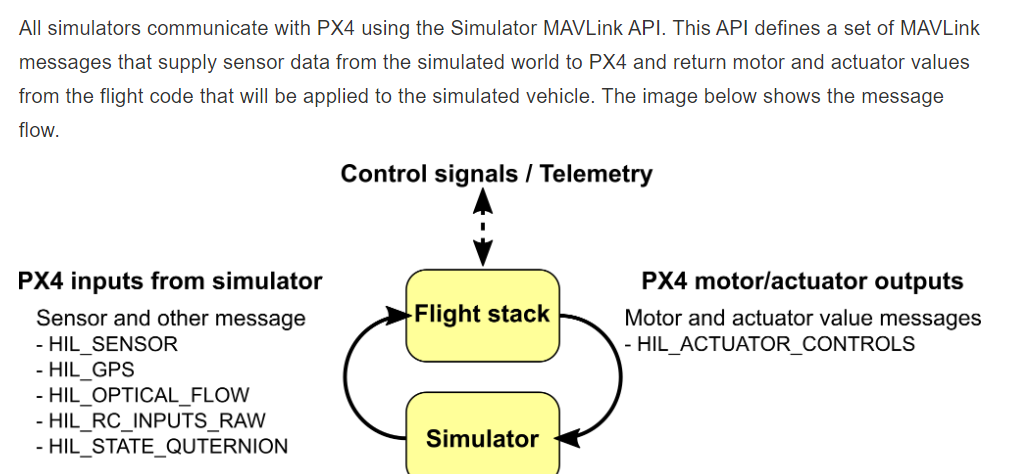
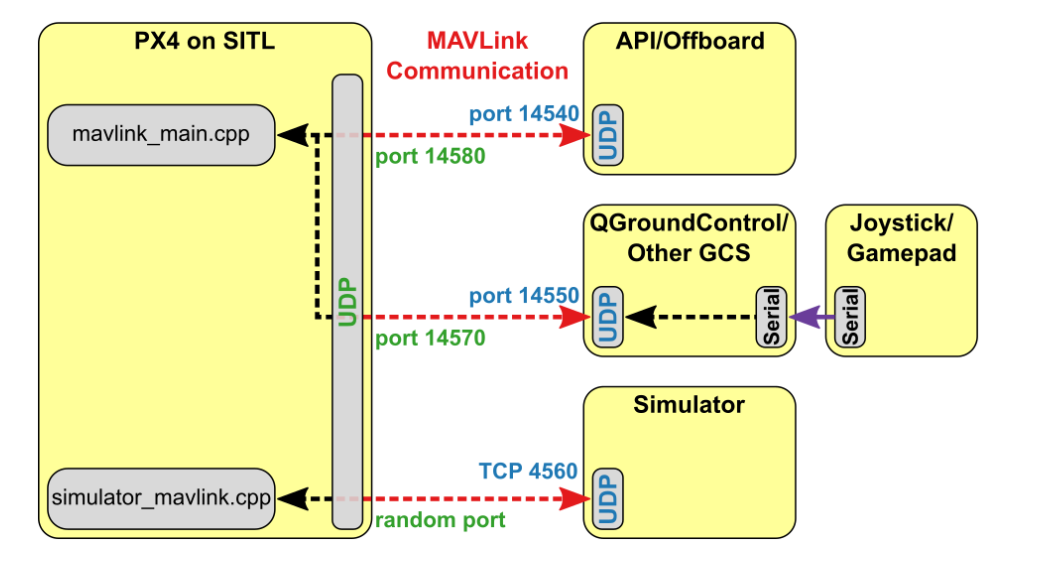
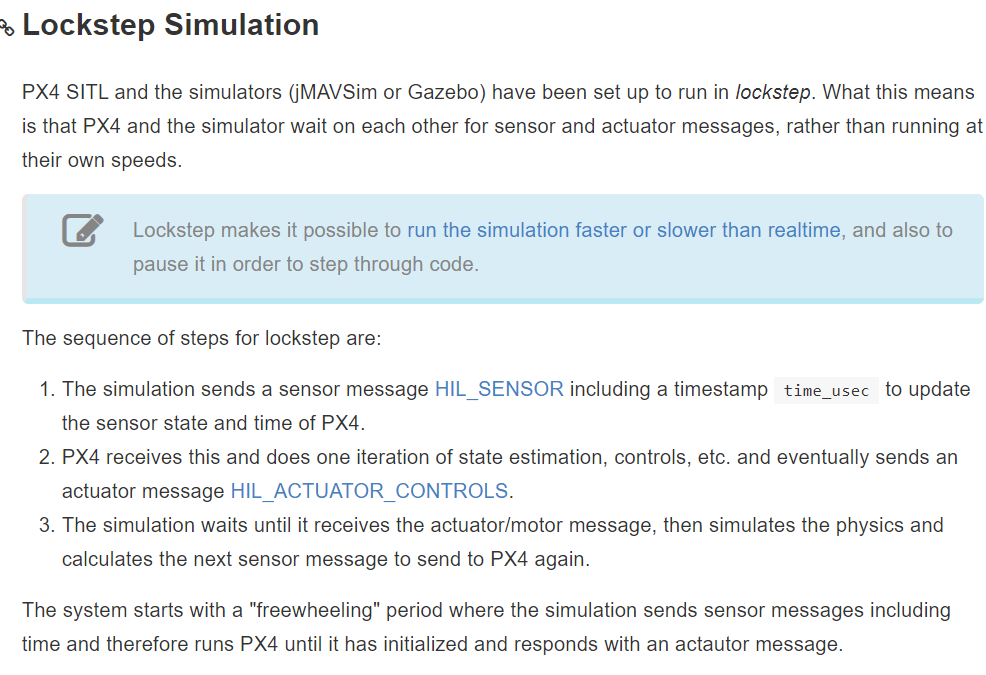
Este es el análisis y lo que he hecho para intentar crear un puente en Python para comunicar PX4 con FG.

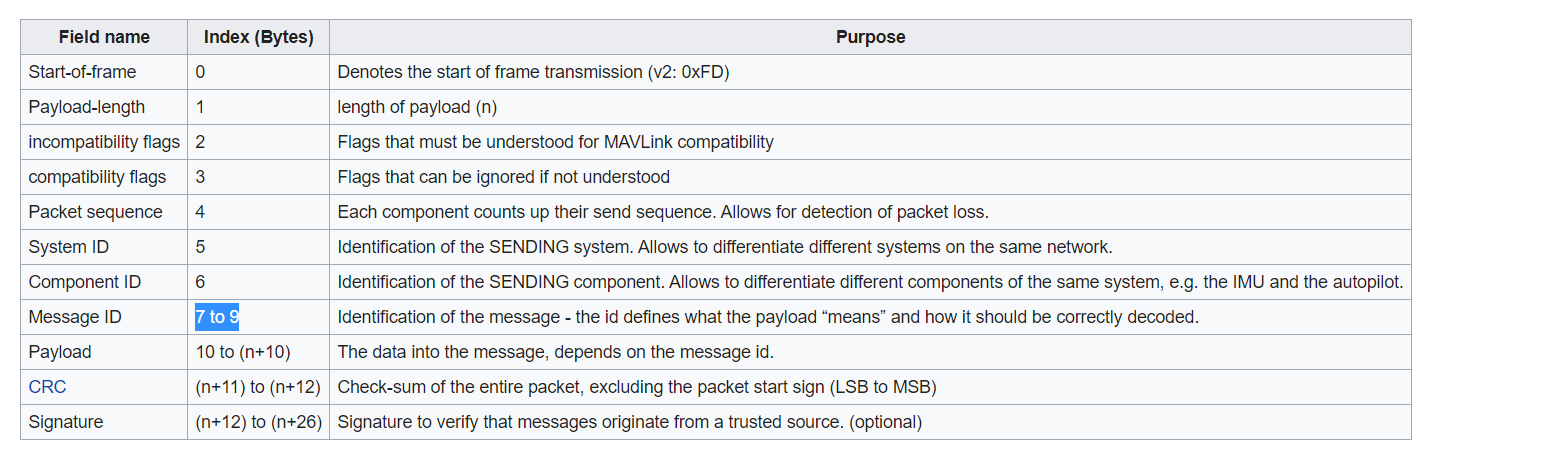
Consideraciones previas:

* Los mensajes intercambiados son los siguientes 
* La comunicación debe ser en el puerto 4560 :

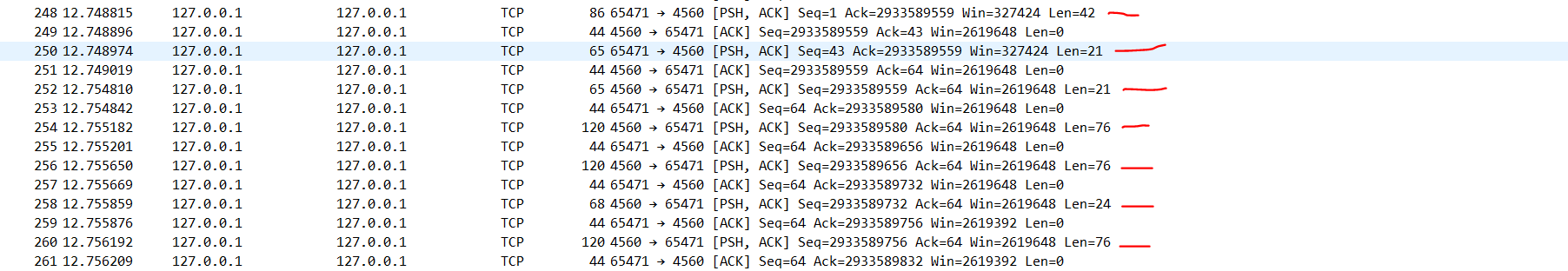


* La simulación tiene el lockstep activado:



* Estructura de mavlink 2: 

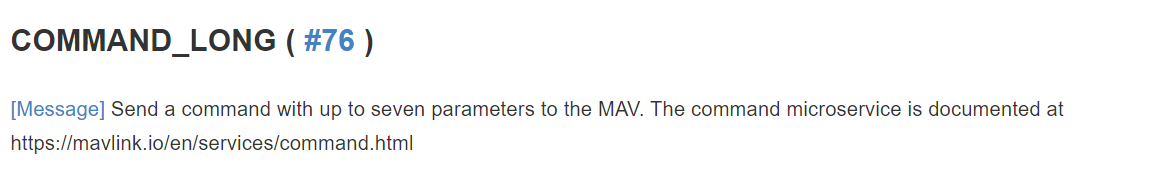
1. En primer lugar, he compilado SITL con el simulador jmavsim y he analizado la comunicación entre jmavsim y PX4 con wireshark desde los primeros mensajes hasta la recepción de jmavsim de los actuadores.



El puerto de PX4 es aleatorio y en esta inicialización se ha definido en el 65471. Comunica 2 primeros mensajes con jmavsim antes de que jmavsim envíe sus mensajes.

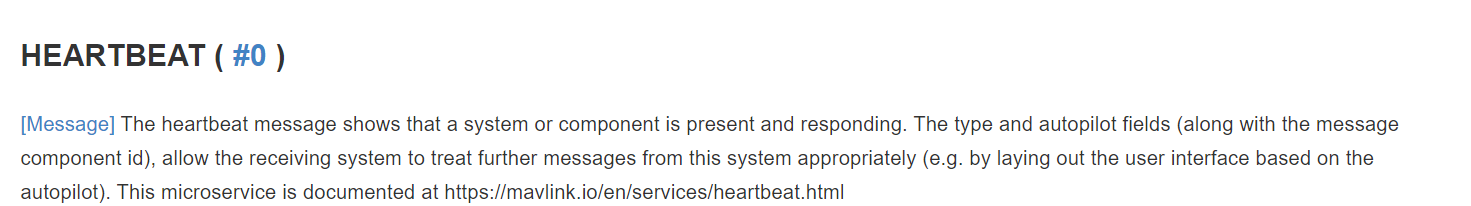
1. fd1e00000001014c00000000e64200409c450000000000000000000000000000000000000000ff011904

fd 1e 00 00 00 01 01 **4c0000** Estos primeros bytes son la cabecera del mensaje, lo importante son los últimos 3 bytes en negrita que son el ID del mensaje. La codificación que usa mavlink es la del byte menos significativo por lo que debe leerse al revés **00004c** que en decimal es **76** que es el mensaje de **command\_long**



1. fd09000001010100000000000000000c0000039158

fd 09 00 00 01 01 01 **000000** este id corresponde al mensaje **HEARTBEAT**

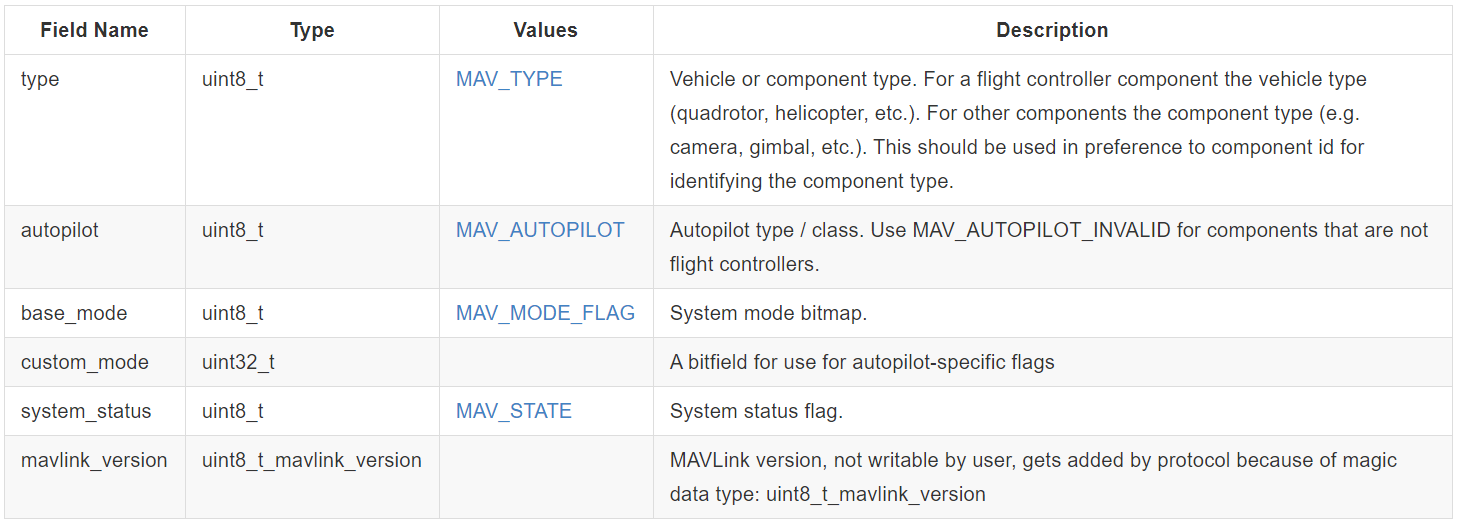


Esto será lo primero que deberá leer el programa antes de lanzar ningún mensaje.

Ahora los mensajes que debe replicar en el envío el programa.

1. fd0900000001330000000000000000000000030d2f

fd 09 00 00 00 01 33 **000000** Esto indica que el mensaje es un HEARTBEAT, en este caso si que es importante conocer el resto de parámetros a enviar.

Esta es la estructura del mensaje según la documentación de PX4: 

Los int8 ocupan 1 byte y el int32 ocupa 4 bytes. 00 00 00 00000000 00 03 0d2f

Type: 00

Autopilot:00

Base\_mode:00

Custom\_mode: 00000000

System\_status:00

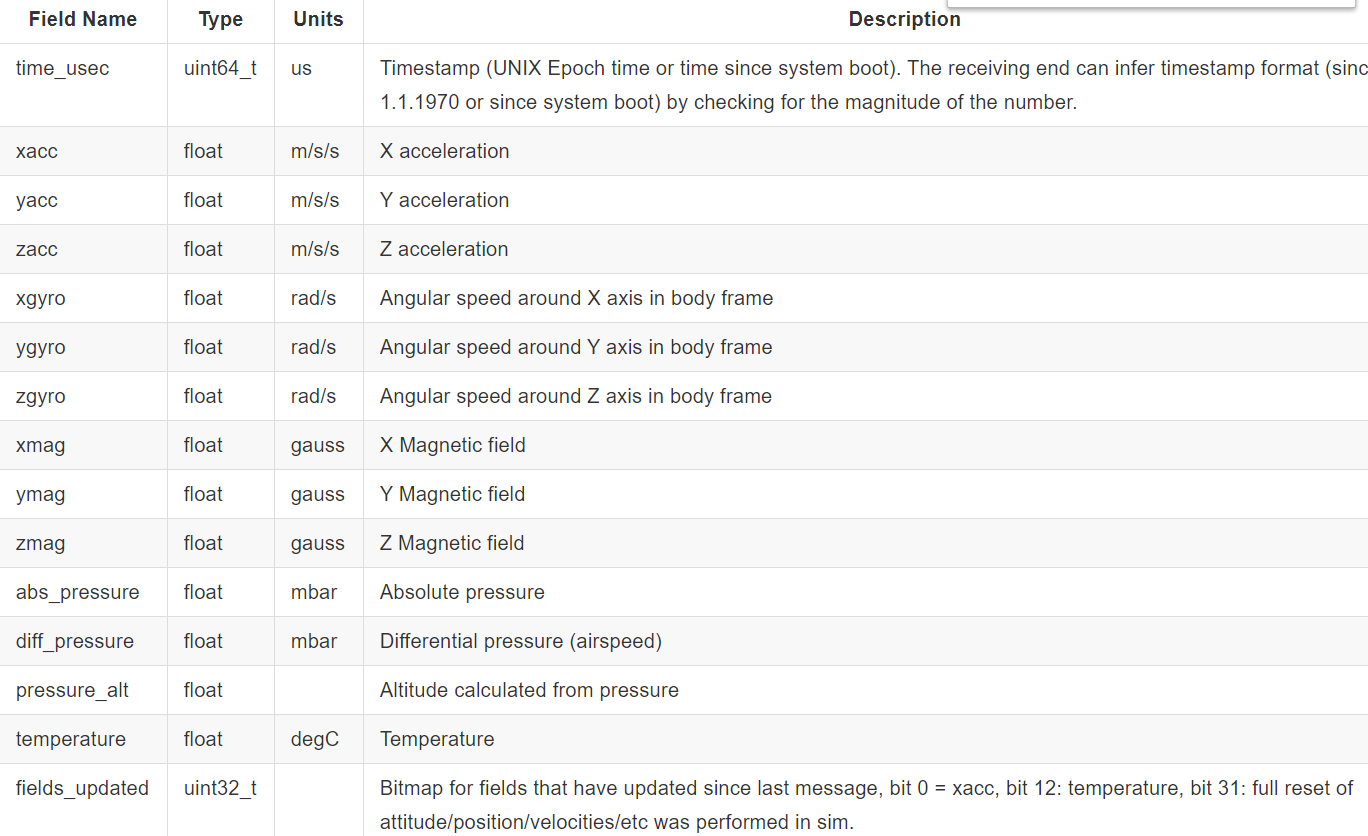
Mavlink\_version:03

Los últimos dos bytes son del protocolo de comunicación, no del mensaje en si mismo.

1. fd4000000101336b000008a31ce3dfa10500af202c3dc64a1a3dd2751dc15606e3bbf2d5053bc2ef21b93ad9583eae8c3e3c1553e03e6aff6e440000000001e0f34300000000ff1b0000d858

fd 40 00 00 01 01 33 **6b0000** el id del mensaje es 00006b 🡪 107 que es el mensaje de HIL\_SENSOR.

Esta es la estructura del mensaje:



08a31ce3dfa10500 af202c3d c64a1a3d d2751dc1 5606e3bb f2d5053b c2ef21b9 3ad9583e ae8c3e3c 1553e03e 6aff6e44 00000000 01e0f343 00000000 ff1b0000 d858

Time\_usec: 08 a3 1c e3 df a1 05 00 🡪 1585357843637000

Xac: af 20 2c 3d🡪 0.042023357

Yac: c6 4a 1a 3d🡪 0.0376689658

Zac: d2 75 1d c1🡪 - 9.841265

Xgyro: 56 06 e3 bb🡪 - 0.00692824554

Ygyro: f2 d5 05 3b🡪 0.002042171

Zgyro: c2 ef 21 b9🡪 - 0.000154434732

Xmag: 3a d9 58 3e🡪 0.211766154

Ymag: ae 8c 3e 3c🡪 0.01163022

Zmag: 15 53 e0 3e🡪 0.438133866

Abs\_pressure: 6a ff 6e 44🡪 955.990845

Diff\_pressure: 00 00 00 00🡪0

Pressure\_alt: 01 e0 f3 43🡪 487.750031

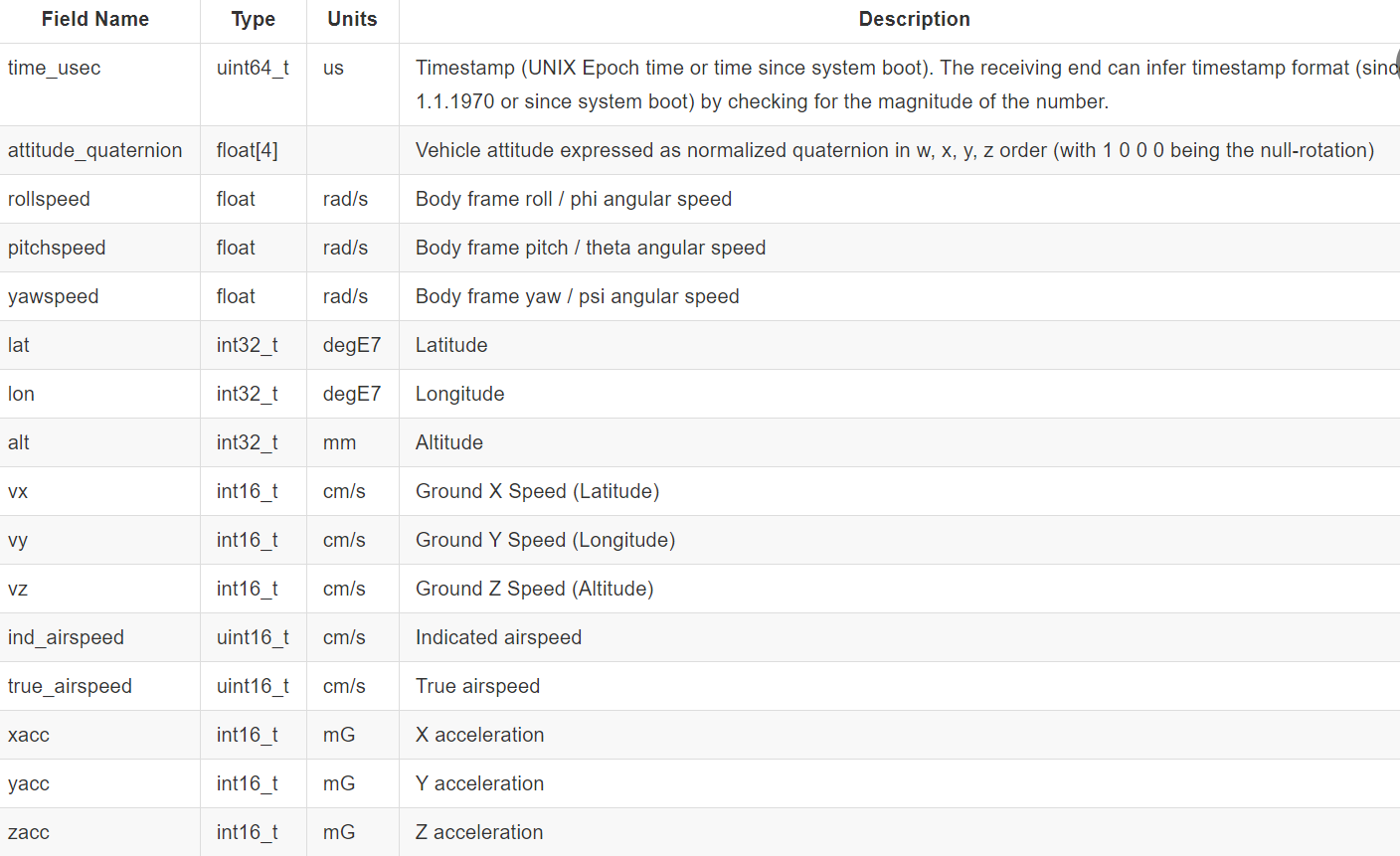
Temperature: 00 00 00 00🡪 0

Fields\_update: ff 1b 00 00🡪 7167

1. fd40000002013373000008a31ce3dfa105000000803f0000000000000000000000000000000000000000000000004c52401c44f4170500000000000000000000000000000000000000005074

fd 40 00 00 02 01 33 **7300000** el id del mensaje es 000073 🡪 115 que es el mensaje de HIL\_STATE\_QUATERNION.

Replicando estos mensajes en Python se consigue una correcta comunicación.



08a31ce3dfa10500 0000803f 00000000 00000000 00000000 00000000 00000000 00000000 4c52401c 44f41705 00000000 0000 0000 0000 0000 0000 0000 0000 0000 5074

Time\_usec: 08 a3 1c e3 df a1 05 00 🡪 1585357843637000 (Se envían en el mismo momento de la simulación)

Attitude\_quaternion:[ 0000803f 00000000 00000000 00000000]🡪[1 0 0 0]

Rollspeed:00 00 00 00 🡪0

Pitchspeed: 00 00 00 00 🡪0

Yawspeed: 00 00 00 00 🡪0

Lat: 4c 52 40 1c 🡪 473977420

Lon: 44 f4 17 05 🡪 85455940

Alt: 00 00 00 00 🡪0

Vx: 00 00 🡪0

Vy: 00 00 🡪0

Vz: 00 00 🡪0

Ind\_airspeed: 00 00 🡪0

True\_airspeed: 00 00 🡪0

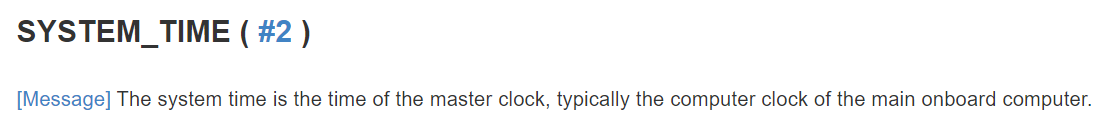
Xacc: 00 00 🡪0

Yacc: 00 00 🡪0

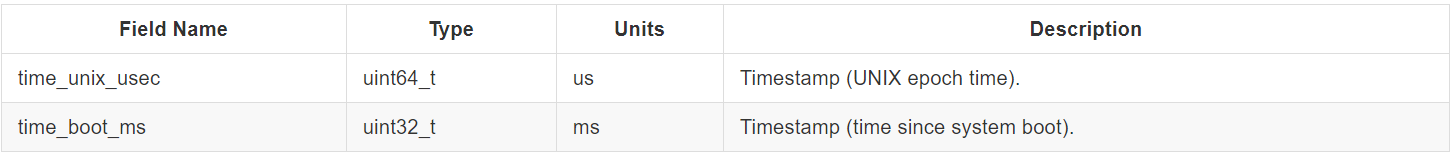
Zacc: 00 00 🡪0 (no tiene mucho sentido este valor…)

1. fd0c0000030133020000104e23e3dfa10500b5ecb01ea735

fd 0c 00 00 03 01 33 **020000** Este id es el 000002 🡪2 que corresponde con el mensaje system\_time



La estructura es la siguiente:



104e23e3dfa10500 b5ecb01e a735

Time\_unix\_usec: 10 4e 23 e3 df a1 05 00🡪 1585357844074000

Time\_boot\_ms: b5 ec b0 1e🡪 514911413

1. fd4000000401336b0000a8b21ce3dfa105005591bbbcd751b0bc8ff91ac1cc974b3cd351c1bb21515b3cd12e563e76e1cd3b446bdd3ecdff6e44000000002bfbf34300000000ff1b0000ded9

fd 40 00 00 04 01 33 **6b0000** el id del mensaje es 00006b 🡪 107 que es el mensaje de HIL\_SENSOR. Esta es la segunda iteración del mensaje de los sensores, con esto se puede saber el time\_step que hay que darle a la simulación.

a8b21ce3dfa10500 5591bbbc d751b0bc 8ff91ac1 cc974b3c d351c1bb 21515b3c d12e563e 76e1cd3b 446bdd3e cdff6e44 00000000 2bfbf343 00000000 ff1b0000 ded9

Time\_usec: a8 b2 1c e3 df a1 05 00 🡪 1585357843641000 (La diferencia entre este tiempo y el del mensaje anterior de los sensores da el time\_step en us)

Time\_step: 1585357843641000 - 1585357843637000= 4.000 us entre cada timestamp para simular el avance de la simulación.

Ahora se valida el código implementado con wireshark :