BAB I
PENDAHULUAN

1.1 Latar Belakang

Penggunaan teknologi *Internet of Thing* (IoT) terus berkembang secara signifikan seiring dengan kemajuan zaman. Berbagai aspek kehidupan seperti *Smart Cities* dan *Smart Transportation* sudah banyak memakai teknologi IoT. Diprediksi, akan terdapat lima ratus miliar perangkat IoT pada tahun 2030 dan hingga 90% dari seluruh kendaraan yang ada di dunia akan dihubungkan dengan IoT [1]. Penerapan koneksi nirkabel saat ini sedang mengalami perkembangan pesat. Salah satu perangkat IoT yang sering digunakan adalah ESP32 *family* milik Espressif.

Komunikasi antar perangkat dapat dilakukan menggunakan kabel atau nirkabel dengan kelebihan serta kekurangan masing-masing. Sejak awal kemunculan komputer hingga sekarang, penggunaan kabel masih sangat umum karena menawarkan performa tinggi dengan latensi yang rendah karena perangkat terhubung secara fisik. Tetapi, penggunaan kabel memiliki keterbatasan pada mobilitas dan fleksibilitas yang memengaruhi kepraktisan. Koneksi nirkabel menawarkan keunggulan jauh pada aspek kepraktisan dengan mengorbankan sedikit performa karena secara mutlak koneksi nirkabel tidak dapat mengungguli koneksi yang menggunakan kabel dalam hal performa. Keandalan dari koneksi nirkabel juga masih di bawah koneksi kabel karena lebih banyak aspek yang dapat memengaruhi kualitas koneksi nirkabel, seperti yang dijelaskan pada penelitian [2]. Sehingga, ketersedian opsi koneksi kabel pada perangkat nirkabel dapat memberikan keamanan ekstra sebagai rencana cadangan apabila koneksi nirkabel bermasalah.

Jenis koneksi nirkabel yang digunakan juga dapat memengaruhi aspek-aspek yang telah disebutkan sebelumnya. Pada ranah IoT, jenis koneksi yang paling sering digunakan adalah Wi-Fi dan *Bluetooth. Bluetooth* menawarkan kepraktisan dalam proses penyetelannya dan berkecukupan untuk sebagian besar jenis proyek IoT. Akan tetapi, *Bluetooth* memiliki jarak jangkauan "cukup" dan *bandwidth* kecil sehingga kurang cocok untuk transfer data intens, terlebih pada jarak yang cukup

jauh. Wi-Fi hadir dengan memberikan performa yang lebih baik daripada *Bluetooth* di segala aspek [3]. Sehingga Wi-Fi lebih cocok untuk komunikasi dengan banyak data sekaligus, seperti komunikasi *real-time*.

Untuk mendukung konsep "*real*-time", terdapat dua protokol yang sering digunakan pada IoT dengan dasar koneksi Wi-Fi, yaitu HTTP (*polling dan long polling)* serta WebSocket. WebSocket memberikan fitur komunikasi *full duplex* yang tidak memerlukan *request* dan *response* ketika melakukan komunikasi data. Berbeda dengan *polling* dan *long polling* yang mengirim data menunggu *request.* Karena WebSocket tidak menutup koneksi ketika transmisi data selesai, sehingga latensi yang dihasilkan menjadi lebih rendah dan lebih cocok untuk transmisi data *real-time* [4].

Adanya teknologi yang dapat mendukung konsep "*real-time"* ini mendorong pesatnya perkembangan IoT, termasuk *Smart Transportation*. Kenyamanan dalam berkendara saat ini sudah ikut menjadi pertimbangan karena kemudahan dalam pengembangannya berkat teknologi IoT. *Quickshifter* adalah salah satu teknologi tersebut. Bekerja sebagai bantuan untuk berpindah gigi pada sepeda motor tanpa memerlukan penggunaan kopling guna meningkatkan kenyamanan [5].

Sebagai akibat dari keterbatasan finansial, konsumen terdorong untuk berpikir kreatif dan menciptakan teknologi *quickshifter* buatan mereka sendiri. Secara teknis, modul dapat diciptakan menggunakan sebuah mikrokontroler seperti Arduino Uno dengan bantuan beberapa perangkat elektronik lain [6]. Modul *quickshifter* yang lebih terjangkau dapat diaplikasikan secara universal kepada semua jenis sepeda motor.

Untuk memaksimalkan kinerja dari *quickshifter*, diperlukan kalkulasi lama pemutusan pengapian (*kill time)* berdasarkan RPM mesin terkini. Modul disertai *mapping kill time* yang dapat diubah-ubah sesuai dengan kebutuhan pengguna. Penyetelan dilakukan sembari memantau RPM agar angka *kill time* optimal dapat ditemukan, sehingga modul harus mampu memberikan bacaan dan menyampaikan RPM ke pengguna secara *real-time*.

Pada penelitian sebelumnya [6], pengawasan RPM dan penyesuaian setelan modul masih menggunakan kabel USB yang terhubung langsung ke mikrokontroler. Meskipun ini memungkinkan komunikasi *real-time* dengan latensi

rendah, pendekatan ini masih kurang praktis. Metode komunikasi nirkabel yang lebih praktis dapat dicapai dengan menggunakan koneksi Wi-Fi 2.4G menggunakan mikrokontroler ESP32 melalui protokol WebSocket [4]. Nilai latensi yang dihasilkan juga masih dalam batas toleransi nyaman untuk dipakai.

Maka dari itu, penelitian ini bertujuan untuk merancang dan mengembangkan komunikasi *real-time* melalui WebSocket dan USB pada sistem dual-mikrokontroler menggunakan studi kasus modul quickshifter guna meningkatkan efisiensi dan kepraktisan dalam penyesuaian modul. Menggunakan kombinasi dua mikrokontroler Arduino Nano dan ESP32-C3 dengan peran fungsional masing-masing. Arduino Uno akan berfungsi sebagai komputasi utama mengenai segala hal yang berhubungan dengan teknis *quickshifter*, sedangkan ESP32-C3 akan mengambil alih semua yang berhubungan dengan komunikasi kepada pengguna layaknya *gateway*. WebSocket akan digunakan sebagai protokol utama dengan USB sebagai *fallback* atau jaringan keselamatan apabila Wi-Fi mengalami kerusakan, sehingga modul tetap dapat dilakukan penyesuaian walaupun pada skenario Wi-Fi bermasalah. Dengan adanya penelitian ini, diharapkan dapat memberikan kepraktisan kepada masyarakat untuk melakukan komunikasi dan penyesuaian modul *quickshifter* milik mereka, terutama yang menggunakan mikrokontroler keluarga Arduino.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, penulis merumuskan beberapa permasalahan utama sebagai berikut:

1. Bagaimana rancangan dari komunikasi WebSocket menggunakan USB *fallback* dual mikrokontroler Arduino Nano dan ESP32-C3 pada studi kasus modul *quickshifter*?
2. Bagaimana implementasi komunikasi WebSocket dengan USB *fallback* mikrokontroler Arduino Nano dan ESP32-C3 pada studi kasus modul *quickshifter* menggunakan aplikasi berbasis Windows?
3. Bagaimana performa WebSocket ketika melakukan komunikasi *real-time* dengan aplikasi *quickshifter* berbasis Windows?

1.3 Tujuan Penelitian

Sesuai dengan rumusan masalah yang telah dituliskan, tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang sistem *quickshifter* dengan komunikasi WebSocket dan USB pada dual mikrokontroler Arduino Nano dan ESP32-C3
2. Melakukan implementasi komunikasi WebSocket dengan USB *fallback* mikrokontroler Arduino Nano dan ESP32-C3 pada studi kasus modul *quickshifter* menggunakan aplikasi berbasis Windows.
3. Melakukan analisis performa WebSocket pada komunikasi *real-time* dengan aplikasi *quickshifter* berbasis Windows.

1.4 Manfaat Penelitian

Dengan adanya penelitian ini, diharapkan dapat dimanfaatkan guna membantu masyarakat untuk melakukan pengembagan ilmu pengetahuan, antara lain:

1. Manfaat Akademis:
    a. Kontribusi pada penelitian IoT: Memberikan kontribusi pada penelitian seputar IoT yang memiliki fokus pada protokol komunikasi WebSocket dan skema komunikasi antar mikrokontroler. Penelitian ini dapat memberikan referensi untuk mengembangkan sistem komunikasi berbasis WebSocket pada kasus Dual-Mikrokontroler.
    b. Menambah wawasan dan literatur ilmiah: Menambah pemahaman dan wawasan seputar literatur ilmiah pada bidang IoT, khususnya komunikasi WebSocket mikrokontroler dengan aplikasi berbasis Windows.
2. Manfaat Praktis:
    a. Memberikan kemudahan pada praktik lapangan: Dengan adanya model komunikasi aplikasi secara nirkabel, dapat membantu proses penyetelan modul *quickshifter* pada berbagai skenario.
    b. Menyediakan aplikasi setting *quickshifter* Arduino: Memberikan aplikasi berbasis Windows yang dapat melakukan penyetelan modul *quickshifter* melalui protokol WebSocket dan USB.

1.5 Batasan Masalah

Untuk memastikan penelitian ini memiliki ruang lingkup yang jelas dan terfokus, ditentukan batasan masalah sebagai berikut:

1. Penelitian ini dilakukan pada modul *quickshifter* yang dirakit sendiri, yang dirancang untuk meniru fungsi *quickshifter* komersial dengan masukan dari RPM mesin dan sensor pemicu, namun dengan modifikasi tertentu untuk penelitian ini. Menggunakan mikrokontroler keluarga ATMega 328p dan keluarga ESP32.
2. Melakukan uji coba kelayakan menggunakan aplikasi buatan sendiri pada platform Windows dengan bahasa pemrograman C# dan aplikasi komunikasi Serial Monitor pada Arduino IDE.
3. Melakukan uji coba hanya pada sepeda motor kopling.