(input);
        if(isWordPresent(input, "Access Denied")) {
            if(wifiHandle == NULL) {
                xTaskCreatePinnedToCore(WifiCommTask, "WifiComm Task",
10240, NULL, 5, &wifiHandle, 1);
            } else{
                vTaskResume(wifiHandle);
            }
        }
    }
}

void WifiCommTask(void *pvParameters) {
    vTaskSuspend(UARTHandle);
    while(1) {
        if (eTaskGetState(UARTHandle) == eRunning) {
            vTaskSuspend(wifiHandle);
        }
        // if we lost connection, we attempt to reconnect (blocking)
        if(!is_client_connected()){
            connect_client();
        }
        bool received = get_message(incoming_msg);
        if(isWordPresent(input, "Access Denied")){
            // Choose a random response to send back
            strncpy(response, responses[0], MAX_BUFFER_LEN);
            send_message(response);
            memset(response, 0, MAX_BUFFER_LEN);
            DEBUG_I("Sent: %s", responses[0]);
        }
    }
}

```

```

        accessDeniedSent = true; // Set the flag to true once
the response is sent
        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
    if(received) {
        DEBUG_I("Received: %s", incoming_msg);
        uint8_t start = 0;

        if(incoming_msg[0] == 'A') {
            sprintf(response, "%s", ACK);
            start++;
        }

        //switch the number and do the appropriate action
        switch(incoming_msg[start]){
            case 'y':
                vTaskResume(UARTHandle);
                Serial.println("Permitted");
                accessDeniedSent = false; // Set the flag to true
once the response is sent
                vTaskDelay(1000/ portTICK_PERIOD_MS);
                // Check if UARThandle is resumed, and suspend
wifiHandle accordingly
                break;

            default:
            case 'n':
                Serial.println("Denied");
                break;
        }

        // If start is bigger than 0, then we have to acknowledge
the reception
        if(start > 0){
            send_message(response);
            // Clear the response buffer
            memset(response, 0, MAX_BUFFER_LEN);
        }
    }
}
vTaskDelete(NULL);
}

```

```
// Function to check if a specific word is present anywhere in a
string
bool isWordPresent(String input, String targetWord) {
    int index = input.indexOf(targetWord);
    return (index != -1);
}

void setup() {
    DEBUG_BEGIN();

    setup_wifi();
    Serial.println("Waiting for server online");
    setup_wifi_communicator();
    xTaskCreatePinnedToCore(receiveUART, "UART Receive", 4096,
NULL, 4, &UARTHandle, 1);

    DEBUG_I("Done setting up!");
}

void loop() {
```

Penjelasan:

'wifiSlave.ino' adalah bagian dari proyek ESP32 yang bertujuan untuk menciptakan mekanisme kontrol pintu yang dapat berkomunikasi dua arah dengan perangkat Python melalui jaringan WiFi. Kode ini menggunakan FreeRTOS untuk membuat dua tugas utama: 'UARTHandle' dan 'wifiHandle'. Tugas 'UARTHandle' menerima pesan dari Python melalui Serial UART dan mengirim respons "Access Denied" jika akses ditolak. Sementara itu, 'wifiHandle' mengelola koneksi WiFi dan merespons pesan dengan mengirim respons sesuai kondisi, termasuk respons "Denied!" jika terdeteksi bahwa akses ditolak.

Selain itu, terdapat implementasi penanganan respons yang berbeda ketika pesan mengandung kata "Access Denied." Jika kata tersebut terdeteksi, kode akan mengirimkan respons "Denied!" dan mengatur flag 'accessDeniedSent' untuk menandakan bahwa respons telah dikirim. Dengan demikian, 'wifiSlave.ino' menyajikan contoh implementasi sederhana dari komunikasi dua arah yang terintegrasi antara ESP32 dan Python melalui jaringan WiFi untuk mengendalikan pintu berdasarkan pesan yang diterima dan respons yang dihasilkan.

h. bluetoothMaster.ino

```
#include <ESP32Servo.h>
#include <BluetoothSerial.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/timers.h>

#if !defined(CONFIG_BT_ENABLED) ||
!defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` and enable it
#endif

#if !defined(CONFIG_BT_SPP_ENABLED)
#error Serial Bluetooth not available or not enabled. It is only available for the ESP32 chip.
#endif

BluetoothSerial SerialBT;

#define BT_DISCOVER_TIME 10000
#define IRSensor 18 //IR Sensor pin
#define ServoOut 14
#define ServoVcc 12
#define ServoGnd 13
static bool btScanAsync = true;
static bool btScanSync = true;
String nearestAddress;
TimerHandle_t bluetoothOneShotTimer;
```

```

String registeredDevices = "ba:98:76:f4:08:7d";
Servo Servo1;

// Function to check if a specific word is present anywhere in a
string
bool isWordPresent(String input, String targetWord) {
    int index = input.indexOf(targetWord);
    return (index != -1);
}

void openGate() {
    Serial.println("Access Permitted");
    Serial.println("Opening the gate!");
    Servo1.write(180);
    if(digitalRead(IRSensor)) {
        vTaskDelay(pdMS_TO_TICKS(700)); // 100ms delay
        Servo1.write(90);
    }
}

void btAdvertisedDeviceFound(BTAdvertisedDevice *pDevice) {
    Serial.printf("Found a device asynchronously: %s\n",
pDevice->toString().c_str());
    String receivedMessage = Serial.readStringUntil('\n');
    if (pDevice->getRSSI() > -50) {
        nearestAddress = pDevice->getAddress().toString();
        Serial.println("Nearest Device MAC Address : " +
nearestAddress);

        if (nearestAddress.equals(registeredDevices)) {
            openGate();
            delay(100);
        } else if (isWordPresent(receivedMessage, "Permitted"))
{
            openGate();
        } else {
            Serial.println("Access Denied");
            vTaskDelay(pdMS_TO_TICKS(500));
        }
    }
}

```

```

void bluetoothTimerCallback(TimerHandle_t xTimer) {
    uint32_t startTime = millis();
    while (!digitalRead(IRSensor)) { // Timeout after 10
seconds
        String receivedMessage = Serial.readStringUntil('\n');
        Serial.println("Object detected");
        if (btScanAsync) {
            Serial.print("Starting discoverAsync... ");
            if (SerialBT.discoverAsync(btAdvertisedDeviceFound))
{
                delay(1000);
                SerialBT.discoverAsyncStop();
            } else {
                Serial.println("Error on discoverAsync");
            }
        }
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}

void IRCheck(void *pvParameters) {
    while (1) {
        //if (!digitalRead(IRSensor)) {
        if (!digitalRead(IRSensor)) {
            String receivedMessage =
Serial.readStringUntil('\n');
            if (isWordPresent(receivedMessage, "Permitted")) {
                openGate();
            }
            if (bluetoothOneShotTimer == NULL) {
                bluetoothOneShotTimer = xTimerCreate(
                    "Bluetooth One Shot",           // Timer name
                    pdMS_TO_TICKS(500),             // Timer
interval
                    pdFALSE,                      // One-shot
timer
                    NULL,                          // Timer
parameter
                    bluetoothTimerCallback        // Timer
callback function
                );
                xTimerStart(bluetoothOneShotTimer,

```

```

portMAX_DELAY); // Start the timer
}
} else {
    // Object not detected, stop the timer
    if (bluetoothOneShotTimer != NULL) {
        xTimerStop(bluetoothOneShotTimer,
portMAX_DELAY);
        xTimerDelete(bluetoothOneShotTimer,
portMAX_DELAY);
        bluetoothOneShotTimer = NULL;
    }
}
vTaskDelay(pdMS_TO_TICKS(100)); // 100ms delay
}

void setup() {
    Serial.begin(115200);
    SerialBT.begin("ESP32 Slave"); // Bluetooth device name
    Serial.println("The device started, now you can pair it with
Bluetooth!");
    pinMode(15, OUTPUT);
    pinMode(ServoVcc, OUTPUT);
    pinMode(ServoGnd, OUTPUT);
    Servo1.attach(ServoOut);
    digitalWrite(ServoVcc, HIGH);
    digitalWrite(ServoGnd, LOW);
    digitalWrite(15, HIGH);
    pinMode(IRSensor, INPUT);

xTaskCreate(IRCheck, "IRCheck", 1024, NULL, 1, NULL);
}

void loop() {
    // Your loop code here, if needed
}

```

Penjelasan:

Kode 'bluetoothMaster.ino' merupakan implementasi pada ESP32 sebagai perangkat master dalam sistem kontrol gerbang berbasis Bluetooth. Dengan menggunakan library BluetoothSerial, kode ini menginisialisasi koneksi Bluetooth dan konfigurasi perangkat keras seperti sensor infrared, motor servo untuk pengendalian gerbang, dan LED. Pada proses operasionalnya, kode secara asinkron menangani perangkat Bluetooth yang terdeteksi, memverifikasi kredensial, dan mengontrol pembukaan gerbang jika perangkat terdaftar. Sebagai pelengkap, terdapat fungsi pemantauan sensor infrared yang memicu penemuan Bluetooth saat objek terdeteksi, dengan pengelolaan waktu menggunakan timer untuk efisiensi daya.

Dengan pendekatan terintegrasi ini, solusi kontrol gerbang menyediakan manajemen akses yang efisien, memanfaatkan komunikasi Bluetooth dan sensor infrared untuk otomatisasi gerbang yang handal.

2.3 HARDWARE AND SOFTWARE INTEGRATION

Integrasi antara hardware dan software pada proyek ini dimulai dengan menghubungkan komponen-komponen fisik, seperti ESP32, servomotor, infrared sensor, dan modul Bluetooth, menggunakan kabel jumper. Kemudian di bagian software, program ESP32 harus dikembangkan untuk membaca data dari modul Bluetooth.

Software yang diimplementasikan pada ESP32, hal pertama yang dilakukan adalah menginisialisasi dan mengaktifkan modul Bluetooth untuk memfasilitasi koneksi dengan perangkat eksternal. Program kemudian melakukan pemindaian perangkat terdekat, mengambil alamat MAC dari perangkat yang mencoba masuk. Selanjutnya, terdapat implementasi pengecekan kredensial dengan membandingkan alamat MAC yang diperoleh dengan daftar terdaftar yang berisi kredensial kendaraan yang diizinkan masuk. Jika alamat MAC ditemukan dalam daftar, menandakan bahwa kendaraan terdaftar, program ESP32 mengirimkan pesan atau sinyal ke perangkat lain, seperti mikrokontroler yang mengontrol servo motor, untuk membuka pintu atau palang bedas. Sebaliknya, jika alamat MAC tidak terdaftar, langkah-langkah keamanan diambil, seperti mengunci pintu atau palang bedas, dan sistem memberikan peringatan yang sesuai. Dengan langkah-langkah ini, program ESP32 berfungsi sebagai pusat kontrol yang efektif, memastikan akses kendaraan yang terotentikasi

dan memberikan respons keamanan yang sesuai dalam lingkungan pusat atau area yang diinginkan. Integrasi kredensial Bluetooth ini membantu memastikan keamanan dan manajemen akses yang optimal dalam konteks proyek tersebut.

Program pada ESP32 memproses sinyal yang diterima dari infrared sensor dengan tujuan mendeteksi keberadaan kendaraan di sekitar palang bedas. Deteksi ini merupakan faktor kunci yang mempengaruhi pengambilan keputusan sistem. Apabila kendaraan terdaftar dan keberadaannya berhasil terdeteksi oleh infrared sensor, program akan memberikan instruksi untuk membuka palang bedas secara otomatis. Sebaliknya, jika kendaraan tidak terdaftar atau tidak terdeteksi, program akan menginstruksikan servomotor untuk menjaga pintu tetap terkunci. Selanjutnya, sistem akan merespon dengan memberikan peringatan yang sesuai. Peringatan ini dapat disampaikan melalui antarmuka pengguna, seperti lampu indikator atau layar display, atau melalui pesan yang dikirimkan ke perangkat terkait, misalnya perangkat pengelolaan keamanan. Dengan begitu, sistem memastikan bahwa setiap keputusan untuk membuka atau mengunci pintu palang bedas didasarkan pada informasi yang lengkap dan dapat diandalkan, memaksimalkan efisiensi dan keamanan kontrol akses pada proyek ini.

CHAPTER 3

TESTING AND EVALUATION

3.1 TESTING

```
Starting discoverAsync...Found a device asynchronously: Name: XFW-BT, Address: ce:9d:e5:27:13:d3, cod: 23603, rssi: -35
Nearest Device MAC Address : ce:9d:e5:27:13:d3
Access Denied
Object detected
Starting discoverAsync...Found a device asynchronously: Name: XFW-BT, Address: ce:9d:e5:27:13:d3, cod: 23603, rssi: -33
Nearest Device MAC Address : ce:9d:e5:27:13:d3
Access Denied
Opening the gate...
Access Permitted
```

Fig 5. Serial Monitor jika palang terbuka namun Bluetooth tidak terdaftar

Serial monitor menunjukkan plat nomor mobil yang tidak terdaftar. Sehingga, program melakukan scanning plat nomor mobil menggunakan kamera, saat sensor infrared mendeteksi mobil. Lalu, sistem akan mengirim pesan ke server untuk menyimpan hasil scanning plat nomor untuk autentikasi di lain waktu.

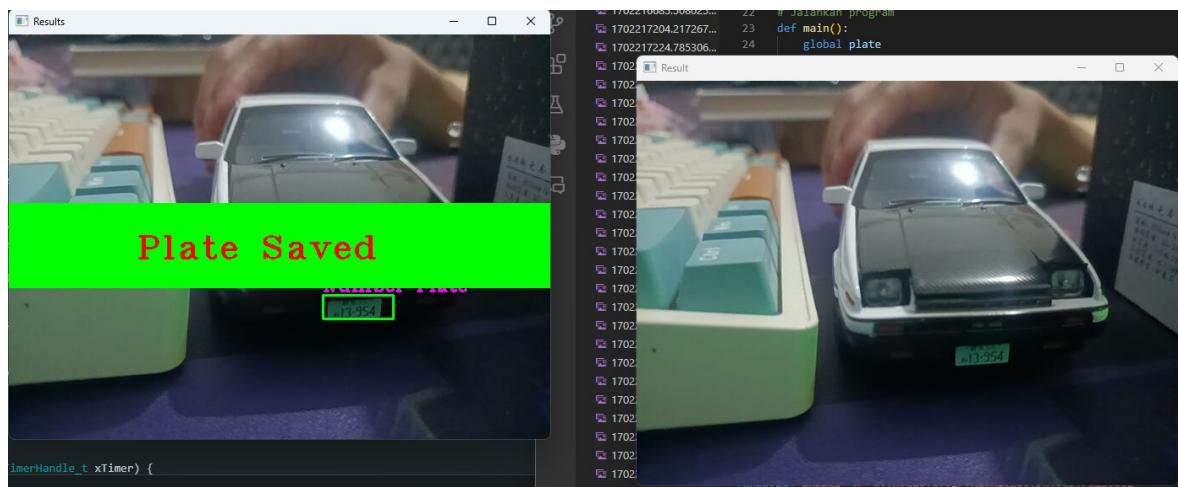


Fig 6. Plat nomor mobil dipindai dengan kamera

Kamera memindai plat nomor mobil kemudian menyimpan hasil pemindaian. Hasil pemindaian ini akan digunakan untuk sistem keamanan di lain waktu.

```
The device started, now you can pair it with Bluetooth!
Object detected
Starting discoverAsync...Object detected
Starting discoverAsync...Found a device asynchronously: Name: Sabbat X12 Pro, Address: ba:98:76:f4:08:7d, cod: 23603, rssi: -65
Object detected
Starting discoverAsync...Found a device asynchronously: Name: Sabbat X12 Pro, Address: ba:98:76:f4:08:7d, cod: 23603, rssi: -48
Nearest Device MAC Address : ba:98:76:f4:08:7d
Access Permitted
Opening the gate...
```

Fig 7. Serial Monitor jika palang terbuka dan Bluetooth terdaftar

Serial monitor menunjukkan plat nomor mobil yang terdaftar. Sehingga, program melakukan scanning plat nomor mobil menggunakan kamera, saat sensor infrared mendeteksi mobil. Lalu, sistem akan menggerakkan palang otomatis untuk posisi terbuka.

3.2 RESULT



Fig 8. Palang terbuka

Palang akan naik atau terbuka jika Bluetooth sudah terdaftar. Palang juga akan terbuka jika plat nomor kamera mengkomparasi dengan database yang ada.



Fig 9. Palang Tertutup

Palang akan tertutup jika Bluetooth tidak membaca bahwa kendaraan tersebut sudah terdaftar. Kemudian palang juga akan tertutup apabila pada saat di scan camera, plat nomor tidak sesuai dengan yang ada di database.

3.3 EVALUATION

Proyek “Barrier Gate” bertujuan untuk meningkatkan efisiensi dan keamanan pengelolaan akses ke area terbatas. Sebagai upaya pengembangan berkelanjutan, proyek ini akan melibatkan implementasi fitur lanjutan, termasuk enkripsi pesan dan integrasi model machine learning.

Pertama, pengenalan enkripsi pesan menjadi langkah kunci dalam menjaga keamanan komunikasi antara ESP32 dan perangkat terhubung. Melalui penggunaan algoritma enkripsi yang kuat, sistem memastikan bahwa data sensitif, seperti kredensial Bluetooth dan perintah sistem, tetap terlindungi dari akses yang tidak sah. Keuntungan dari langkah ini mencakup peningkatan keamanan, integritas data, dan jaminan privasi bagi pengguna.

Selanjutnya, integrasi model machine learning akan memberikan dimensi adaptif pada sistem. Dengan mengumpulkan dan menganalisis data historis, proyek ini bertujuan untuk melatih model machine learning agar dapat membuat keputusan cerdas secara real-time. Hal ini dapat meningkatkan efisiensi dan respons sistem terhadap pola penggunaan yang berubah, memungkinkan sistem untuk belajar dan beradaptasi seiring waktu.

CHAPTER 4

CONCLUSION

Secara keseluruhan, proyek Sistem Kontrol Pintu Gerbang Cerdas ini menciptakan solusi inovatif untuk efisiensi dan keamanan manajemen akses ke area terbatas. Integrasi perangkat keras seperti ESP32, servo motor, infrared sensor, Bluetooth module, dan Arduino dengan sistem perangkat lunak yang terampil membentuk dasar sistem otomatis yang andal.

Dengan fokus pada otentikasi Bluetooth dan deteksi infrared, proyek ini mengatasi tantangan dalam mengelola akses kendaraan dengan cara yang lebih cerdas dan aman. Fitur keamanan tambahan seperti enkripsi pesan dan integrasi model machine learning menghadirkan dimensi tambahan dalam melindungi data dan meningkatkan respons sistem. Evaluasi proyek menyoroti potensi peningkatan di masa depan, terutama melalui implementasi enkripsi pesan untuk memastikan keamanan komunikasi maksimal. Sementara itu, integrasi model machine learning dapat memberikan adaptabilitas sistem yang lebih baik terhadap perubahan pola penggunaan.

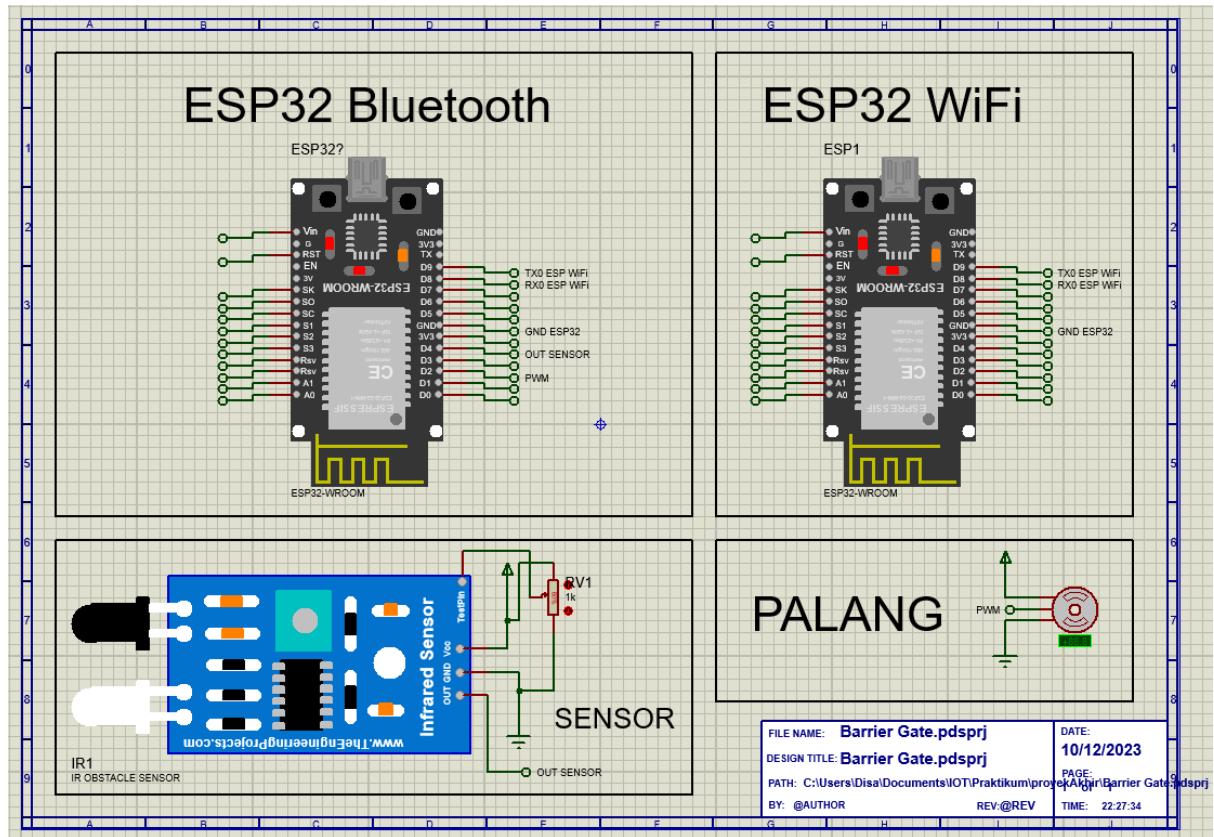
Dengan demikian, proyek ini bukan hanya menciptakan solusi kontrol pintu gerbang yang efisien, tetapi juga membuka pintu untuk pengembangan lebih lanjut yang dapat memperkuat keamanan, privasi, dan kecerdasan sistem. Keseluruhan, Sistem Kontrol Pintu Gerbang Cerdas ini mencerminkan langkah progresif menuju solusi pintu gerbang yang lebih pintar, aman, dan adaptif.

REFERENCES

- [1] DSwithBappy, “Real time car number plates extraction in 30 minutes 🔥 | OpenCV Python | Computer Vision | EasyOCR,” *YouTube*. Jan. 29, 2023. [Online]. Available: <https://www.youtube.com/watch?v=ltpnWBBT7NI>
- [2] Radhi Sghaier, “Send Commands from Python to ESP32 Wirelessly (Wi-Fi, Sockets),” *YouTube*. May 08, 2023. [Online]. Available: <https://www.youtube.com/watch?v=HLzrSOVXotw>
- [3] Xeohacker, “ESP32 Library for Proteus,” *The Engineering Projects*, Jul. 31, 2023. <https://www.theengineeringprojects.com/2023/07/esp32-library-for-proteus.html>
- [4] Syedzainnasir, “Infrared Sensor Library for Proteus,” *The Engineering Projects*, May 25, 2021. <https://www.theengineeringprojects.com/2018/07/infrared-sensor-library-for-proteus.html>
- [5] “Wi-Fi - ESP32 - — ESP-IDF Programming Guide latest documentation.” https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html
- [6] “Bluetooth® API - ESP32 - — ESP-IDF Programming Guide latest documentation.” <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/index.html>
- [7] “QuickStart - Blynk documentation.” <https://docs.blynk.io/en/getting-started/what-do-i-need-to-blynk>

APPENDICES

Appendix A: Project Schematic



Appendix B: Documentation

