Apéndice E

Protocolo de comunicación del brazo robótico

En este anexo se describe el protocolo de comunicación a bajo nivel que que se ha diseñado para codificar el paso de mensajes entre el brazo robótico y un controlador externo, en este caso la interfaz gráfica diseñada.

Se trata de un protocolo de comunicación para mensajes de longitud variable multipropósito y con comprobación de errores mediante un *checksum*.

El protocolo se ha diseñado pensando en posibilitar y facilitar su ampliación posterior. Por ahora se incluyen algunas funcionalidades básicas que podrán ser ampliadas.

Tal y como se describe en el anexo [**?**] la información enviada va codificada en formato hexadecimal, byte a byte. Nuevamente los mensajes contienen uos elementos comunes:

- Encabezado: Los primeros dos bytes del mensaje estarán compuestos por encabezado que será el que marque el inicio del mensaje. Estos bytes serán: 0xFF 0xFF
- Longitud: se codifica, en el propio mensaje, la longitud en bytes que se debe leer.
- Un fin de mensaje: El último byte del mensaje estará marcado por un valor llamado *CheckSum* que será el encargado de verificar que todo el paquete ha llegado correctamente. El *CheckSum* es el inverso del valor binario de la suma de todos los bytes enviados a excepción del encabezado y el propio *CheckSum*.

Los paquetes enviados, en ambos sentidos, comparten la siguiente estructura:

- Bytes 0 y 1: reservados para el encabezado.
- Byte 2: Codifica la longitud del mensaje sin contar el encabezado. Se puede entender como el número de bytes a leer una vez detectado el inicio del mensaje.
- Byte 3: codifica la instrucción que se desea realizar. Se han establecido diferentes tipos de mensaje, interpretándose cado de una forma e identificandose mediante un número codificado en este byte.

- Bytes del 4 al N: Parámetros que se quieran envian.
- Byte N+1: *CheckSum*

En la figura E.1 se puede ver una representación gráfica de este esquema de mensaje.

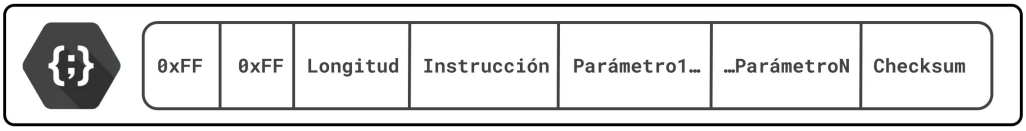

Figura E.1: Paquete de información genérico para comunicar con los Servos G15 Cube

Como se ha visto, el byte 3 acepta diferentes valores en función del propósito del mensaje. Estas funcionalidades se codifican con un id, que en binario será:

- Opción 0: Mensaje de actualización de la información del robot, en el cual el mismo codifica la información general sobre su estado.
- Opción 1: Mensaje de error del robot, donde se codifican y envian errores en las articulaciones o servos.
- Opción 2: Comandos para actuar sobre el robot. En este caso es la interfaz gráfica (o cualquier otro controlador con una conexión serie) quien evía comandos de posiciones articulares a alcanzar al robot.

E.1. Estado del robot

En la tabla E.1 puede verse la cadena de mensaje que se envía con este propósito. Cabe destacar que parte de los datos sobrepasan el valor que puede codificarse en un byte y se codifican en dos quedando el de más bajo nivel el primero seguido del byte superior.

Aunque la información codificada es la que se presenta en la tabla, ésta puede ser ampliada de forma sencilla añadiendo más bytes al mensaje.

E.2. Informes de error

El robot es capaz de notificar errores en los servos. Este tipo de mensaje se construye de la siguiente manera:

- Bytes 0 y 1: con el encabezado: 0xFF en ambos.
- Byte 2: 5 (longitud)
- Byte 3: 0x01 (tipo de mensaje).
- Byte 4: Codifica el estado de los servos. Siguiendo la codificación descrita en la tabla E.2 se puede almacenar la información de hasta ocho servos en un solo byte. Este error codifica posibles errores en la comunicación con los servos, cada servo se identifica según la articulación en la que se encuentra, no según el ID interno a cada uno. Se almacena el error poniendo el bit correspondiente a uno y la ausencia del mismo poniéndolo a cero

Tabla E.1: Codificación del estado del robot.

Fuente: Autor

Byte	**Información contenida**
0	Encabezado (Fijo: 0xFF)
1	Encabezado (Fijo: 0xFF)
2	Longitud a leer (18)
3	Tipo de mensaje (0)
Información de la primera articulación	
4	Posición articular
5	Velocidad (con dirección). Byte L
6	Velocidad (con dirección). Byte H
7	Par aplicado (con dirección). Byte L
8	Par aplicado (con dirección). Byte H
Información de la segunda articulación	
9	Posición articular
10	Velocidad (con dirección). Byte L
11	Velocidad (con dirección). Byte H
12	Par aplicado (con dirección). Byte L
13	Par aplicado (con dirección). Byte H
Información de la tercera articulación	
14	Posición articular
15	Velocidad (con dirección). Byte L
16	Velocidad (con dirección). Byte H
17	Par aplicado (con dirección). Byte L
18	Par aplicado (con dirección). Byte H
19	Checksum

- Byte 5: Como en el caso anterior y siguiendo las indicaciones de la tabla E.3 se puede almacenar el estado de error de hasta ocho articulaciones en un byte. Este error informa de posibles inconsistencias en el funcionamiento de la articulación. Un caso de ejemplo puede ser que el movimiento del servo no se traduzca en un movimiento articular.
- Byte 6: *CheckSum*

Tabla E.2: Codificación de error en servos.

Fuente: Autor

Bit	**Información contenida**	**Máscara de error**
0	error servo (articulación 1)	0x01
1	error servo (articulación 2)	0x02
2	error servo (articulación 3)	0x04

Tabla E.3: Codificación de error en articulaciones.

Fuente: Autor

Bit	**Información contenida**	**Máscara de error**
0	error articulación 1	0x01
1	error articulación 2	0x02
2	error articulación 3	0x04

E.3. Comandar el robot

El mensaje que codifica objetivos para el robot es similar a los mensajes enviados por el mismo, manteniendo la misma estructura incluye:

- Bytes 0 y 1: con el encabezado: 0xFF en ambos.
- Byte 2: 6 (longitud del mensaje sin encabezado)
- Byte 3: 0x02 (tipo de mensaje).
- Byte 4: Objetivo de posición para la primera articulación.
- Byte 5: Objetivo de posición para la segunda articulación.
- Byte 6: Objetivo de posición para la tercera articulación.
- Byte 7: *CheckSum*