**Decoding the Ankle Exo’s Hex Data Format**

**Motivation**

The ankle exoskeleton is currently transmitting data in bytes. Our wireless transmission capabilities allow for 10 kilobytes per second transmission—this means we get about 100 bytes per 100 Hz loop. Reporting all the given data takes up 80 bytes when compressed into binary. There is significant packet loss without binary.

**Getting Started**

1. Check out the file “example\_output.txt”

This file contains a complete example hex output from the Axo. I suggest using a hex -> ASCII online tool to translate it, just to get an idea of how it works. For convenience, here is a screenshot of the ASCII translation:

Text

Description automatically generated

What’s going on here? The data transmits in bytes, which are stored in the log file as hex (each hex character is 4 bits, or half a byte). Only the sensor data (numbers) are actually encoded in binary. The rest of the output is in ASCII binary encoding. That is what is being displayed.

The first couple lines prints out the key, which explains what

As before, the data starts by transmitting a set of headers; this is encoded in ASCII and can be directly translated. **This set of headers should be the first couple lines in the output CSV** (each on its own line).

Next comes the actual data. This will always start with an ASCII message of “LOG,TIME | Motors started.” Boring. Just write that line to the CSV. Note: all LOG messages will be in ASCII format and have no specified length.

**Note: unless specified as unsigned, assume all values are signed.** For a 16-bit (2 byte) number, check if it is greater than (2^16) / 2 and subtract 2^16 from it if so.

.\_\_\_.

/     \

| O \_ O |

/  \\_/  \

.' /     \ `.

/ \_|       |\_ \

(\_/ |       | \\_)

\       /

\_\_\\_>-<\_/\_\_

~;/     \;~

See below for information on indicators…

# Indicators

|  |  |  |  |
| --- | --- | --- | --- |
| **Message Type** | **Message Indication (**🡪**ASCII bytes**🡨**)** | **Expected Bytes (after indicator)** | **Byte Translation Key** |
| IMU | 🡪\nI 🡨 | 36 | Byte num: comment  0: uint8 0-3  1: uint8 0-3  2: uint8 0-3  3: uint8 0-3  4: uint8 0-3  5: uint8 0-3  6: uint8 0-3  7: uint8 0-3  8-9: Float \* 10^4  10-11: Float \* 10^4  12-13: Float \* 10^4  14-15: Float \* 10^4  16-17: Float \* 10^4  18-19: Float \* 10^4  20-21: Float \* 10^4  22-23: Float \* 10^4  24-25: Float \* 10^2  26-27: Float \* 10^2  28-29: Float \* 10^2  30-31: Float \* 10^2  32-33: Float \* 10^2  34-35: Float \* 10^2 |
| Motor | 🡪\nM 🡨 | 20 | Byte num: comment  0-3: uint32  4-5: uint16  6-7: uint16  8-9: int16  10-11: int16  12-13: uint16  14-15: uint16  16-17: uint16  18-19: uint16 |
| FSR | 🡪\nF 🡨 | 6 | Byte num: comment  0-3: uint32  4-5: uint16 |
| Load | 🡪\nL 🡨 | 6 | Byte num: comment  0-3: uint32  4-5: uint16 |
| Log | 🡪\nLOG,🡨 | Undefined | ASCII |