Output Communications Protocol

# Problem Description

Video game controllers are peripherals, which is a commonly used term to describe any device that can write output or receive input from a host machine, without being part of the host machines internal hardware. In our context, a host is any device that has its own CPU and is able to execute video games, provided with peripherals and displays for user interaction.

This working sheet investigates the communication protocol of peripherals. The goal is to establish what communication protocol would be optimal when designing a new type of peripheral. The context and motivation is developing a peripheral that combines the output streams of an arbitrary number of other peripherals, e.g., from video game controllers and smart trainers (STs). This new peripheral should ideally resemble the output of usual peripherals closely to be an actual alternative.

# Description of Protocols

The description of protocols provides an overview of MQTT, WebSocket (WS), ANT+, Bluetooth, and gRPC, highlighting their key features, applications, and advantages in various contexts. Each protocol's unique characteristics are explored, including their suitability for real-time communication, efficiency, power consumption, and interoperability.

## MQTT (Message Queuing Telemetry Transport):

MQTT is a lightweight massaging protocol, that is designed for minimal bandwidth network and battery power usage. This protocol transfers messages between devices. It’s particularly well-suited for connecting remote devices with a minimal network footprint, which makes it popular in the context of the internet of things (IoT).

* **Publish-subscribe-model:** MQTT operates on a topic-based publish-subscribe model. This model allows devices\clients to publish messages to a server (called broker), which then distributes the messages to other clients subscribed to those topics.
* **Last will and testament (LWT):** This protocol allows clients to specify an LWT message when connecting to an MQTT broker. If the connection is unexpectedly lost, this message will be sent to subscribers. It is useful in scenarios where monitoring device connectivity and status are crucial.
* **Retained messages:** MQTT brokers have the possibility to store a message for a specific topic and then deliver this message to new clients that subscribe to that topic later. This will keep the new subscribers updated with the most recent information available.
* **Quality of service (QoS):** MQTT supports three levels of QoS, to provide different guaranties on the delivery of messages based on the needs of various needs of applications.
  1. **QoS 0** (known as “at most once”)**:** In scenarios where speed is more crucial than precision, the QoS 0would be beneficial. It ensures that messages are delivered no more than once and delivery is not confirmed.
  2. **QoS 1** (known as “at least once”)**:** If message delivery is essential but duplicate messages are also acceptable, the QoS 1will be used, as it guarantees that the message is delivered at least once, but multiple delivering of messages can also happen.
  3. **QoS 2** (known as “exactly once”)**:** Ideal for situations where message accuracy and reliability are essential, as it provides the highest level of assurance by ensuring that each message is delivered to the client exactly once through a four-step handshake process.

[**https://mqtt.org/**](https://mqtt.org/)

## WebSocket(WS) :

This communication protocol provides full-duplex communication channels over a single TCP connection, meaning that both the client and the server can send and receive data at the same time over the same connection. WS can be run over the web facilitating real-time data transfer between a client and a server. WebSocket will suit well in web gaming, live sports update, and real-time trading platforms, where low latency communication is required.

* **Real-time communication**: WS enables continuous, interactive exchanges between a user's browser and the server, ensuring instant updates that are essential for applications requiring quick responsiveness, such as online gaming and live data feeds.
* **Low Latency:** WS connections are persistent, thereby there will be no need to establish a new connection for each request. This reduces latency and overhead, making WS ideal for applications requiring low-latency communication, such as real-time chat applications and multiplayer games.
* **Broad support**: Most of the contemporary web and game development frameworks provide easy integration for WS, simplifying its implementation across different environments.
* **Secure Communication:** WebSocket connections can be encrypted using Transport layer security (TLS), providing data confidentiality and integrity. This ensures that sensitive information exchanged between clients and servers remains secure.
* **Efficiency:** WS are designed to minimize the complexity of communication and also the amount of data transmitted over wire.

<https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/>

## ANT+ :

ANT+ is a wireless protocol, designed to facilitate the collection and transfer of sensor data for battery-operated devices. It operates on the ANT protocol, which is also a wireless sensor network technology. ANT+ standardizes the way data is transmitted over ANT networks, ensuring interoperability between different devices from different manufacturers. ANT+ sets precise guidelines for how different devices (ex. heart rate monitors, bike speed sensors, weight scales...) should communicate and what data should be transmitted.

* **Ultra-low power consumption:** ANT+ is beneficial in situations where minimal energy use is required. This is ideal for battery-operated devices like heart rate monitors, bike sensors and fitness trackers.
* **Robust communication:** ANT+ employs an adaptive isochronous transmission method, to ensure robust and reliable communication. This technology is beneficial in situations where there is a high risk of interference, like gyms or outdoor areas, where multiple devices might be operating in close proximity. The adaptive isochronous transmission method utilized by ANT+ ensures a steady and reliable connection by coordinating the timing of data transmissions.
* **Network flexibility:** Network flexibility in ANT+ allows to support various complex network topologies including star, tree and mesh configurations. This allows a master device to communicate with multiple slave devices and even support multiple master devices in one network. This capability is particularly beneficial in environments where different devices need to interact seamlessly and efficiently, such as in fitness centers where multiple sensors and monitoring devices operate simultaneously.
* **Interoperability:** Any devices supporting ANT+ (such as heart rate monitors, bike speed sensors, or cadence meters), can communicate with other ANT+ devices regardless of the manufacturer. The standardized communication protocols in ANT+ makes it happen to facilitate a seamless and unified ecosystem where different devices can work together smoothly, providing a consistent and reliable user experience in monitoring and tracking health and fitness metrics.

## Bluetooth:

Bluetooth is a wireless technology standard designed to exchange data over short distances using short-wavelength UHF radio waves in the ISM band from 2.400 to 2.485 GHz. It is commonly used in personal area networks (PANs) and has become ubiquitous in consumer electronics such as smartphones, laptops, headphones, and other devices.

* **Range and power classes:** Bluetooth devices are categorized into three power classes to suit different applications based on range and power needs.
  + **Class 1 devices** offer the longest range up to 100 meters and are ideal for large spaces but consume more power.
  + **Class 2 devices**, the most common in consumer electronics, provide a moderate range of about 10 meters, balancing range with power efficiency.
  + **Class 3 devices** have the shortest range of approximately 1 meter and consume the least power, suitable for devices that operate in close proximity like computer peripherals. These classifications help optimize Bluetooth device performance for specific usage scenarios.
* **Bluetooth low energy (LE):** Bluetooth low energy, introduced with Bluetooth 4.0, is a significant advancement tailored for devices requiring minimal power consumption. This version of Bluetooth is particularly well-suited for applications that do not need constant data streaming but rather periodic updates, such as in wearable technologies and fitness trackers. Bluetooth LE is highly efficient, extending battery life significantly, which is crucial for small, portable devices that users wear continuously. The technology enables these devices to communicate small amounts of data intermittently while maintaining long battery life, making Bluetooth LE ideal for a wide range of health and fitness applications, smart home devices, and more.
* **Security features:** Bluetooth technology integrates various security features to protect data transmitted between devices. Key security measures include authentication and encryption, which help prevent unauthorized access and ensure the privacy and integrity of the data exchanged. Over the years, with each new version of Bluetooth, these security protocols have been enhanced to address emerging vulnerabilities and threats. This continuous improvement helps maintain the trustworthiness of Bluetooth connections, making them secure for a wide range of applications, from personal device connectivity to confidential business communications.
* **Versions and compatibility:** Bluetooth technology has continuously evolved, with each new version enhancing key aspects such as speed, range, and energy efficiency. **Bluetooth 4.0** introduced **Bluetooth Low Energy** for devices requiring low power consumption. Following that, **Bluetooth 5.0** improved upon the data transmission rates and increased the operational range significantly. The most recent, **Bluetooth 5.2**, has further optimized these capabilities, ensuring even faster and more reliable data transmission over the air. These advancements make Bluetooth increasingly suitable for a broader array of applications, from simple personal device connectivity to complex industrial and IoT implementations.
* **Profiles and protocols:** Bluetooth employs a series of profiles to specify how the technology is utilized in various applications. These profiles define specific sets of rules and protocols governing the communication between devices for particular functionalities. For example, the “advanced audio distribution profile” (A2DP) ensures high-quality audio streaming between Bluetooth-enabled devices like headphones and smartphones. The hands-free profile (HFP) facilitates communication between devices such as car kits and mobile phones, enabling hands-free calling functionality. Similarly, the” human interface device” (HID) profile is used for devices like keyboards and mice to enable seamless interaction with computers and other compatible devices. By standardizing these profiles, Bluetooth ensures interoperability and smooth communication across a wide range of devices and applications.

## gRPC

gRPC stands as a high-performance Remote Procedure Call (RPC) framework, initially crafted by Google. Its primary function lies in facilitating seamless communication between distributed systems. It achieves this by enabling clients to invoke methods or procedures on remote servers in a manner that resembles local function calls. This framework streamlines the interaction between different parts of an application, simplifying the development of distributed systems and encouraging efficiency in data exchange across networked components.

* **Protocol buffers :** gRPC utilizes protocol buffers as its interface definition language (IDL) for defining service interfaces and message structures. Protocol buffers provide an efficient, language-neutral, and extensible serialization format, which makes it ideal for exchanging data and ensures compatibility between different programming languages and platforms. This enables seamless communication between distributed systems, enhancing interoperability and performance in networked applications.
* **HTTP/2-based communication:** gRPC leverages the HTTP/2 protocol as its underlying transport mechanism, which offers significant improvements over traditional HTTP/1.x. HTTP/2 introduces features (such as multiplexing, header compression, and server push), enhancing performance, reducing latency, and maintaining network resources. These optimizations make gRPC a good choice for high-performance, low-latency communication between distributed systems, ensuring efficiency in data exchange and improved overall system responsiveness.
* **Bidirectional streaming:** Bidirectional streaming in gRPC allows simultaneous communication between clients and servers, enabling real-time data exchange for applications like chat or streaming media services. This feature supports interactive, continuous communication, enhancing the user experience in real-time applications.
* **Strong typing and code generation:** gRPC generates client and server code in various languages from the protocol buffers service definitions. This approach provides strong typing, reduces errors and enhances developer productivity by eliminating the need to write standardized code for communication.
* **Cross-platform support:** Cross-platform support in gRPC, ensures that the framework can be used with various programming languages (such as C++, Java, Python, Go, and JavaScript). This adaptability allows developers to build distributed systems across different environments and platforms without being limited to a specific programming language.
* **Interoperability:** While gRPC is designed to be highly efficient for internal microservices communication within environments like Kubernetes, it also offers interoperability with other systems through features like HTTP/1.x compatibility and the ability to generate code for RESTful APIs from protocol buffers definitions.

<https://grpc.io/docs/what-is-grpc/>

# Description of Hosts Input Requirements

A common communication protocol between peripherals and hosts is the USB Human Interface Device (HID) standard. When used via Bluetooth, the protocol is often auxiliated by the Generic Attribute Profile (GATT), which follows a similar structure as HID.

The HID standard can be used for all kinds of peripherals, such as mouse and keyboard, or gaming controllers. Conceptually, the HID standard works by sending a *report description* immediately after connecting to the host. The report description defines the peripherals output format. The data stream of a peripheral contains *reports*, which are packages of data that adhere to the format defined in the report description, which was exchanged at the initial connection to the host. These reports are an array of bits, where the resource description describes their semantics, including index of elements, elements’ size, and data type. Additionally, the HID standard utilizes *usage IDs*, which are unique identifiers that identify a specific button, key, or control of a peripheral. To prevent conflicts and overlaps in usage IDs and to organize them in a logical manner, HID defines *usage pages*. A usage page is a collection of usage IDs for one specific domain of input, e.g., a keyboard. To give an example, the usage page for keyboards specifies one usage ID for each button on a common keyboard with 104 buttons [https://usb.org/sites/default/files/hut1\_4.pdf, p 89]. This usage page can be used by any designer of a keyboard to ensure compatibility with hosts implementing the HID standard.

Because any peripheral can describe their own report description, HID is a highly flexible framework that leaves the responsibility of defining its output format to the peripheral themselves.

<https://usb.org/sites/default/files/hut1_4.pdf>

In summary, a solution that combines input streams from multiple peripherals, should be able to produce HID, if it connects to a host via USB, while it should be able to produce GATT, if it connects to a host via Bluetooth. These communication standards ensure compatibility with most contemporary host machines.

# Conclusion

Even though we have found that USB and Bluetooth are the usual connection protocols between peripherals and hosts, we chose Websocket to communicate with our receiving mock implementation for the sake of a proof of concept. It will be easiest for us to setup a demonstration of our solution using this protocol.

We recommend future endeavors that focus on user experience to implement USB and Bluetooth technology to be able to connect to actual host machines.