**2. Data Transmission Method Evaluation and Design Decision**

*For Camera Motion Tracking and Real-Time Scene Synchronization*

**2.1 Introduction**

This section evaluates three communication interfaces—USB, Ethernet, and Wi-Fi—for transmitting fused IMU and IR tracking data from the microcontroller to external systems. The system’s objective is to track camera motion in real time and send this information to rendering engines or media servers to dynamically adjust the scene displayed on LED walls or chroma key backgrounds in virtual production environments.

**2.2 Communication Options Overview**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **USB** | **Ethernet** | **Wi-Fi (ESP32)** |
| Interface Type | USB CDC (Virtual COM) | UDP over Ethernet | UDP over Wi-Fi (802.11n) |
| Topology | Point-to-point | Broadcast / Multicast capable | Broadcast / Multicast capable |
| Typical Data Rate | ~1 Mbps | 10/100 Mbps | ~2–5 Mbps practical |
| Latency | Low | Very low (≤1 ms typical) | Moderate (variable, 10–30 ms) |
| Power Consumption | Low | Moderate | Higher (due to radio activity) |
| Driver Requirements | Built-in OS drivers | No drivers; uses sockets | No drivers; uses sockets |
| Integration Target | Debugging, PC logging | Game engines, studio networks | Mobile or wearable systems |
| Hardware Complexity | Low | Medium (PHY + RJ45) | Low (ESP32 SoC) |
| EMI Susceptibility | Medium | Low (shielded, wired) | High (wireless interference) |
| Reliability | Good | Excellent | Variable |

**2.3 System Requirements**

The communication interface must:

* Stream fused motion data at 20–100 Hz
* Maintain low latency and jitter to ensure scene stability
* Integrate into standard studio or stage production networks
* Avoid reliance on proprietary or platform-specific drivers
* Be robust against environmental noise and suitable for live use

**2.4 Design Decision**

After evaluating the options, **Ethernet** is selected as the primary communication interface for the following reasons:

* Offers low-latency and high-bandwidth data transmission
* Supports open protocols such as UDP and OSC used in game engines (e.g., Unreal) and real-time media systems
* Provides reliable and deterministic performance, critical for accurate scene rendering
* Integrates well into professional audio-visual infrastructure

**USB** remains useful for debugging during development. **Wi-Fi** may be revisited in future mobile or battery-operated configurations, but is not suited for high-reliability studio environments at this stage.

**2.5 Next Steps**

* STM32 microcontroller with built-in Ethernet MAC (e.g., STM32F407, F429, H743)
* Elect an Ethernet PHY (e.g., LAN8720 or DP83848) via RMII
* Adding RJ45 Ethernet connector with integrated magnetics
* Considering UDP or OSC protocol for sending fused motion data
* Consider time synchronization if multiple trackers or cameras are involved

**3. Microcontroller Selection and System Architecture Evaluation**

**3.1 Hardware Interfaces and Peripheral Requirements**

**Communication Interfaces**

|  |  |  |
| --- | --- | --- |
| **Interface** | **Purpose** | **Notes** |
| I2C or SPI | Communication with IMU (e.g., MPU9250) | SPI preferred for better performance |
| UART | Debugging or serial logging | Optional |
| SWD/JTAG | Firmware flashing and debugging | Required (what’s happening during runtime) |
| Ethernet (MAC+PHY) | OSC/UDP communication | Requires external PHY if STM32 lacks internal |
| USB (Power) | 5V input source (up to 3A) | Not used for communication; protected power entry |

**GPIO and Control**

|  |  |  |
| --- | --- | --- |
| **Usage** | **Type** | **Notes** |
| IR LEDs | GPIO or PWM | Multiple outputs with optional driver control |
| Status LEDs | Digital Output | Indicate state or errors |
| User/Reset Buttons | Digital Input | Bootloader or runtime control |
| Sync Input (optional) | Digital Input | For timing synchronization from camera |

**Timer, PWM, and ADC/DAC Requirements**

|  |  |  |
| --- | --- | --- |
| **Function** | **Peripheral** | **Notes** |
| LED modulation | PWM/Timers | Optional for visibility enhancement |
| Timestamping | Timers | Required for aligning sensor data |
| ADC | Not required | Unless battery monitoring is added |
| DAC | Not used | All signals are digital |

**GPIO Utilization Estimate**

|  |  |
| --- | --- |
| **Function Category** | **Estimated GPIOs** |
| IMU communication | 3–5 |
| Ethernet PHY | ~10–12 |
| IR LEDs control | 4–6 |
| Status and control lines | 2–4 |
| Debug and comm lines | 3–4 |
| Optional sync input | 1–2 |
| **Estimated total** | **20–30 GPIOs** |

**3.2 Software Processing and Performance Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Frequency** | **Execution Time (Est.)** | **Purpose** |
| IMU data polling/filtering | 200–500 Hz | < 100 µs | Raw sensor data acquisition |
| Sensor fusion (Kalman/VQF) | 100–200 Hz | ~0.5–1 ms | Attitude and motion estimation |
| OSC/UDP transmission | 20–100 Hz | ~0.1–0.3 ms | External communication |
| IR LED toggling | 100–500 Hz | < 100 µs | Visual tracking support |
| Debug UART logging | As needed | Background | Serial output *(optional)* |
| External sync processing | Edge triggered | Instantaneous | Time alignment |

**CPU Load Estimate**: With most tasks requiring < 1 ms execution and running at ≤ 500 Hz, the overall CPU load is comfortably within 40–50% of an 84 MHz ARM Cortex-M4.

**3.3 Memory Estimate**

|  |  |  |
| --- | --- | --- |
| **Module/Function** | **RAM Estimate** | **Flash Estimate** |
| IMU Driver | ~2 KB | ~8 KB |
| Kalman/VQF Fusion | ~4–8 KB | ~10–25 KB |
| OSC/UDP Stack | ~4 KB | ~10–20 KB |
| Runtime Buffers & Control | ~8–10 KB | ~10 KB |
| Logging/Debug (optional) | ~1–2 KB | ~5 KB |
| **Total Estimated** | ~30–40 KB | ~60–100 KB |
| **STM32F401 Capacity** | 64 KB RAM | 256 KB Flash |

There is adequate headroom for updates, debugging, and feature expansion.

**3.4 Architectural Suitability**

|  |  |
| --- | --- |
| **Requirement** | **Evaluation** |
| Processing Core | 32-bit ARM Cortex-M4 (STM32F401) |
| GPIO Availability | Sufficient for projected needs |
| Communication Support | Ethernet, USB, UART, I2C, SPI |
| Real-Time Performance | Sufficient margin for deterministic timing |
| Flash and RAM Capacity | Adequate for current and future use |
| External Memory Need | Not required unless for advanced features like OTA or logging |

The STM32F401 MCU is an appropriate and balanced choice for the design given the computation profile, memory demand, and I/O requirements.

**3.5 Ethernet Interface Decision**

Enable reliable, real-time data transmission (e.g., OSC over UDP) from the microcontroller to an external system (e.g., Unreal Engine or virtual production setup) with minimal latency and complexity.

**Options Considered**

|  |  |  |  |
| --- | --- | --- | --- |
| **Option** | **Description** | **Pros** | **Cons** |
| **W5500 Ethernet Module** (External MAC + PHY via SPI) | A standalone hardware TCP/IP controller (WIZnet W5500) communicating over SPI | - Low cost and readily available - Simplified firmware (no need for TCP/IP stack) - Small footprint - Compatible with STM32F401 | - Slightly limited throughput (sufficient for OSC/UDP) - Less flexible for complex protocols |
| **Native MAC + PHY** (e.g., STM32F407 + LAN8720) | Microcontroller with built-in Ethernet MAC and external PHY (RMII interface) | - Full control over TCP/IP stack (LwIP) - Higher flexibility - Higher throughput | - More expensive MCU - Complex PCB layout and clocking - Increased software complexity and debug time |

**Design Decision**

After evaluating cost, complexity, and project-specific requirements, the **W5500 module** was selected.

* **Reasons:**
  + STM32F401 lacks native MAC → W5500 fills this gap without needing MCU upgrade.
  + OSC/UDP communication needs minimal bandwidth → W5500 performance is sufficient.
  + Short development cycle → hardware-managed networking reduces firmware complexity.
  + Clean SPI interface → easier to route and integrate into a compact board.

**Hardware & Software Integration**

* **W5500 Module (3.3V)** connected via **SPI** to STM32F401
* Requires **RJ45 jack with integrated magnetics** or breakout module
* **Power supply**: 3.3V regulated, ~150mA peak
* Lightweight W5500 driver via SPI
* Use of **UDP sockets** for OSC packet transmission
* No RTOS or LwIP needed → minimal memory footprint

For the purpose of tracking motion and transmitting data to a studio system with low latency and low complexity, the **W5500-based Ethernet solution** offers the best balance between **cost**, **simplicity**, and **performance** within the constraints of the STM32F401 architecture.