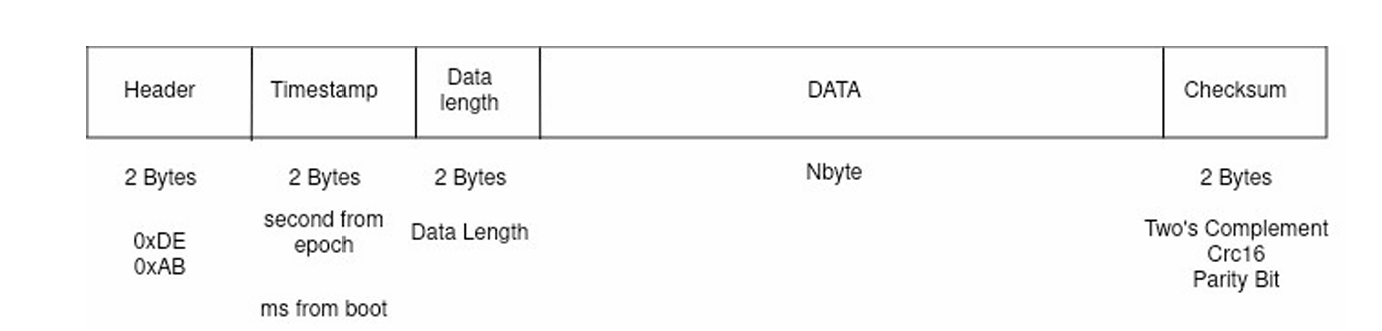
**Tài liệu thiết kế firmware**

**1 Tại sao lại sử dụng kiểu dữ liệu nhị phân thay vì mã ASCII**

* Dữ liệu nhị phân thường sẽ nhỏ gọn hơn so với mã ASCII.
* Do dữ liệu được truyền trực tiếp dưới dạng byte thô, quá trình truyền và xử lý sẽ nhanh hơn đáng kể so với việc truyền các ký tự ASCII.
* Với dữ liệu nhị phân, bạn có thể dễ dàng tích hợp các thuật toán kiểm tra độ chính xác (checksum, CRC) vào mỗi gói tin một cách hiệu quả, giúp phát hiện và sửa lỗi nhanh chóng.

**2 Khung truyền dữ liệu**

****

**Bao gồm:**

* Header để xác định điểm bắt đầu gói tin.
* Thời gian gửi gói tin (Timestamp).
* Độ dài dữ liệu (Data Length).
* Dữ liệu (Type packed + DATA).
* Checksum để kiểm tra tính toàn vẹn dữ liệu.

**3 Tài liệu diễn giải**

**3.1 Layer software**

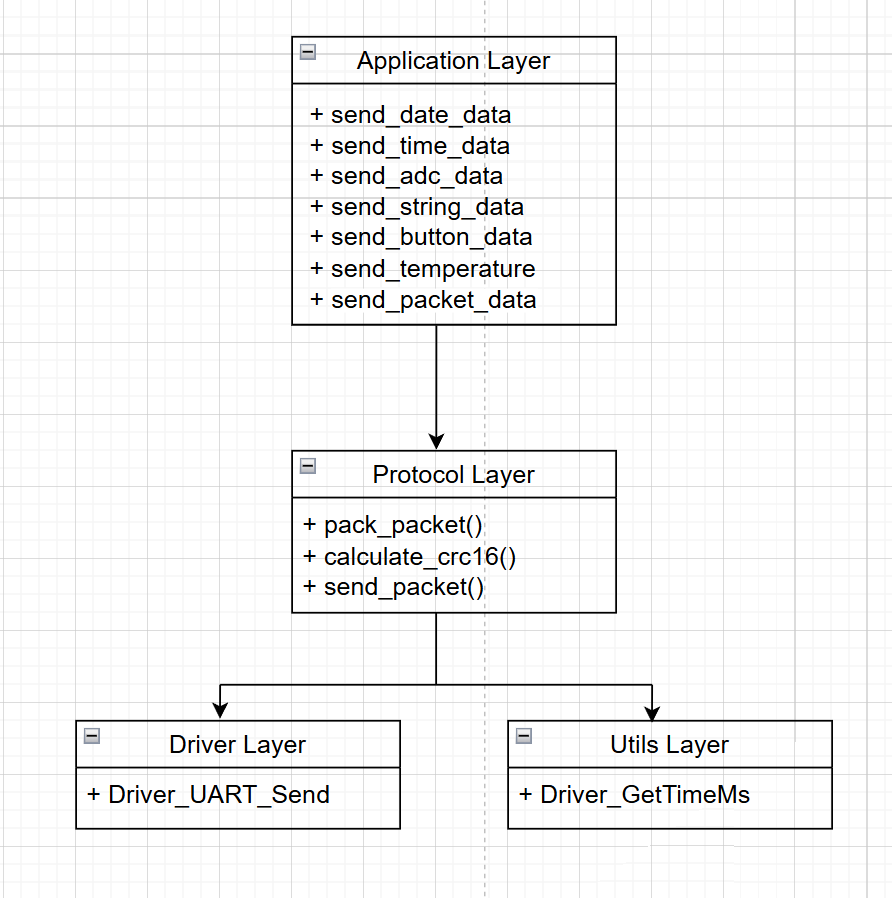
Application

Protocol

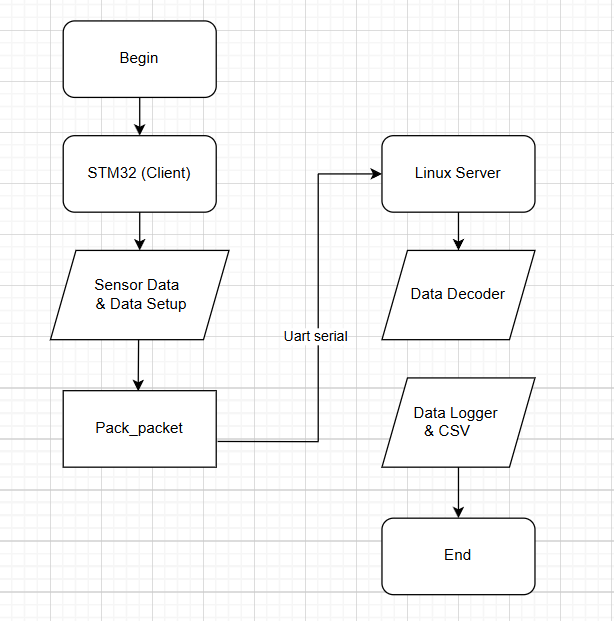
Utils

Driver

* **Application Layer** (Tầng ứng dụng): Tầng dùng để gửi các message.
* **Protocol Layer** (Tầng giao thức): Tầng dùng để đóng gói các message.
* **Driver/Utils Layer** (Tầng trình điều khiển và tiện ích): Tầng dùng để giao tiếp trực tiếp với các API.



**3.2 Sơ đồ khối**

****

**Mô tả chi tiết các khối:**

1. **STM32 (Client):**

* **Sensor Data Acquisition & Data Setup:**  
  Thu thập dữ liệu từ các cảm biến hoặc nguồn dữ liệu nội bộ. Dữ liệu được định dạng (đóng gói) theo các kiểu như Date, Time, ADC, String, Button, Temperature.
* **Application Layer:**
* Chứa các hàm như send\_date\_data(), send\_time\_data(), send\_adc\_data(), send\_string\_data(), send\_button\_data(), send\_temperature().

1. **Protocol Layer:**

* Đóng gói dữ liệu từ tầng Application thành packet nhị phân theo giao thức định nghĩa (bao gồm header, timestamp, payload\_size, payload, checksum).
* Tính CRC (calculate\_crc16) và gọi send\_packet() để chuyển packet qua giao tiếp.

1. **Driver Layer:**

* Gửi packet qua UART bằng hàm hoặc Driver\_UART\_Send.
* Lớp này không cần quan tâm đến logic đóng gói, chỉ thực hiện truyền dữ liệu.

1. **Linux Server (Host):**

* **UART Receiver:**  
  Nhận dữ liệu từ cổng UART và nhận các packet nhị phân từ STM32.
* **Protocol/Data Decoder:**  
  Giải mã các packet nhận được: kiểm tra header, giải mã timestamp, payload và xác nhận CRC.  
  Nếu CRC không khớp, server sẽ ghi log lỗi và loại bỏ packet (để tránh ghi nhận dữ liệu bị hỏng do mất kết nối TX/RX).
* **Data Logger & CSV Writer:**
  + In ra khoảng thời gian (Received Data Interval) của các gói tin nhận được.
  + Ghi dữ liệu được giải mã vào các file CSV riêng biệt dựa trên loại dữ liệu (ví dụ: date.csv, time.csv, adc.csv, …).Lưu log các sự kiện và thông báo lỗi theo một định dạng tự do.

4