Pembuatan Sistem Otomatis Pagar menggunakan Arduino dan Aplikasi Mobile

Bernadus Indra Wijaya[#1], Erico Darmawan Handoyo[*2]

[#]*Program Studi S1 Teknik Informatika, Universitas Kristen Maranatha*

Jl. Surya Sumantri No. 65, Bandung, Indonesia

[1]2072003@maranatha.ac.id

[2]erico.dh@it.maranatha.edu

Abstract **— These days, automatic gate systems are widely used. These existing automatic gate systems use a remote to control it. The remote that is used to control the gate uses a sub-1Ghz radio spectrum. The use of sub-1Ghz radio spectrum is a security issue due to the raising popularity of sub-1Ghz signal spoofing device. Another problem with current automatic gate technology is that the user must still interact with the remote to control the gate. From these problems, a new automatic gate control system will be made to hopefully solve the problems of existing automatic gate technology. The new automatic gate control system will be internet based leveraging a microcontroller that will be controlled by a mobile app. The mobile app will have features such as automatic opening of the gate using GPS, and easy access sharing. This automatic gate control system will also be equipped with a backup connection so that the gate can still be controlled in the event of internet outage.**

Keywords— **Automatic Gate, Mobile App, Microcontroller, Signal Spoofing**

I. PENDAHULUAN

Otomatis pagar merupakan sebuah sistem yang dapat membuka dan menutup pagar secara otomatis dengan sebuah remote. Teknologi ini dapat memudahkan pengguna ketika ingin keluar atau masuk rumah. Hal ini dikarenakan pengguna tidak perlu untuk mendorong pagar sendiri. Kelebihan ini akan sangat terasa ketika dalam cuaca hujan di mana pengguna dapat membuka atau menutup pintu pagarnya dari dalam mobil atau rumahnya. Namun penggunaan remote ini memiliki kekurangan.

Remote yang digunakan oleh sistem otomatis pagar yang sudah ada menggunakan sinyal radio di bawah 1Ghz (sub-1Ghz) untuk berkomunikasi dengan sistem kontrol elektroniknya. Penggunaan spektrum radio di bawah 1Ghz ini menimbulkan masalah keamanan. Hal ini dikarenakan munculnya perangkat yang dapat menangkap sinyal radio dari remote dan dapat melemparkan sinyal tiruan seperti "Flipper Zero". Hal ini dapat memungkinkan pihak lain untuk mengendalikan pagar tanpa harus memiliki remote aslinya.

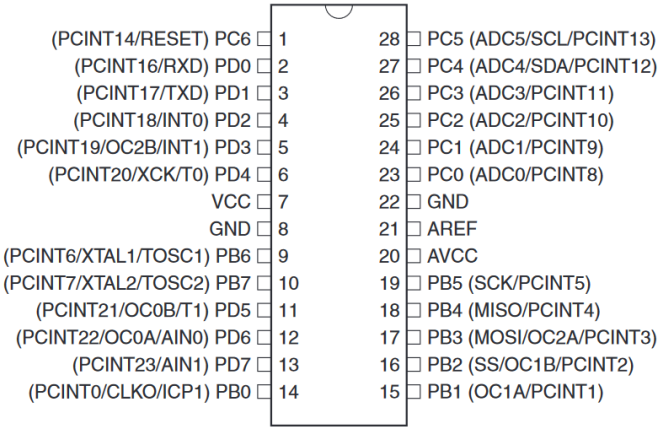
Kekurangan lain dengan penggunaan remote ini adalah pengguna harus berinteraksi dengan remote untuk membuka pagar rumahnya. Hal ini akan menyulitkan pengguna, terutama yang menggunakan kendaraan beroda dua. Selain itu, penggunaan remote ini akan menyulitkan pengguna ketika ingin memberikan akses sementara kepada tamu karena harus meminjamkan remote. Maka, berdasarkan kekurangan – kekurangan di atas, perlu dibuatkan sebuah sistem kontrol otomatis pagar baru.

Sebuah sistem kontrol otomatis pagar akan dibuat yang diharapkan dapat menangani masalah – masalah tersebut. Sistem kontrol otomatis pagar ini akan dibuat menggunakan aplikasi mobile yang akan dihubungkan ke sebuah microcontroller melalui internet.

II. KAJIAN TEORI

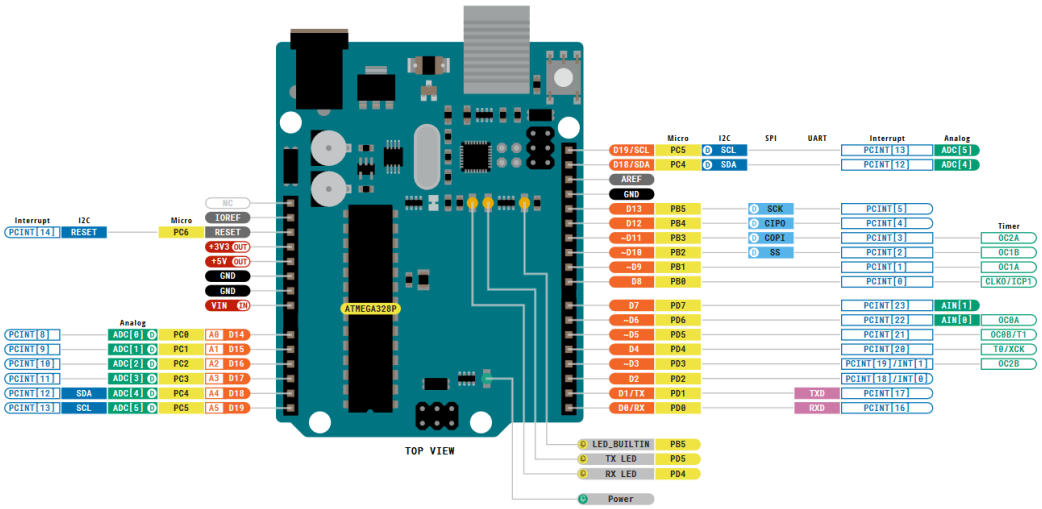
A. Microcontroller

Microcontroller merupakan sebuah processor dengan RAM dan komponen – komponen lainnya yang jadi satu dalam sebuah chip [1]. Perbedaan antara *microcontroller* dengan *microprocessor* berada pada tata letak pinnya. Sebuah *microprocessor* akan memiliki pin – pin untuk alamat memori dan juga pin – pin untuk data. Sedangkan sebuah *microcontroller* tidak memiliki pin – pin alamat memori ataupun data. Melainkan, *microcontroller* memiliki pin – pin General Purpose Input / Output (GPIO) [1] yang juga dapat memiliki fungsi tambahan seperti serial atau interrupt [2].



Gambar 1. Pin out pada microcontroller Atmega328P

Untuk dapat mempermudah pemrograman dan penggunaan *microcontroller*, dibutuhkan sebuah *printed circuit board* (PCB). PCB ini biasa disebut dengan microcontroller board. Ada banyak pembuat *microcontroller* board. Salah satunya adalah Arduino. Pada *microcontroller* board akan terdapat 2 macam GPIO pin, yaitu: analog dan digital [3].



Gambar 2. Pinout Microcontroller Board Arduino Uno R3

B. Embedded Programming

Sebuah microcontroller akan membutuhkan program untuk dapat berfungsi. Pemrograman *microcontroller* biasa dilakukan menggunakan bahasa pemrograman Assembly atau C [1]. Kode yang telah dibuat akan di-compile ke dalam bahasa mesin sesuai dengan *microcontroller* nya, lalu akan di-download ke *microcontroller* menggunakan serial atau parallel berdasarkan *microcontroller board* [4]. Ada berbagai cara kode dapat di-compile dan di-download ke *microcontroller*. Salah satunya adalah dengan menggunakan Arduino IDE.

1) Arduino Programming Environment: Arduino Programming Environment adalah salah satu cara untuk memprogram *microcontroller board* [5]. Pada Arduino Environment, bahasa yang digunakan adalah bahasa C/C++. Terdapat 2 fungsi utama yang wajib tertera pada program, yaitu “setup” dan “loop” [3]. Fungsi “setup” digunakan untuk program initialization. Fungsi ini hanya akan dijalankan sekali pada saat program pertama dijalankan oleh microcontroller. Fungsi ini dapat dimanfaatkan untuk men-set mode dari sebuah pin pada *microcontroller board* atau untuk setup koneksi serial ke komputer. Fungsi penting lainnya yaitu “loop”. Fungsi ini akan dijalankan terus menerus secara berulang selama *microcontroller board*

menyala. Fungsi ini akan berisi kode program utama. Kode program utama ini dapat berupa logika untuk menunggu input dari sebuah pin, atau dapat berupa logika yang akan set sebuah pin high atau low, dll.

```
example.ino > setup()
1   void setup() {
2       Serial.begin(115200);
3       pinMode(LED_BUILTIN, OUTPUT);
4   }
5
```

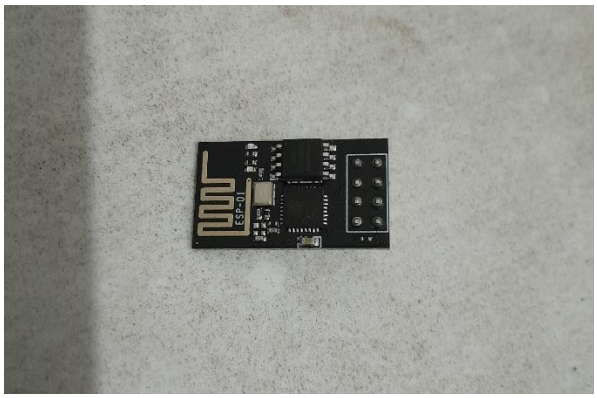
Gambar 3. Contoh kode fungsi setup

```
6   void loop() {
7       digitalWrite(LED_BUILTIN, HIGH);
8       Serial.println("LED on");
9       delay(500);
10      digitalWrite(LED_BUILTIN, LOW);
11      Serial.println("LED off");
12  }
```

Gambar 4. Contoh kode fungsi loop

C. ESP8266 SDK

ESP8266 merupakan salah satu Wi-Fi chip yang kompatibel dengan Arduino Environment Programming melalui Software Development Kit (SDK) [6]. Sama halnya dengan sebuah *microcontroller*, penggunaan Wi-Fi chip ini akan lebih mudah bila menggunakan sebuah modul [7]. Modul ini akan memiliki komponen – komponen pendukung tambahan untuk Wi-Fi chip seperti antena Wi-Fi. Salah satu contoh modul ini adalah modul Wi-Fi ESP-01.



Gambar 5. Modul Wi-Fi ESP-01

Pemrograman modul Wi-Fi memerlukan sebuah adapter “USB to Serial” jika modul Wi-Fi tidak memiliki adapter bawaan [8]. Microcontroller board Arduino memiliki adapter “USB to Serial” sehingga dapat dipakai untuk memprogram modul yang tidak memiliki adapter bawaan.

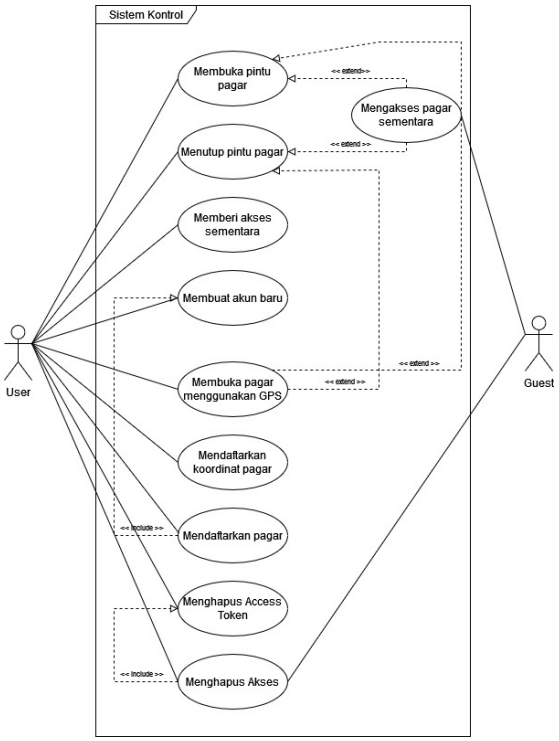
Untuk dapat melakukan pemrograman, Wi-Fi chip harus di masukan ke dalam “Bootloader mode”. Hal ini biasanya dilakukan secara otomatis bila menggunakan modul development board seperti “Sparkfun ESP8266 Thing”. Namun bila menggunakan modul yang tidak mendukung pengubahan secara otomatis, Wi-Fi chip dapat tetap di masukan ke dalam “Bootloader mode” dengan cara menyambungkan pin GPIO-0 ke pin GND, lalu reset modul dengan menyalakan ulang power supply modul [9].

1) Received Signal Strength Indicator: Received Signal Strength Indicator (RSSI) adalah sebuah indikator untuk mengukur kekuatan sinyal pada perangkat penerima. Secara garis besar, nilai dari RSSI akan bertambah atau berkurang sesuai dengan jarak relatif antara perangkat pemancar sinyal dan penerima sinyal [10]. Pada umumnya, nilai RSSI memiliki satuan dBm (Decibel for milliwatt). Nilai ini pada umumnya memiliki skala antara -60 – 0, -100 – 0, atau -127 – 0 tergantung pada perangkatnya [10].

III. ANALISIS DAN RANCANGAN SISTEM

A. Use Case Diagram

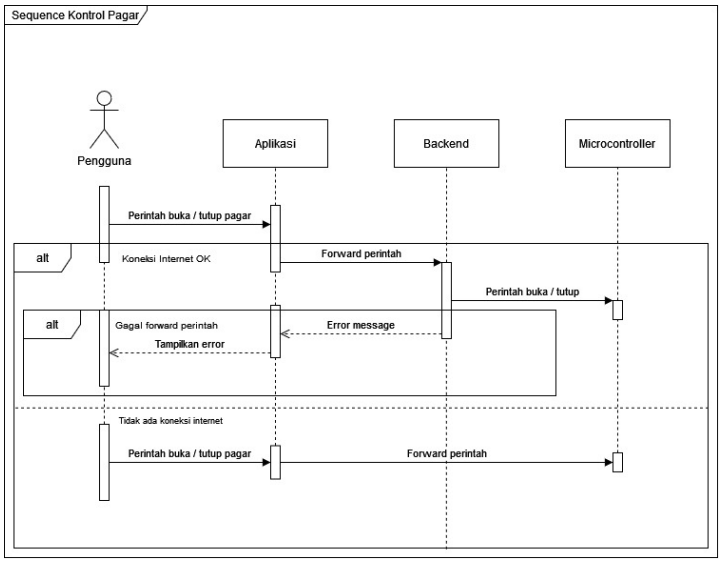
Pada proyek ini, pengguna (User) dapat menggunakan fitur – fitur seperti: membuka / menutup pintu pagar, memberi akses pintu pagar sementara kepada orang lain, dan membuat akun baru. Sedangkan untuk orang lain (Guest) dapat mengakses pagar untuk sementara dengan seizin pengguna utama (User).



Gambar 6. Use Case Diagram Proyek

B. Desain Komunikasi Aplikasi dengan Microcontroller

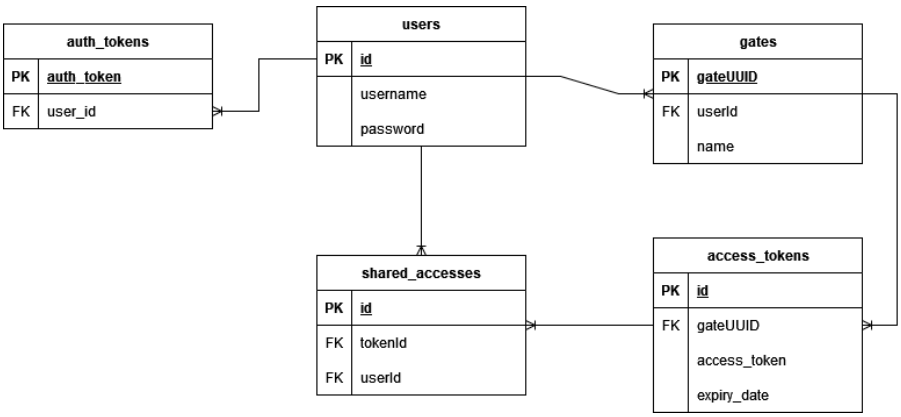
Komunikasi antara aplikasi dengan *microcontroller* dapat dijalin dengan 2 cara, yaitu melalui internet, atau secara lokal. Sistem akan berkomunikasi melalui internet pada kondisi normal. Namun bila koneksi internet pengguna terputus, maka sistem akan mencoba untuk berkomunikasi secara lokal dengan *microcontroller* sehingga pengguna tetap dapat mengakses pagar ketika koneksi internet bermasalah.



Gambar 7. Sequence Diagram Komunikasi Sistem

C. Desain Basis Data Backend

Tugas dari *backend* pada proyek ada 2, yaitu: meneruskan perintah dari aplikasi ke *microcontroller*, dan menangani akses sementara kontrol pintu pagar oleh pengguna lain. Dari 2 tugas ini, dibutuhkan sebuah basis data untuk menyimpan data – data akun pengguna utama (pemilik), data pagar yang terdaftar pada akun, dan data access token sementara untuk pagar sebuah pagar.



Gambar 8. Entity Relationship Diagram Proyek

IV. IMPLEMENTASI

A. Penjelasan Keseluruhan Sistem

Implementasi proyek tugas akhir ini dibagi menjadi 3 modul yang akan saling bekerja sama untuk membangun sebuah sistem otomatis pagar. Ketiga modul itu adalah: Backend, Mobile App, Microcontroller Firmware.

B. Tabel Implementasi Komunikasi Lokal

Komunikasi lokal antara aplikasi mobile dengan microcontroller dilakukan menggunakan protokol UDP. Aplikasi akan mengirimkan perintah dalam bentuk string untuk diproses oleh *microcontroller*.

Berikut adalah tabel perintah – perintah yang tersedia untuk koneksi lokal. Tulisan yang bercetak tebal merupakan parameter:

TABEL I
DAFTAR PERINTAH KONEKSI LOKAL

No	Perintah	Fungsi
1	"CON:**SSID**:**PASS**"	Perintah untuk koneksikan modul Wi-Fi ke SSID dan PASS yang diberikan.
2	"REF"	Perintah untuk memuat ulang daftar Wi-Fi yang terdeteksi oleh modul Wi-Fi.
3	"OPN:**UUID**"	Perintah untuk membuka pagar. Dilengkapi dengan cek UUID untuk memastikan perintah berasal dari pemilik pagar.
4	"CLS:**UUID**"	Perintah untuk menutup pagar. Dilengkapi dengan cek UUID untuk memastikan perintah berasal dari pemilik pagar.

Implementasi komunikasi UDP ini dilakukan dengan memanfaatkan *sockets*. Untuk dapat menggunakannya, sebuah *socket* harus diperintahkan untuk beroperasi pada port jaringan tertentu. Dalam proyek ini, port yang digunakan adalah 8585 (untuk mengirimkan perintah ke *microcontroller*), dan 8586 (untuk mengirimkan data Wi-Fi ke aplikasi *mobile*).

```
if (!udp.begin(UDP_RECV)) {
    error = true;
    return;
}
```

Gambar 9. Potongan Kode Inisiasi Socket UDP

Perintah yang telah dikirimkan ke *microcontroller* harus diproses kembali oleh *microcontroller* untuk mendapatkan parameternya dengan menggunakan *tokenizer*.

```
char* command[3];   // alocate for 3 tokens (max posible token from app)

if (udp.parsePacket()) {
    String udpMessage = udp.readStringUntil(';');

    command[0] = strtok((char*) udpMessage.c_str(), ":");

    if (!strcmp(command[0], "CON")) {
        connectNetwork(command);
    } else if (!strcmp(command[0], "REF")) {
        refreshNetworks();
    } else if (!strcmp(command[0], "OPN")) {
        open(command);
    } else if (!strcmp(command[0], "CLS")) {
        close(command);
    }
}
```

Gambar 10. Potongan Kode Tokenizer Perintah

C. Tabel Implementasi Komunikasi WebSocket

Komunikasi WebSocket antara aplikasi *mobile* dengan *microcontroller* dilakukan melalui modul *backend* sebagai penengah. Aplikasi akan mengirimkan request HTTP kepada *backend*, lalu *backend* akan meneruskan perintah melalui WebSocket ke *microcontroller* dalam bentuk perintah string.

TABEL II
DAFTAR PERINTAH KONEKSI WEBSOCKET

No	Perintah	Fungsi
1	"OPEN"	Perintah untuk buka pagar.
2	"CLOSE"	Perintah untuk tutup pintu pagar.

Sebelum sebuah WebSocket bisa dipakai, perlu adanya koneksi ke sebuah server WebSocket. Pada proyek ini, WebSocket akan beroperasi pada port 8001.

```
webSocket.beginSSL(BACKEND_ADDRESS, 11147, "/", nullptr, "");

webSocket.onEvent(handleWebsocket);
```

Gambar 11 Potongan Kode untuk Menyambung ke Server WebSocket

```
void handleWebsocket(WStype_t type, uint8_t * payload, size_t length) {
    switch (type) {
    case WStype_CONNECTED:
        // Connection established to backend, send UUID
        webSocket.sendTXT("UUID:" + String(UUID));
    break;
    case WStype_TEXT:
        // Command from backend
        String cmd = (char *) payload;

        if (cmd == "OPEN") {
            Serial.write("OPN;");
        } else if (cmd == "CLOSE") {
            Serial.write("CLS;");
        }
    break;
    }
}
```

Gambar 12. Potongan Kode untuk Memproses Perintah Backend

D. Implementasi Backend

Modul ini akan berfungsi sebagai penengah antara aplikasi *mobile* dengan *microcontroller* untuk memungkinkan komunikasi melalui internet. Modul *Backend* akan dibuat menggunakan *framework* Laravel dan akan berkomunikasi dengan aplikasi mobile melalui API JSON. Sedangkan komunikasi dengan *microcontroller* akan dilakukan menggunakan WebSocket.

1) Implementasi Komunikasi antar Laravel dengan WebSocket: Komunikasi antara web server (Laravel) dengan WebSocket server dilakukan dengan memanfaatkan WebSocket untuk pengiriman perintah dari API Backend ke WebSocket server.

```
// Connect to WebSocket server and tell it to open the gate
$client = new Client("wss://127.0.0.1:21");
$client->setContext([
    "ssl" => [
        "verify_peer" => false,
        "verify_peer_name" => false
    ]
]);
$client->text("OPEN:" . $validated["gate_uuid"]);
$client->close();
```

Gambar 13. Potongan Kode untuk Koneksi Backend dengan WebSocket

E. Implementasi Firmware Modul Wi-Fi

Firmware modul Wi-Fi bertugas untuk menerima perintah untuk buka atau tutup pagar melalui WebSocket ataupun melalui socket UDP. Perintah ini akan diteruskan ke *microcontroller* dengan sambungan Serial. Sebelum komunikasi serial dapat dilakukan, kedua perangkat yang ingin berkomunikasi harus diatur agar memiliki baud rate yang sama. Dalam proyek ini, baud rate yang dipilih adalah 115200 yang di-*set* pada awal inisiasi firmware microcontroller dan modul Wi-Fi.

```
void setup() {
    Serial.begin(115200);   // For communication with Arduino
```

Gambar 14. Potongan Kode untuk Set Baud Rate

Setelah baud rate di-*set*, komunikasi dapat dilakukan dengan mengirimkan data yang dapat berupa string atau data biner.

```
if (cmd == "OPEN") {
    Serial.write("OPN;");
} else if (cmd == "CLOSE") {
    Serial.write("CLS;");
}
eak;
```

Gambar 15. Potongan Kode untuk Mengirimkan Data menggunakan Serial

F. Implementasi Firmware Microcontroller (Arduino)

Dalam proyek ini, *firmware* Arduino bertugas untuk mengeksekusi perintah buka atau tutup pagar yang didapatkan dari modul Wi-Fi. Setelah menerima perintah dari modul Wi-Fi, *firmware* akan menyalakan atau mematikan pin untuk *relay* sesuai dengan perintahnya. Selain itu, *firmware* juga akan memonitor pin untuk sensor buka atau tutup pagar untuk memberhentikan pagar ketika sudah sepenuhnya terbuka atau tertutup.

Pin yang digunakan untuk memerintahkan sistem penggerak adalah pin D2 (buka), dan D3 (tutup). Kedua pin ini akan di-set modenya menjadi “output”.

```
pinMode(RELAY_OPEN_PIN, OUTPUT);
pinMode(RELAY_CLOSE_PIN, OUTPUT);
```

Gambar 16 Potongan Kode untuk Set Mode Pin

```
if (!open) {
    digitalWrite(RELAY_OPEN_PIN, HIGH);
} else {
    digitalWrite(RELAY_OPEN_PIN, LOW);
}

if (!close) {
    digitalWrite(RELAY_CLOSE_PIN, HIGH);
} else {
    digitalWrite(RELAY_CLOSE_PIN, LOW);
}
```

Gambar 17. Potongan Kode untuk Membuka / Menutup Pagar

Pin yang digunakan untuk sensor pagar adalah pin D4 (Sensor buka), dan D5 (Sensor tutup). Kedua pin ini akan di-set modenya menjadi “input”. Kedua pin ini akan terus mengecek apakah pagar sudah terbuka atau tertutup sepenuhnya.

```
pinMode(OPEN_L_SW, INPUT);
pinMode(CLOSE_L_SW, INPUT);
```

Gambar 18 Potongan Kode untuk Set Pin Sensor

```
if (digitalRead(OPEN_L_SW) == HIGH) {
    open = false;
}

if (digitalRead(CLOSE_L_SW) == HIGH) {
    close = false;
}
```

Gambar 19. Potongan Kode untuk Pengecekan Sensor

G. Implementasi Aplikasi Mobile

Aplikasi *mobile* pada proyek ini bertugas untuk mengirimkan perintah buka atau tutup pagar dari *smartphone* pengguna. Perintah dapat dikirimkan melalui sambungan lokal UDP atau melalui internet.

Saat pengguna memerintahkan untuk membuka atau menutup pagar, program akan mengecek terlebih dahulu apakah ada koneksi ke *backend* dari *smartphone* pengguna sebelum menentukan koneksi yang akan digunakan untuk mengirimkan perintah.

```
final ping = await Ping("node46095-vpstest.user.cloudjkt01.com", count: 1).stream.first;
final Auth cachedAuth = await const AuthCache().read();

if (ping.response != null) {
  // Internet connection OK, use HTTP + WebSocket.
  GateCommandWrapper gateCmd =
      GateCommandWrapper(repo: GateWebSocketCommand());
  gateCmd.open(uuid, cachedAuth.authToken!);
} else {
  // Internet connection not OK, use local UDP.
  GateCommandWrapper gateCmd = GateCommandWrapper(repo: GateUDPCommand());
  gateCmd.open(uuid, cachedAuth.authToken!);
}
```

Gambar 20. Potongan Kode untuk Menentukan Koneksi yang Dipakai

Koneksi lokal UDP akan digunakan ketika aplikasi *mobile* tidak dapat terhubung ke *backend*. Koneksi lokal UDP akan menggunakan *socket* yang akan dihubungkan ke *broadcast address* dari jaringan Wi-Fi pengguna dengan *port* 8585.

```
@override
Future<void> open(String uuid, String authToken) async {
  RawDatagramSocket socket = await RawDatagramSocket.bind("0.0.0.0", 8585);
  String broadcastAddr = await NetworkInfo().getWifiBroadcast() ?? "0.0.0.0";
  socket.broadcastEnabled = true;
  socket.send("OPN:$uuid;".codeUnits, InternetAddress(broadcastAddr), 8585);
  socket.close();
}
```

Gambar 21. Potongan kode untuk Koneksi Lokal UDP

Koneksi melalui internet akan digunakan ketika aplikasi *mobile* dapat terhubung ke *backend*. Koneksi internet ini akan mengirimkan *request* HTTP yang akan diproses oleh *backend* sebelum di kirimkan melalui WebSocket ke *microcontroller*.

```
@override
Future<void> open(String uuid, String authToken) async {
  try {
    await post(
      headers: {
        "Accept": "application/json",
      },
      Uri.parse("$backendAddress/gate/open"),
      body: {
        "gate_uuid": uuid,
        "auth_token": authToken
      }
    );
  } on Exception catch (_) {}
}
```

Gambar 22. Potongan Kode untuk Koneksi melalui Internet

H. Implementasi Sharing Access Sementara

Sharing access sementara pada sistem ini akan menggunakan teknik pembagian *link access* yang mirip seperti Google Drive. Pengguna utama akan memilih pagar yang ingin dibagikan aksesnya lalu akan mendapatkan sebuah *link* yang dapat dibagikan ke orang – orang yang diinginkan. *Link* yang dibuat oleh sistem akan ditangani oleh aplikasi *mobile* ketika dibuka

oleh pengguna lain. Aplikasi akan mengambil *access token* dari *link* tersebut dan akan mengirimkan sebuah *request* ke *backend* untuk divalidasi. Jika validasi berhasil, maka akun yang membuka *link* tersebut akan diberikan akses.

I. Implementasi Auto Unlock GPS

Implementasi pembukaan pagar secara otomatis menggunakan GPS pada proyek ini memanfaatkan Android Service API untuk menjalankan kode program walaupun aplikasi utama sudah ditutup, dan API untuk membuat *geofence* dari Android. Android Service API ini menggunakan bahasa pemrograman Kotlin, sehingga dibutuhkan sebuah "Method Channel" agar kode Dart dapat berkomunikasi dengan kode Kotlin.

V. PENGUJIAN

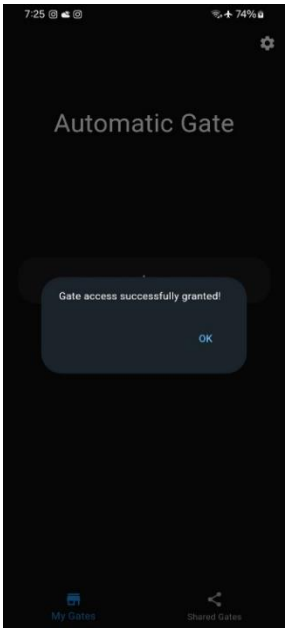
A. Pengujian Fitur Sharing Access

Pengujian ini bertujuan untuk membuktikan kemampuan sistem untuk membagikan akses pagar secara sementara kepada orang lain dengan aman. Pengujian fitur *sharing access* akan menggunakan sebuah *smartphone* Android dan juga sebuah emulator Android sebagai representasi perangkat kedua.

Kedua perangkat Android ini akan mewakili pihak – pihak sebagai berikut:

1. *Emulator* Android sebagai pihak pemilik pagar
2. *Smartphone* Android sebagai pihak yang akan diberikan akses (disebut juga sebagai pihak lain)

1) Metode Pengujian: Pengujian diawali dengan menjalankan fitur *sharing access* pada emulator Android. Link yang didapatkan akan dibuka pada *smartphone* yang belum ada akun tersimpan untuk menguji apakah pembagian akses berhasil atau tidak terkhususnya ketika pihak lain masih belum memiliki akun di mana aplikasi harus dapat memastikan bahwa akses yang dibagikan betul – betul berhasil diberi kepada pihak lain.



Gambar 23. Tangkapan Layar ketika Akses Berhasil Diberikan

B. Pengujian Fitur Auto Unlock GPS

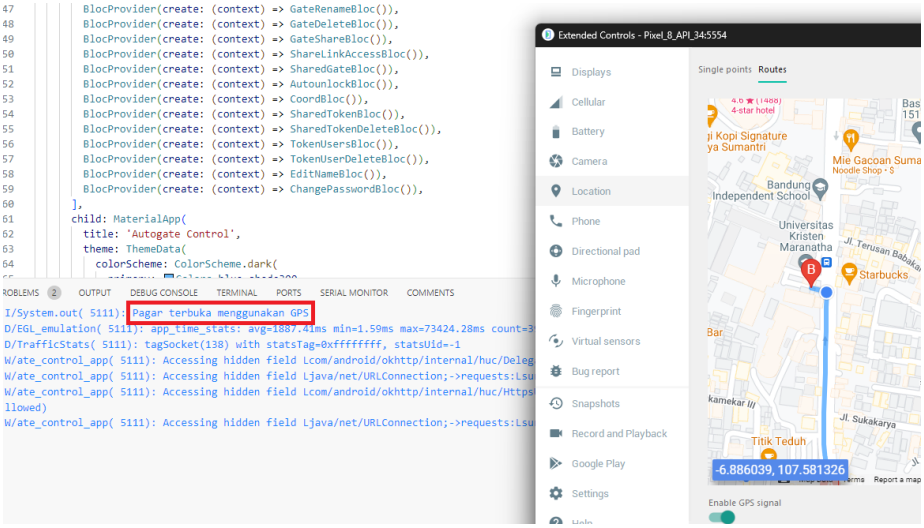
Pengujian ini bertujuan untuk membuktikan fungsi dari fitur pembukaan pagar secara otomatis berdasarkan posisi GPS *smartphone* pengguna. Pengujian akan dilakukan menggunakan emulator Android dan memanfaatkan fitur *mock location* untuk menyimulasikan perjalanan menuju rumah pemilik pagar.

Untuk mempermudah pengujian, aplikasi akan ditambahkan kode untuk *print* ke debug *console* sebagai representasi dari respons sistem atas kedatangan pemilik rumah.

```
if (transition == Geofence.GEOFENCE_TRANSITION_ENTER) {
    val uuid = intent.getStringExtra( name: "uuid")
    val authToken = intent.getStringExtra( name: "auth_token")

    MainActivity.methodChannel.invokeMethod( method: "openGate",  arguments: null)

    openGate(uuid!!, authToken!!)
    println("Pagar terbuka menggunakan GPS")

    if (context != null) {
        val serviceIntent = Intent(context, AutounlockService::class.java)
        context.stopService(serviceIntent)
    }
}
```

Gambar 24. Tambahan Kode Print untuk Pengujian

1) Metode Pengujian: Pengujian akan diawali dengan pendaftaran titik koordinat pagar. Setelah koordinat terdaftar, maka *service* untuk pembukaan pagar otomatis akan dijalankan. Lalu emulator Android akan menyimulasikan kedatangan ke rumah dengan menggunakan fitur *mock location*.



Gambar 25. Tangkapan Layar Debug Console ketika Simulasi Sudah Sampai

C. Pengujian Koneksi Lokal Cadangan

Pengujian ini bertujuan untuk membuktikan fitur koneksi cadangan pada sistem agar pagar tetap dapat dikendalikan ketika ada kendala jaringan internet.

Berikut adalah peran perangkat – perangkat yang digunakan untuk pengujian ini:

1. Perangkat Android pertama sebagai perangkat yang menjalankan aplikasi *mobile*
2. Perangkat Laptop Windows sebagai perangkat yang akan menjadi Wi-Fi *hotspot* untuk menyimulasikan masalah koneksi

1) Pengujian Ketika hanya Internet Nonaktif: Pengujian ini akan dilakukan untuk menguji fungsi dari fitur koneksi cadangan ketika *microcontroller* masih tersambung ke jaringan Wi-Fi rumah, namun koneksi ke internet tidak ada atau bermasalah. Untuk menyimulasikan koneksi internet yang bermasalah, kode program aplikasi *mobile* akan diubah agar mengecek IP *address* yang salah sehingga aplikasi akan mengira bahwa koneksi ke *backend* bermasalah. Untuk dapat membuktikan bahwa aplikasi *mobile* berhasil untuk menggunakan koneksi cadangan, maka akan ditambahkan kode untuk print ke *debug console* ketika aplikasi akan mengirimkan perintah buka / tutup pagar.

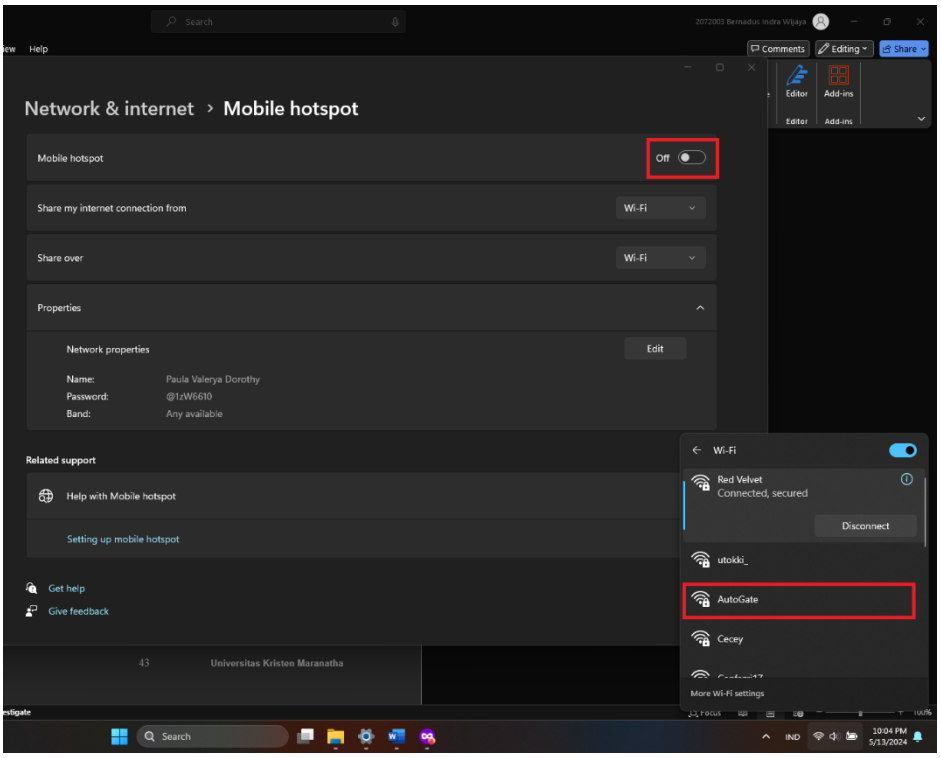
```
if (ping.response != null) {
  // Internet connection OK, use HTTP + WebSocket.
  GateCommandWrapper gateCmd =
      GateCommandWrapper(repo: GateWebSocketCommand());
  print("Buka pagar melalui internet");
  gateCmd.open(uuid, cachedAuth.authToken!);
} else {
  // Internet connection not OK, use local UDP.
  GateCommandWrapper gateCmd = GateCommandWrapper(repo: GateUDPCommand());
  print("Buka pagar melalui koneksi lokal (cadangan)");
  gateCmd.open(uuid, cachedAuth.authToken!);
}
```

Gambar 26. Potongan Kode untuk Pembuktian Koneksi yang Digunakan

2) Metode Pengujian: Pengujian akan diawali dengan menekan tombol "Open" pada aplikasi *mobile* dan memonitor *debug console* untuk memastikan koneksi melalui HTTP berjalan lancar. Sesudah itu, server *backend* akan dimatikan untuk menyimulasikan masalah koneksi internet. Setelah *backend* dimatikan maka tombol "Open" akan ditekan kembali dan memonitor *debug console* untuk memastikan aplikasi menggunakan koneksi cadangannya.

Gambar 27. Tangkapan Layar Aplikasi Menggunakan Koneksi Lokal

3) Pengujian Ketika Access Point Wi-Fi Nonaktif: Pengujian ini akan dilakukan untuk menguji fungsi dari fitur koneksi cadangan ke-2 ketika *microcontroller* tidak dapat terhubung ke jaringan Wi-Fi rumah di mana *microcontroller* harus membuat jaringan Wi-Fi *hotspot*-nya sendiri. Untuk dapat menyimulasikan masalah koneksi jaringan Wi-Fi rumah, maka pengujian ini akan menggunakan sebuah perangkat laptop Windows untuk dijadikan Wi-Fi *hotspot* yang akan berperan sebagai jaringan Wi-Fi rumah.



Gambar 28. Tangkapan Layar saat Microcontroller Tidak Terhubung ke Wi-Fi

VI. SIMPULAN DAN SARAN

A. *Simpulan*

Sistem kontrol otomatis pagar ini juga dapat mengatasi masalah pada sistem kontrol pagar konvensional yang masih menggunakan *remote* dengan frekuensi radio di bawah 1Ghz dengan menggunakan sistem kontrol berbasis IoT dan memanfaatkan keamanan – keamanan yang tersedia dari penggunaan teknologi IoT.

Proyek sistem kontrol otomatis pagar juga ini diharapkan dapat mempermudah pengguna ketika ingin mengoperasikan pagar rumahnya dengan fitur – fitur seperti pembukaan pagar secara otomatis menggunakan GPS dari *smartphone* pengguna dengan mengecek secara berkala posisi dari *smartphone* pengguna, dan ketika posisi sudah di dekat pagar, maka sistem akan membukakan pagar secara otomatis. Selain itu, pagar juga dapat dikendalikan dari jarak jauh melalui internet, dan juga dapat dibagikan aksesnya secara sementara ke pengguna lain.

B. *Saran*

Setelah pengerjaan proyek dan pengujian sistem dari proyek ini, ditemukan beberapa saran bagi yang ingin mengembangkan lagi sistem kontrol otomatis pagar ini di masa depan berdasarkan keterbatasan – keterbatasan yang masih ada dalam proyek ini:

1. Fitur auto unlock GPS masih harus dinyalakan secara manual setiap kali pengguna ingin menggunakannya. Fitur ini akan secara otomatis mati kembali ketika sudah membukakan pagar secara otomatis. Hal ini bertujuan untuk menghindari kasus di mana pengguna melewati dekat rumah untuk ke tujuan lain dan pagar akan terbuka secara otomatis tanpa kesadaran pengguna. Dengan ini, maka disarankan untuk mengimplementasikan *event geofence* on exit pada fitur auto unlock sehingga *service* dari fitur auto unlock tidak perlu dinyalakan secara manual lagi setiap kali pengguna ingin menggunakan fiturnya.
2. Logika pengecekan koneksi internet pada aplikasi hanya menggunakan teknik *ping* ke server *backend*. Dengan cara ini, aplikasi hanya dapat mengecek apakah server sedang *online* atau tidak. Aplikasi tidak dapat mengecek apakah *backend* benar – benar siap untuk menerima *request* atau tidak. Dengan ini, maka disarankan untuk mengimplementasikan sistem pengecekan yang lebih dapat diandalkan dengan cara membuat sebuah *method* baru pada *backend* yang akan dipanggil melalui HTTP *request* dari aplikasi dan mengembalikan sebuah nilai untuk menandakan bahwa *backend* benar – benar siap untuk menerima *request* dari aplikasi.
3. Ketika sebuah *access token* hangus, *backend* hanya menyembunyikan *token* tersebut dari pengguna, dan tidak menghapusnya dari *database*. Hal ini dapat menyebabkan *database* penuh dalam penggunaan jangka panjang. Dengan

ini, maka disarankan untuk mengimplementasikan sebuah sistem untuk mengecek *token* pada *database* secara berkala, dan hapus *token* yang sudah hangus dari *database* sehingga penggunaan tempat penyimpanan dapat lebih efisien.

4. Fitur untuk menampilkan *state* pengoperasian pagar masih belum dapat menangani kasus ketika pagar sudah sepenuhnya terbuka atau tertutup. Sehingga aplikasi akan mengira bahwa pagar masih membuka atau menutup dan menampilkan tulisan "pause". Dengan ini, maka disarankan untuk mengimplementasikan komunikasi balik dari *microcontroller* ke aplikasi *mobile*, serta mengubah agar komunikasi aplikasi ke *backend* dilakukan melalui WebSocket sehingga memungkinkan *microcontroller* untuk memperbarui *state* pada aplikasi secara *real time*.

VII. DAFTAR PUSTAKA

[1] G. Gridling and B. Weiss, "Introduction to microcontrollers," *Vienna University of Technology Institute of Computer Engineering Embedded Computing Systems Group*, p. 25, 2007.

[2] Microchip Technology Inc., *megaAVR® Data Sheet*, 2020.

[3] A. S. Ismailov and Z. B. Jo'Rayev, "Study of arduino microcontroller board," *Science and Education*, vol. III, no. 3, pp. 172-179, 2022.

[4] T. VanSickle, Programming microcontrollers in C, Virginia: Newnes, 2001.

[5] R. K. Kodali and K. S. Mahesh, "Low cost ambient monitoring using ESP8266," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, Warangal, 2016.

[6] I. Grokhotkov, "ESP8266 arduino core documentation," *ESP8266*, 2017.

[7] M. Bazzi and H. Mohammad, "Motor Current Signature Analysis using Microcontroller with WIFI Connection," 2021.

[8] M. Schwartz, ESP8266 Internet of Things Cookbook, Birmingham: Packt Publishing Ltd, 2017.

[9] M. Schwartz, Internet of Things with ESP8266, Birmingham: Packt Publishing Ltd, 2016.

[10] A. Suarez, J. A. Santana, E. Marcías, V. E. Mena, J. M. Canino and D. Marrero, "RSSI prediction in WiFi considering realistic heterogeneous restrictions," *Network Protocols and Algorithms*, vol. VI, no. 4, pp. 19-40, 2014.