Universidad de Barcelona

Arthur Font Gouveia 20222613

Ángel Rubio Giménez 20222484

Programación de Arquitecturas Empotradas

Práctica 4 - Conexión al Robot

Barcelona

2020

Índice

1. Proyecto

1.1 Objetivos

El objetivo de esta práctica es aprender sobre cómo funciona la conexión al robot, más específicamente la comunicación del microcontrolador a los módulos AX-12 (motores) y AX-S1 (sensores) del robot.

Primero debemos entender la configuración del Unified Clock System (*Unified Clock System*) para que la UART (*Universal Asincronous Receiver Transmiter*) trabaje a la frecuencia requerida por los módulos Dynamixel. También debemos ir modificando los registros de la UART para que se pueda establecer la comunicación y así enviar y recibir paquetes de información de los módulos (funciones TxPacket() y RxPacket()).

Por último, nos dedicaremos a la creación de una librería de funciones para controlar robot como por ejemplo mover adelante, mover atrás, girar, reducir velocidad, capturar datos del sensor...

1.2 Recursos utilizados

Hemos utilizado los módulos del robot Dynamixel AX-12 (motores, con ID2 y ID3) y AX-S1 (sensores, con ID1). La configuración de la UCS para que la UART trabaje a la frecuencia requerida por los módulos del robot ya estaba implementada en grande parte en el código proporcionado por el profesorado.

Por parte de los timers, hemos utilizado el timer A1 para la detección de errores durante la recepción de los datos (time out).

Si tuviéramos el robot disponible, también podríamos utilizar la pantalla para enseñar que valores están recibiendo los sensores, la dirección en que se mueve el robot y su estado actual.

1.3 Análisis y configuración de los recursos

Los módulos Dynamixel utilizan la comunicación asíncrona Half-duplex (tienen una línea para transmitir y recibir) mientras que la UART en principio es Full-duplex (tiene una línea para transmitir UCAxTXD y una para recibir UCAxRXD). Por lo tanto necesitamos decir al microcontrolador si queremos transmitir o recibir datos. Esto lo conseguiremos controlando la señal mediante el valor de pin P3.0 (DIRECTION_PORT) por medio de las funciones Sentit_Dades_Rx_emu() y Sentit_Dades_Tx_emu().

Para establecer comunicación segura entre los módulos del robot y la UART la función També tenim la funció void TxUAC2_emu(byte bTxdData) hace el processador esperar hasta que el buffer de datos esté listo para la transmisión de datos.

Una vez disponemos de los recursos descritos arriba, debemos implementar la funciones RxPacket() y TxPacket(), que son responsables por enviar y recibir la información. Para eso, tenemos que tener claro el protocolo de comunicación entre los módulos del robot y el microcontrolador, que se basa en: el microcontrolador envía un "InstructionPacket" (formado por varios bytes) al módulo del robot y éste le contesta con un "Status Packet".

Sigue abajo la información detallada del “InstructionPacket" y del "Status Packet":

INSTRUCTION PACKET

Format dels Paquets: seqüència de bytes enviat pel microcontrolador

OXFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	...	PARAMETER N	CHECK SUM

0xFF, 0xFF: Indiquen el començament d’una trama.

ID: Identificador únic de cada mòdul *Dynamixel* (entre 0x00 i 0xFD).
El identificador 0xFE és un “*Broadcasting ID*” que van a tots els mòduls (aquests no retornaran *Status Packet*)

LENGTH: El número de bytes del paquet (trama) = Nombre de paràmetres + 2

INSTRUCTION: La instrucció que se li envia al mòdul.

PARAMETER 1…N: No sempre hi ha paràmetres, però hi ha instruccions que si necessiten.

CHECK SUM: Paràmetre per detectar possibles errors del comunicació, es calcula així:

Check Sum = ~(ID + LENGTH + INSTRUCTION + PARAMETER 1 + ... + PARAMETER N)

Figure 1*. Instruction Packet details*

STATUS PACKET

Format dels Paquets: seqüència de bytes amb que respon el mòdul

OXFF	0XFF	ID	LENGTH	ERROR	PARAMETER1	PARAMETER2	...	PARAMETER N	CHECK SUM

0xFF, 0xFF: Indiquen el començament d'una trama.

ID: Identificador del mòdul.

LENGTH: El número de bytes del paquet.

ERROR:

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	Overload Error	Set to 1 if the specified maximum torque can't control the applied load.
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect.
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range.
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

PARAMETER 1...N: Si es necessiten.

CHECK SUM: Paràmetre per detectar possibles errors del comunicació, es calcula així:

Check Sum = ~(ID + LENGTH + ERROR + PARAMETER 1 + ... + PARAMETER N)

***Figure 2**. Status Packet details*

1.4 Documentación de la librería

La librería que hemos implementado esta compuesta por diversas funciones. Primero hemos analizado las siguientes funciones para así poder entender y completar las funciones TxPacket (), que envía un paquete de datos a un módulo concreto, y RxPacket (), que recibe y guarda el paquete de respuesta de los módulos

- Sentit_Dades_Rx_emu (): Esta función imprime por pantalla el mensaje de que se ha cambiado la direción para el sentido de recepción de datos (RX). Pero realmente debería cambiar el valor del pin P3.0 (DIRECTION_PORT) para ponerlo en el sentido de recepción de datos.

- Sentit_Dades_Tx_emu (): Esta función imprime por pantalla el mensaje de que se ha cambiado la dirección para el sentido de transmisión de datos (TX). Pero realmente debería cambiar el valor del pin P3.0 (DIRECTION_PORT) para ponerlo en el sentido de transmisión de datos.

- TxUAC2_emu (byte bTxdData**):** Esta función guarda y pone el byte que queremos enviar al registro UCA2TXBUF para poder establecer la comunicación a los módulos del robot.

Luego completamos la implementación de las funciones TxPacket() y RxPacket() para la transmisión de datos, explicadas a seguir con más detalles:

- TxPacket (byte bID, byte bParameterLength, byte bInstruction, const byte * Parametros**):**

Esta función es responsable por enviar byte a byte un paquete de información que queremos que reciba el módulo con el que nos comunicamos. Este paquete que enviamos (I*nstruction Packet*), está compuesto por:

- Dos bytes que indican el comienzo de una trama (0xFF)
- Un byte con el ID del módulo con el que nos comunicamos (el ID de broadcast es el 0xFE, los motores tienen los ID de 0x01 al 0x04, y el sensor tiene el ID 0x64, pero en esta práctica hemos utilizado el ID 0x01)
- Un byte con el número total de bytes del paquete (*length*)
- Un byte con la instrucción a transmitir al módulo (0x02 es READ, 0x03 es WRITE, 0x04 es REG_WRITE y 0x05 es ACTION)
- N bytes con los parámetros necesarios para la instrucción.
- Por último, un byte con el valor referente al checksum del paquete.

- RxPacket (): Esta función es responsable por recibir un paquete de información de un módulo que responde a un Instruction Packet. Este paquete, que se llama Status Packet, permite hacer lecturas de los registros propios de los módulos y también se puede saber si ha pasado algún error en la transmisión de los datos . Este paquete de respuesta, de forma similar que los paquetes de transmisión, está compuesto por:

- Dos bytes de aviso (0xFF).
- Un byte con el ID del módulo que envía el paquete.
- Un byte con la longitud total del Status Packet (*length*)
- Un byte con el código de error.
- N bytes de parámetros adicionales.
- Por último, el byte del checksum, para comprobar que hayamos recibido el paquete correctamente y no ha ocurrido ningún error de transmisión.

Finalmente, ya tenemos implementadas las funciones necesarias para comunicarnos con el robot. Por lo tanto, para controlar el robot hemos creado la siguiente librería de funciones.

Por parte de los motores, las funciones disponibles son:

- **move_forward ():** Esta función escribe en los registros Moving Speed (0x20) de los motores ID2 i ID3 en el sentido debido para que el robot se mueva adelante.
- **move_backward ():** Esta función escribe en los resgitros Moving Speed (0x20) de los motores ID2 i ID3 en el sentido debido para que el robot se mueva atrás.
- **turn_right ():** Esta función escribe en los registros Moving Speed (0x20) de los motores ID2 i ID3 en el sentido debido para que el robot se gire a la derecha.
- **turn_left ():** Esta función escribe en los registros Moving Speed (0x20) de los motores ID2 i ID3 en el sentido debido para que el robot se gire a la izquierda
- **stop_movement ():** Esta función escribe en los registros Moving Speed (0x20) de los motores ID2 i ID3 para que el robot pare totalmente su movimiento.
- **fast_move_forward ()**: Esta función escribe en los registros Moving Speed (0x20) de los motores ID2 y ID3 en el sentido debido para que el robot se mueva adelante con una velocidad elevada.

Para cada función hay otra que sirve para comprobar que los registros tienen los valores correctos para realizar el movimiento deseado.

Por parte de los sensores, las funciones disponibles son:

- **get_front_sensor ()**: Esta función lee el sensor frontal del módulo AX-S1 del robot y imprime el valor en la pantalla LCD. Para eso, lee el valor del registro CCW Compliance Margin (0x1B).
- **get_right_sensor ():** Esta función lee el sensor derecho del módulo AX-S1 del robot y imprime el valor en la pantalla LCD. Para eso, lee el valor del registro CW Compliance Slope (0x1C).
- **get_left_sensor ():** Esta función lee el sensor izquierdo del módulo AX-S1 del robot y imprime el valor en la pantalla LCD. Para eso, lee el valor del registro CW Compliance Margin (0x1A).

- **set_front_sensor ()**: Esta función escribe en el sensor frontal del módulo AX-S1 del robot y imprime el valor en la pantalla LCD. Para eso, lee el valor del registro CCW Compliance Margin (0x1B).
- **set_right_sensor ():** Esta función escribe en el sensor derecho del módulo AX-S1 del robot y imprime el valor en la pantalla LCD. Para eso, lee el valor del registro CW Compliance Slope (0x1C).
- **set_left_sensor ():** Esta función escribe en el sensor izquierdo del módulo AX-S1 del robot y imprime el valor en la pantalla LCD. Para eso, lee el valor del registro CW Compliance Margin (0x1A).

Para cada sensor hay también una función test que sirve para comprobar que el valor del sensor es el correcto.

1.5 Problemas

Hemos tenido dificultades a la hora entender grande parte del código proporcionado, por ejemplo cómo funcionaba las funciones RxPacket() y TxPacket(), cómo manejar los registros de la UART.

Pero después de leer al material de teoría que hay en el Campus Virtual, buscar información a la internet y leer las documentaciones al fondo pudimos entender mejor el funcionamiento. También tuvimos problemas en la implementación de las funcionalidades del robot y sus testes, una vez que no tenemos el robot a disposición para probar todas las funcionalidades implementadas.

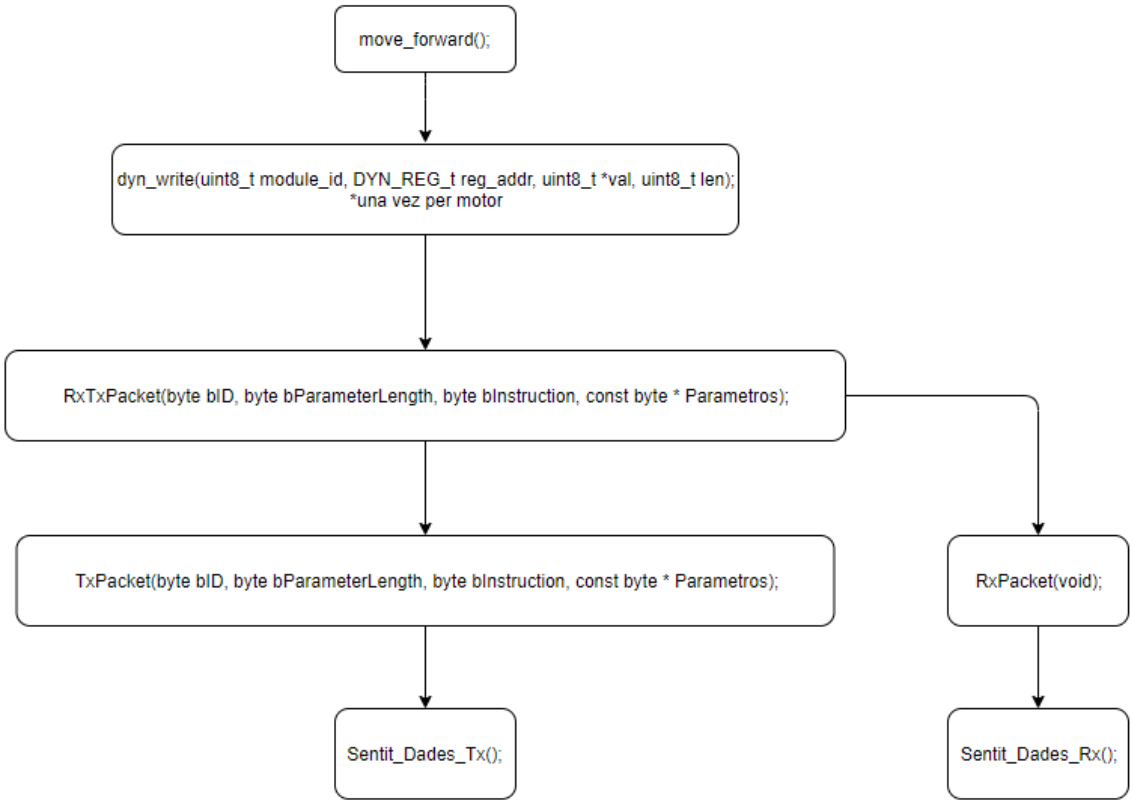
1.6 Conclusiones

Al principio tuvimos dificultades de entender el código que nos fue ofrecido, pero gracias al soporte del profesorado y a las documentaciones de los módulos AX-S1 y AX -12 disponibles en el Campus Virtual hemos comprendido toda la mecánica de comunicación con el robot y por lo tanto hemos tenido condiciones y las informaciones necesarias para la implementación de la librería de funciones para controlar el robot.

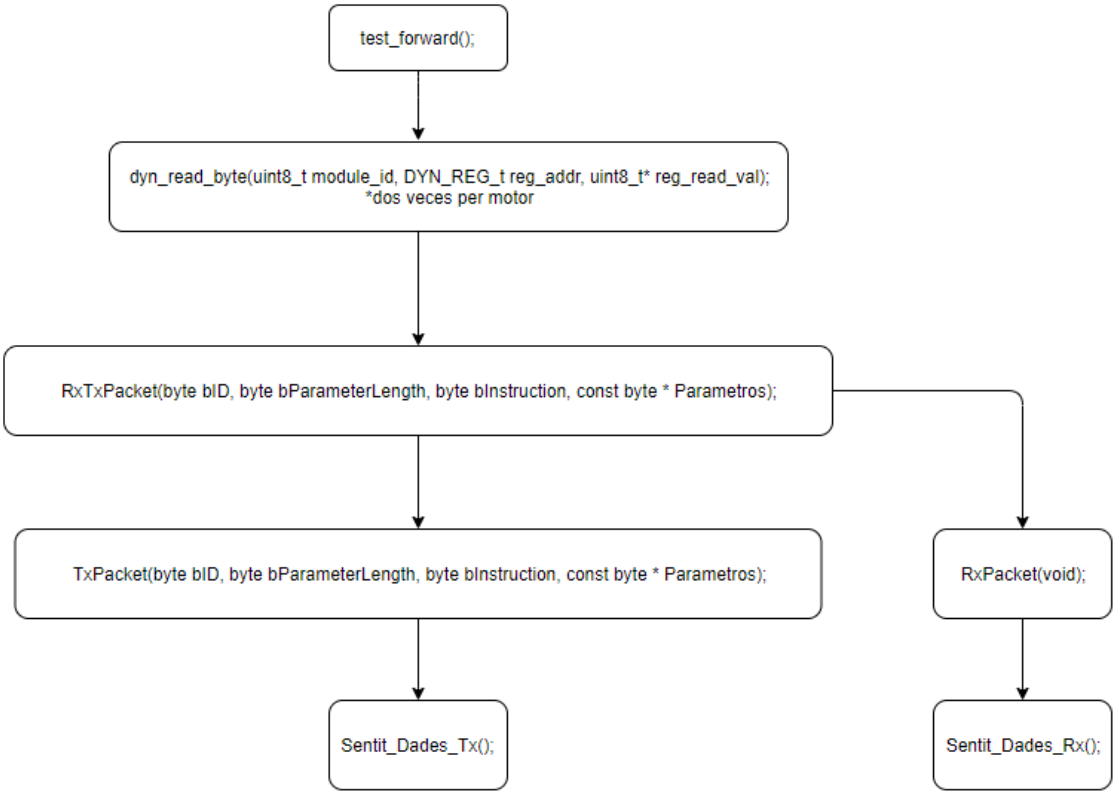
Debido a las largas horas gastas en la búsqueda y a la implementación obtuvimos resultados y una buena base para poder llevar a cabo el proyecto final de la asignatura.

2. Diagrama de bloques

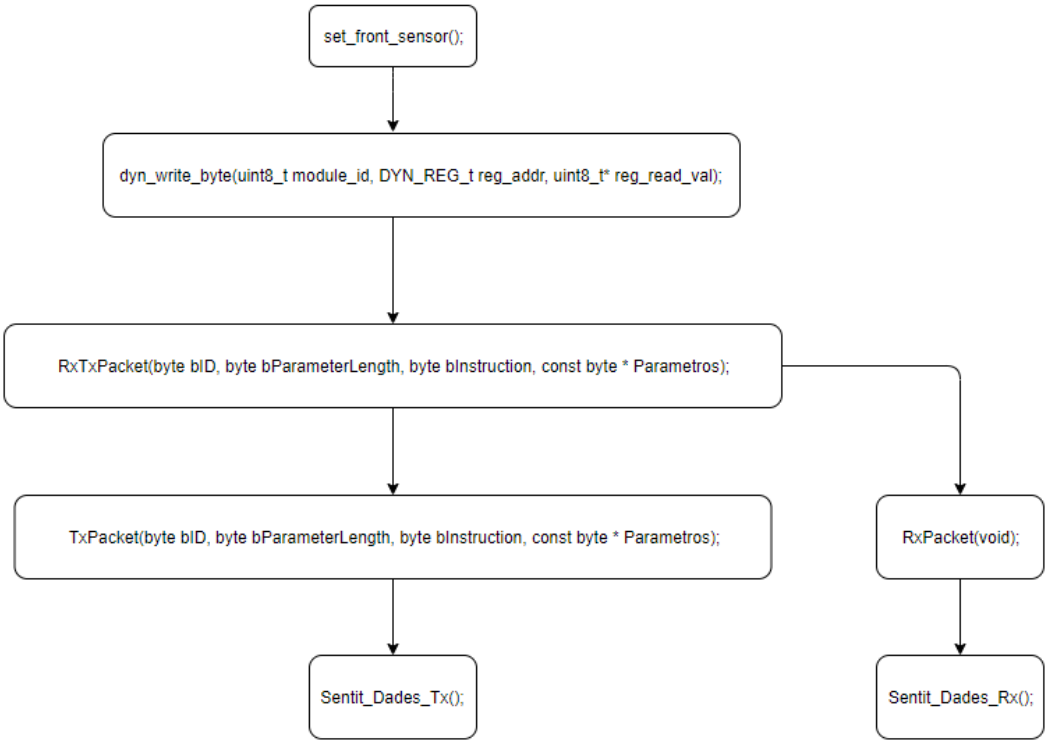
Ejemplo de diagrama para las funciones que hacen mover el robot.



Ejemplo de diagrama de las funciones que testean el movimiento del robot



Ejemplo de diagrama para cambiar el valor de un sensor.



El resto de funciones utilizan uno de los esquemas anteriores para escribir o leer valores de los registros pero cambiando los parámetros de entrada de la función.