Informe pràctica 4

Eduard Martín Graells
Joan Peracaula Prat

March 2018

1 Introducció

En aquesta última pràctica l'objectiu és preparar el robot per al projecte final. Això inclou establir un canal de comunicació entre el microcontrolador i els diferents mòduls (sensor i motors) i un conjunt de funcions per tal de tenir una llibreria que ens permeti controlar el robot.

Per fer-ho, crearem una funció que configuri els mòduls, una per enviar "ordres" als mòduls des del microcontrolador i una altra per rebre'n informació. A partir d'aquestes funcions i guiant-nos pel manual de cada mòdul definirem una llibreria que permetrà controlador el robot a través de funcions a alt nivell.

A raó de proporcionar una manera ràpida de comprovar el funcionament de la llibreria, hem afegit una sèrie de funcions que a partir d'un "input" de la placa superior (*Joystick*, S1...) utilitza la llibreria per generar un "output" que pot observar la persona operant el robot.

2 Recursos utilitzats

Els recursos que hem utilitzat per implementar la comunicació i la llibreria han estat: els mòduls dels dos motors (*Bioloid AX-12*), el del sensor (*Bioloid AX-S1*), la *UART* i el *Timer* TA1 que l'utilitzarem amb la font de rellotge ACLK (*Auxiliary Clock*).

Després, el *Joystick*, els botons *S1* i *S2*, la pantalla LCD, els LEDs RGB de la placa superior per tal d'interactuar amb l'usuari del robot.

A aquestes alçades del curs, la majoria de recursos mencionats anteriorment ja s'ha entès i treballat; excepte els mòduls i les rutines per utilitzar-los, que és on es centra aquesta pràctica.

3 Configuració dels recursos

Com ja he comentat, ens centrarem en els mòduls i en la seva configuració. Sobre la resta de recursos dir breument que la seva inicialització està encapsulada i s'utilitzen correctament.

Seguint la teoria donada a classe el que hem fet és generar dues funcions per comunicar els mòduls *Dynamixel* amb el controlador utilitzant la *UART*. El primer que hem de fer per tenir comunicació és inicialitzar aquest últim component, la *UART*, en la funció *init_UART()*; iniciant-li un *baud rate* igual que el dels mòduls i la "setegem" de manera que rebi dades indicant-ho en el *Direction_Port*, que en el nostre cas és el pin 3.0 que iniciarem com a GPIO. Recordem que els mòduls són *half-duplex*, obligant-nos a canviar de direcció el nostre canal depenent de si volem rebre o transmetre.

A continuació habilitem la interrupció de la *UART* en els tres diferents nivells (vistos en pràctiques anteriors) per tal de rebre informació dels mòduls de manera asíncrona, tenint en compte que és el bit 18 del *NVIC*.

Un cop configurada la *UART*, hem implementat les rutines que realment envien i retornen instruccions dels mòduls, que tot i que realment no és tant configurar recursos com utilitzant-los, creiem que s'haurien de comentar en aquesta secció.

La funció *TxPacket* que s'encarrega de la comunicació del microcontrolador als mòduls, és a dir, enviar instruccions. Aquesta funció ens la van donar feta i, bàsicament, el que fa és a partir d'uns paràmetres, crea una instrucció que passa byte a byte al mòdul amb la ID especificada.

La funció *RxPacket* és el contrari que l'anterior. El mòdul ens retorna byte per byte una resposta que nosaltres hem de reconstruir en un "status packet" per saber quin és el seu estat. Tot i que aquesta funció ens la donaven començada i en un principi semblava que simplement necessitàvem donar la volta a la funció de transmetre i ajustar-nos a la diferent estructura del "status packet", ha sigut una tasca més complicada del que esperàvem. El que fem és posar el robot en mode "escoltador", és a dir, posem la *UART* en sentit de rebre i iniciem un comptador per tal que el robot no es quedi massa temps sense fer res en el cas de que no arribi cap dada. Anem llegint byte a byte que ens arriben per interrupció que genera el mòdul fins arribar al quart, que ens indica quants bytes queden per rebre. Llegim la resta i comprovem si el *checksum* que ens retorna és correcte. Si és així retornem la trama, si hi ha hagut error de *checksum* o *timeout*, també retornem un valor d'error que ens indica què ha passat.

4 Funcions dels recursos

Hem implementat diverses funcionalitats que podrien ajudar en el desenvolupament del projecte final: detecció de distàncies, detecció de llum, detecció de sorolls i control dels dos motors.

Els tres primers es basen en la utilització del mòdul del sensor (*AX_S1*. El que fem és enviar-li una instrucció dient que volem llegir els valors de certs registres (corresponents al que volem mesurar) i aquest ens retorna un "status packet" que conté la informació.

El control dels motors (*AX_12* en canvi es basa en enviar-li instruccions de *WRITE* en els registres que emmagatzemen la velocitat i la direcció en la que el motor ha de rotar. A partir d'aquesta idea hem fet cinc funcions: una per anar cap endavant, una per anar endarrere, per girar a l'esquerra i a la dreta i una última per parar.

Un cop s'han entès els manuals dels mòduls aquestes funcions són bastant simples d'escriure ja que no és més que veure quin registre és el que necessites i passar-li el valor que desitges.

5 Problemes

Tot i que al final ens van aplaçar l'entrega, crec que aquesta pràctica és la que més problemes ens ha donat i més temps hem dedicat. Comprensible al ser el penúltim treball que farem en aquesta assignatura.

Lògicament la majoria de problemes ha sigut en els mòduls *dynamixel* ja que els tractàvem per primer cop.

Les funcions de transmetre i rebre semblaven bastant clares i senzilles un cop les llegíem sense pressa. No obstant això, la de rebre no ens funcionava (sortint per *timeout* sempre); no entenent perquè vam revisar tot el codi fins que ens vam fixar que no arribàvem a activar la interrupció de la *UART*, fent que sigués impossible que rebéssim cap informació dels mòduls.

De manera semblant, l'últim dia vam estar programant el càlcul de lluminositat que emetíem per la pantalla *LCD*. Quan passàvem de llum a quasi no llum per alguna raó el nombre emès en comptes de baixar pujava fins a valors superiors d'un byte, fet impossible donat que el valor que tractem és com a màxim un byte. Després de rumiar durant temps ens vam adonar que era el mateix error que ens passava en la practica 3: no formatejar l'output del *printf* perquè quan el valor tingui menys dígits fiqui zeros com "placeholders". No obstant, això només ho utilitzàvem per fer testeig, és per això que en la versió que hem entregat del codi no apareixen; excepte el del càlcul de distància que ho mostrem a través del LED RGB.

També hem tingut certs problemes lleus com ara que "màgicament", al afegir una línia de codi en una funció del control dels motors, la resta deixés de funcionar, si més no, tractant les interrupcions i amb el *debug* anant línia a línia sense cap problema.

6 Conclusió

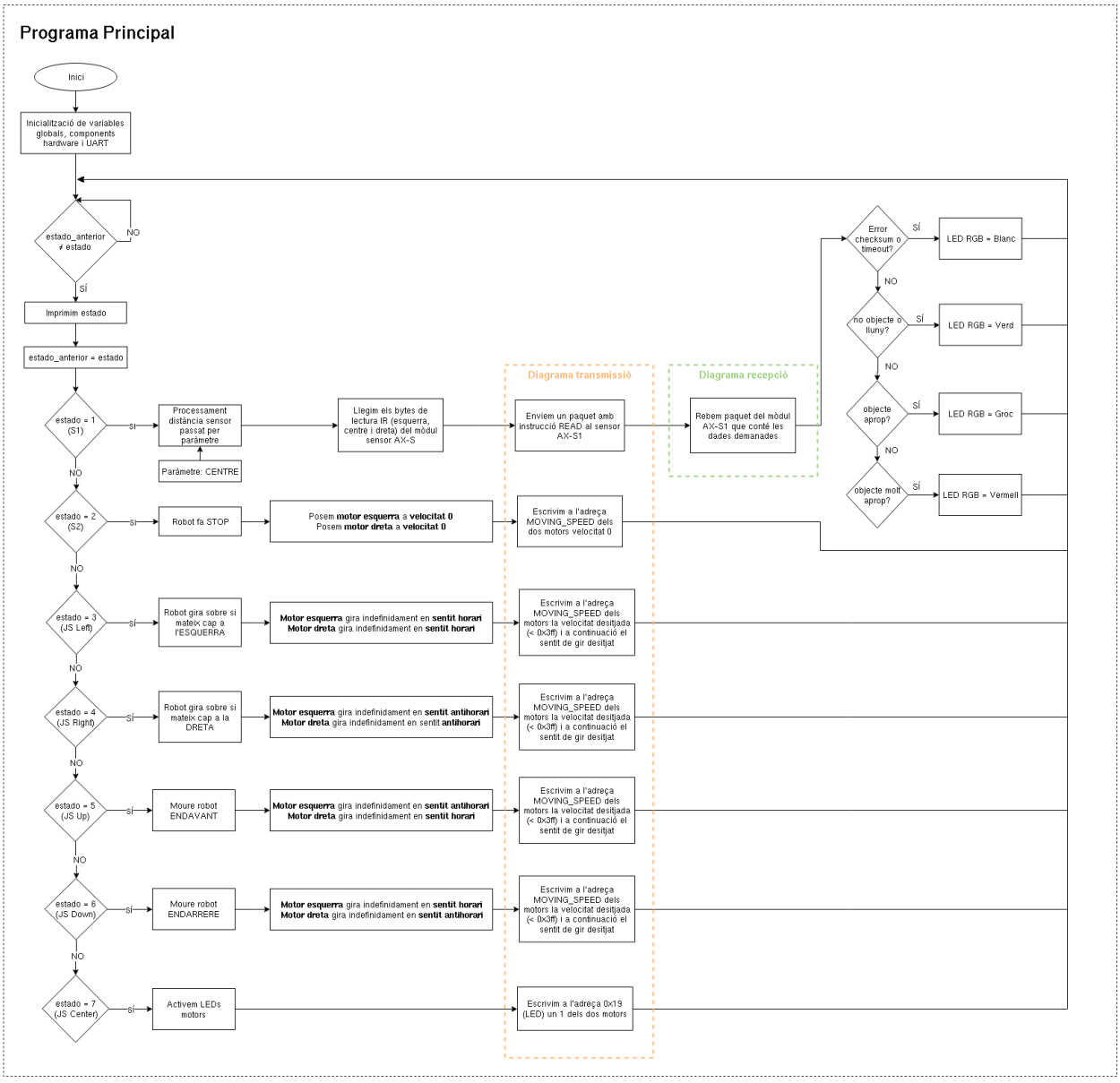
Aquesta pràctica ha servit per concloure el procés d'aprenentatge de la manipulació del robot. Hem consolidat els recursos que ja coneixíem com el *Timer* i n'hem afegit de nous com el control dels mòduls *dynamixel*.

En particular hem aprés com es comunica un microcontrolador amb diferents mòduls, com configurar aquest canal i com enviar i rebre informació. Conseqüentment permetent-nos que el nostre robot interactuï amb el seu entorn, deixant de ser un objecte estàtic i cec.

Encara que aquesta pràctica ha sigut molt entretinguda per la seva pròpia natura, no podem deixar de pensar que amb més temps podríem haver-li tret més suc. No obstant això, crec que és un bon reflex del que ens podríem trobar en projectes futurs on treballar contra rellotge és un fet habitual.

7 Diagrames

7.1 Main



7.2 Lectura

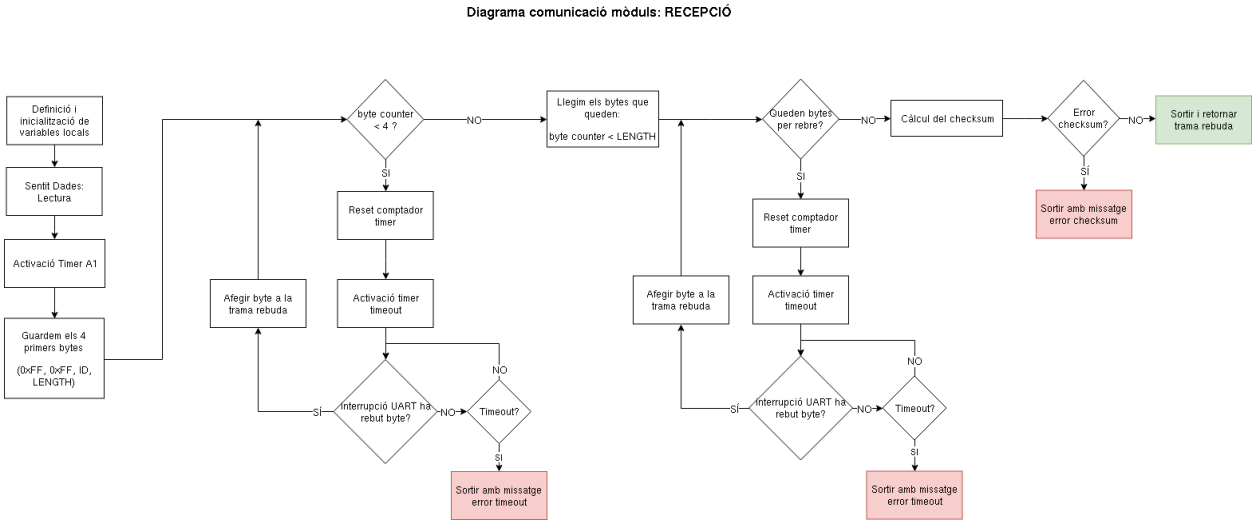
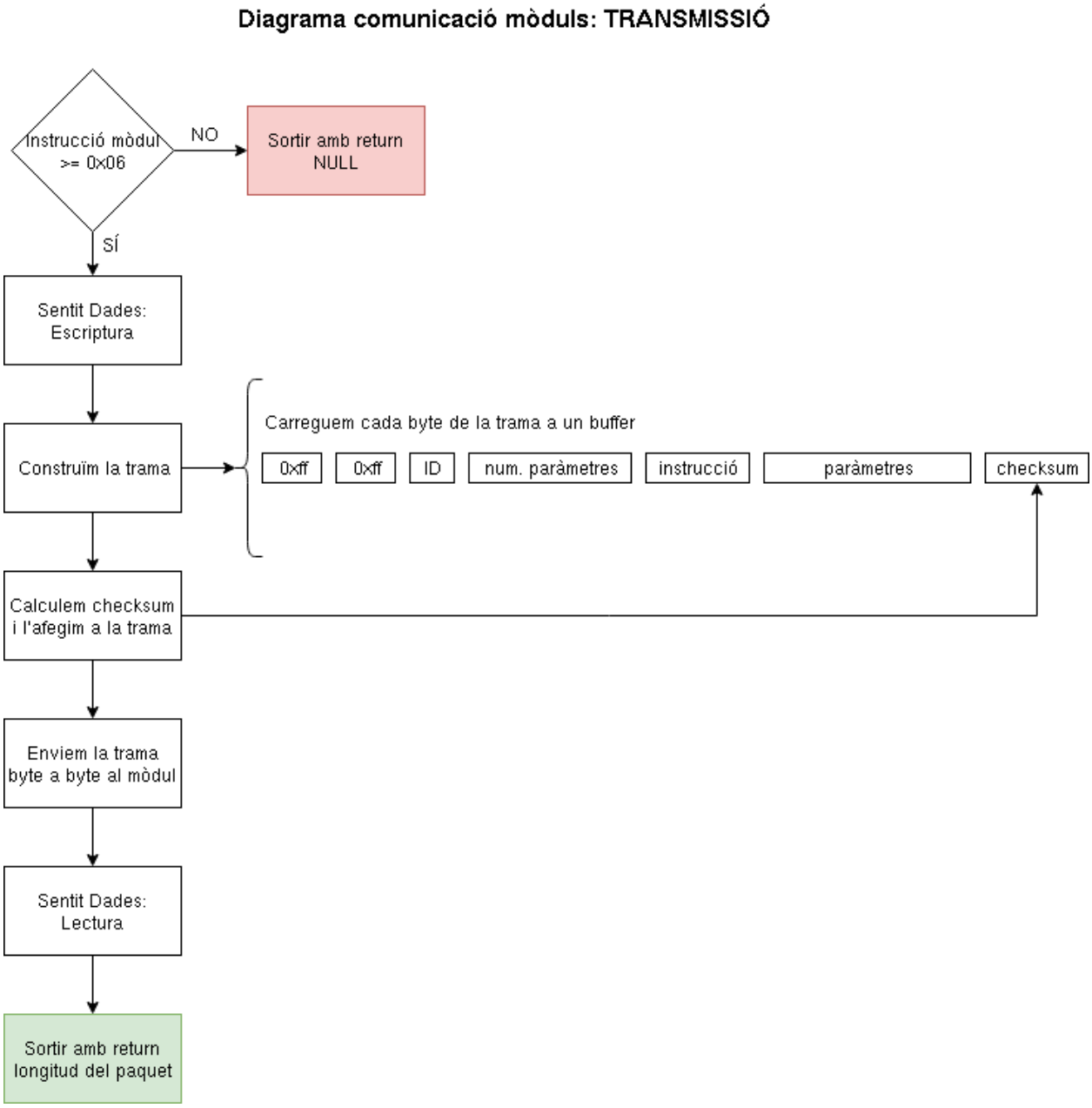


Diagrama comunicació mòduls: RECEPCIÓ
Definició i inicialització de variables locals
Sentit Dades: Lectura
Activació Timer A1
Guardem els 4 primers bytes
(0xFF, 0xFF, ID, LENGTH)
byte counter < 4 ?
NO
SI
Reset comptador timer
Activació timer timeout
Interrupció UART ha rebut byte?
SÍ
NO
Timeout?
NO
SI
Sortir amb missatge error timeout
Afegir byte a la trama rebuda
Llegim els bytes que queden:
byte counter < LENGTH
Queden bytes per rebre?
NO
SI
Reset comptador timer
Activació timer timeout
Interrupció UART ha rebut byte?
SÍ
NO
Timeout?
NO
SI
Sortir amb missatge error timeout
Afegir byte a la trama rebuda
Càlcul del checksum
Error checksum?
NO
SÍ
Sortir amb missatge error checksum
Sortir i retornar trama rebuda

7.3 Escriptura



7.4 Interrupcions

Gestió de les interrupcions