Marcos Plaza González
Niub: 20026915

Connexió amb el robot - Creació de les Funcions Bàsiques de Comunicació i d'una Llibreria de Funcions per Controlar el Robot - Informe

Quins són els objectius de la pràctica?

En aquesta pràctica ens centrarem en la comunicació del microcontrolador amb els mòduls del robot; AX-12: Motors, AX-S1: Sensors. Principalment tractarem tot el relatiu amb la configuració per a comunicar el micro amb els móduls anteriorment esmentats, és a dir la configuració de la Unified Clock System la qual s'encarrega de que la comunicació UART treballi a la freqüència requerida pels mòduls Dynamixel i la configuració de la UART. Conseqüentment estudiarem com funciona el flux de transimssió de dades entre el micro i el robot, i emplearem dues funcions per a transmetre i rebre 'paquets' de dades. Posteriorment utilitzarem tot això per a crear una llibrería per al moviment dels motors i la detecció d'elements mitjançant els sensors on provarem que tot funcioni de manera correcta.

Quins recursos farem servir?

Dins el microcontrolador trobem la *USCI (Universal Serial Communication Interface)* la qual estarà programada amb el tipus de comunicació *UART (Universal Asincronous Receiver Transmiter),* donat que el protocol de comunicació entre la placa i el robot serà en sèrie asíncron. A la vegada al micro trobem que només 4 *USCIs* treballen amb aquest tipus de comunicació. Tal i com diu a l'enunciat de la pràctica nosaltres farem servir la *UCA2* la qual accedirem mitjançant el port 3 amb dos pins de lectura/escritura.

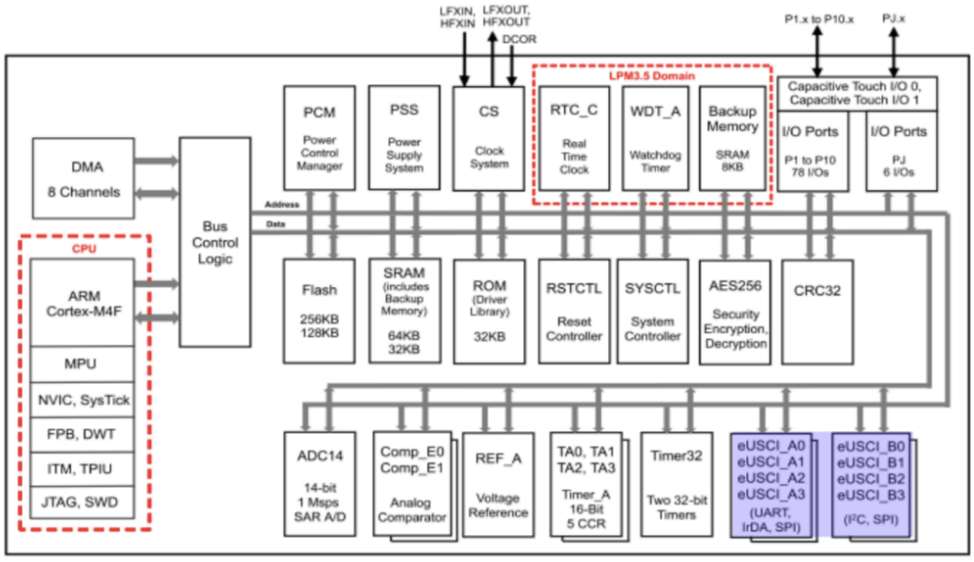

Figura 1. Esquema dels recursos del micro

Marcos Plaza González
Niub: 20026915

Com ja sabem un altre dels recursos que farem servir, seràn els mòduls del robot *Dynamixel AX-12 (Motors, amb ID2 i ID3)* i *AX-S1 (Sensors, amb ID100)*.

A més farem ús dels *timers* (més concretament el *timer A1*) per a la detecció d'errors durant la recepció de les dades (*time out)*. Hem de recordar que també hem de configurar la *UCS* per a que la *UART* treballi a la freqüència requerida pels mòduls del robot.

Per últim, també cal mencionar que farem servir la botonera i el joystick de la placa a més de la pantalla LCD per a informar l'estat en que es troba el robot, quins valors estan rebent els sensors (un cop ja estiguin funcionant), i les diferents direccions que prendrà el robot un cop haguem fet les funcions per moure'l.

Configuració dels diferents recursos

Inicialment passem a fer-li un cop d'ull a la funció void init_UART(void). Tot i que ja ens la dónen feta a les transparències de teoria, cal dir que si la observem podem veure que és la funció encarregada d'inicialitzar els pins per a poder fer possible la transmissió i recepció dels *'paquets d'instruccions'* els quals tenen un format específic en el que haurem de fixar-nos quan haguem de fer les rutines *TxPacket* i *RxPacket*.

Com per a la recepció de bytes mitjançant la UART es fa per interrupcions, necessitem habilitar-les, per això acudim als ports que ja coneixem de les practiques anteriors; els ports *ICPR* i *ISPR*. Com sempre, manipulant *l'ICPR* ens asegurem que no quedi cap interrupció residual pendent per a aquest port i finalment amb *l'ISPR* habilitem la interrupció a nivell de dispositiu. En aquest cas la rutina de la interrupció, l'anomenada void EUSCIA2_IRQHandler(void), desactiva la interrupció en el pin de lectura, llegim les dades en qüestió i a continuació tornem a rehabilitar l'interrupció.

Hem de recordar, de les classes de la teoria de l'assignatura, que els mòduls *Dynamixel* fan servir comunicació asíncrona però *Half-duplex* (el que vol dir que només té una línia *"Data"* per transmetre i rebre) mentre que la *UART* en principi és *Full-duplex* (té una línia per transmetre *UCAxTXD* i una per rebre *UCAxRXD*), per tant necessitem dir-li al micro que volem fer, si transmetre dades o rebre dades, ja que només podrem estar en *Half Duplex*. Això ho aconseguirem controlant el senyal mitjançant el *DIRECTION_PORT* com podem veure a les funcions void Sentit_Dades_Tx/Rx(void), on posem el pin a 1, si volem transmetre dades als móduls i viceversa.

Marcos Plaza González
Niub: 20026915

També tenim la funció void TxUAC2(byte bTxdData) la qual fa que el processador esperi mentres el buffer de dades esta ocupat.

Un cop tenim tot el necessari per a començar la comunicació necessitem fixar-nos primer en el format de les *'trames' (o paquet de dades)* que enviarem i rebrem en les rutines de transmissió *(TxPacket)* i de recepció *(RxPacket)* respectivament.

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)

Figura 2. Format de les instruccions

Podem veure que el format de les dades es molt específic, i he considerat necessari mostrar-ho ja que és important per tal d'entendre el que es fa en cadascuna de les rutines esmentades abans.

Primerament, la funció byte TxPacket(byte bID, byte bParameterLength, byte bInstruction, byte Parametros[16]), s'encarregarà de; coneixent el motor amb el que ens volem comunicar a través de *l'ID,* dir-li a aquest quina instrucció volem que faci, i la direcció de memòria on volem escriure els diferents paràmetres que coneixem a través de l'array *'Parametros'*. Prèviament afegirem un fragment de codi que ens van donar, per a assegurar-nos que no podem escriure fora de les direccions permeses de memòria. Crearem un *buffer* on anirem ficant les dades segons el format de les instruccions a la vegada que habilitem la tràsmissio. Al final de la instrucció haurem de tornar a habilitar el sentit de les dades dels mòduls al micro per tal de poder rebre dades.

Marcos Plaza González
Niub: 20026915

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)

Les dades tindràn un format diferent quan habilitem la recepció d'aquestes, tal i com indica la *figura 3*.

Figura 3. Format de l'Status Packet

Tal i com tenim implementada la subrutina RxPacket, la qual s'encarregarà de llegir tots els bytes rebuts del *Status Packet,* també s'encarregarà de detectar si hi ha hagut algun error mitjançant un parell de funcions auxiliars que hem creat; les anomenades uint32_t checkSum(byte packet[16], uint8_t length) i bool timeOut(uint16_t t). Tal i com es mostra a la f*igura 3* el checkSum és el paràmetre per a comprovar si hi ha hagut errors en la comunicació i es calcula tal i com ho trobem a la fòrmula que tenim en color blau. Haurem de fer el Ca1 de (id + paràmetros + el nombre de paràmetres), que és el que comprovem a la funció checkSum(). D'altra banda la funció timeOut() és l'encarregada de comprovar si el temps que passem per paràmetre és major o menor que el que guarda una variable que augmenta mitjançant la subrutina del timer. Aquesta retornarà true si hi ha un excés de temps en llegir *l'Status Packet.* Com a conseqüència d'això sortirem del bucle i retornarem un error.

Per al *timer A1* que utilitzarem també ens caldrà habilitar la interrupció a nivell de dispositiu a la funció void init_interrupciones(). A la seva subrutina augmentarem progressivament una unitat la variable *'temps'*. D'altra banda tenim la funció void init_timers(), on seleccionem la font del clock, en aquest cas hem aprofitat que l'*SMCLK* pot anar a una freqüència de 24 Mhz, i l'inicialitzem en Mode *STOP,* habilitarem les interrupcions i establirem que cada míl·lessima es produeixi una interrupció (ja que és la velocitat a la que anirà el microcontrolador (init_ucs_24MHz())). D'aquesta manera podem manipular de forma més eficient el temps.

Marcos Plaza González
Niub: 20026915

Per últim dire que tant per als botons com pel joystick com per a la pantalla LCD tot es manté de la mateixa manera que a les dues útimes pràctiques.

Com i per a que farem servir els recursos de la pràctica?

Un cop establert tot el relatiu a la comunicació del microcontrolador amb els mòduls del robot, ara ens cal fer la llibrería de les funcions per a moure'l i detectar obstacles modificant així la seva ruta. A continuació mostraré un manual de funcionament del robot, basant-me en l'estructura swicth(estado) del meu programa i les funcions implementades a la llibrería.

Quan pressionem... (estat)	Funció del robot
El botó S1	El robot començará a moure's endavant a una velocitat moderada de 112 rpm (ja que la velocitat és mou en un rang de 0 a 255) i s'encedràn els leds de moviment.
El botó S2	El robot es dentindrà allà on estigui i les llums de moviment s'aturaràn també.
A dalt al Joystick	Farà exactament el mateix que quan pressionem el botó S1, és mourà a una velocitat moderada cap endevant i els leds de moviment romandràn encessos.
A baix al Joystick	Ara el robot farà la marxa enrrera i els leds de moviment estaràn encessos.
A la dreta al Joystck	El robot girarà en sentit horari i s'indicarà cap a quin sentit s'està anant mitjançant els leds (s'encendrà el led dret i l'altre romandrà apagat).
A l'esquerra al Joystick	Aquest cop el robot girarà en sentit antihorari i s'indicarà quin sentit s'està anant mitjançant els leds (s'encendrà el led esquerra i l'altre romandrà apagat).
El botó central del Joystick	Les llums de moviment s'encendràn i seguirà la direcció de la trayectoria que el robot està seguint en el moment, però aquest cop a la velocitat màxima a la que poden girar les rodes dels motors (255 rpm).

Figura 4. Manual d'instruccions

Marcos Plaza González
Niub: 20026915

Val a dir també que en tot moment s'indicaràn els valors dels tres sensors, a través de la funció struct RxReturn leerSensor(void), i per la pantalla s'indicarà quan el sensor detecti un obstacle proper amb el missatge 'Sensor Activated'. A propòsit, quan el sensor detecti algun obstacle que estigui a prop aquest es detindrà i l'haurem d'allunyar-lo a un lloc on no hi hagin obstacles a prop per a poder tornar-lo a utilitzar.

Totes les funcionalitats anteriorment citades aniràn recolzades pel diagrama de flux que trobarem a la part final d'aquest informe.

Problemes que han sorgit durant la pràctica

Aquesta pràctica ha sigut la més llarga i la més difícil per a mi, per tant és normal que m'hagi sorgit algún que altre problema, tot i que finalment els he aconseguit solucionar amb éxit. Inicialment per a fer les proves de transmissió de les dades amb la funció *TxPacket* vaig pensar que els motors estaven espatllats ja que cap led, ni del motor 2, ni del motor 3, s'encenia i vaig repassar fins a la sacietat la meva funció (tot i que ja ens la donaven feta) quan de sobte em vaig donar compte que el botó de la placa secundària que controla els motors no estava encés. Com a recomanació aquest fet podría constar en algun apartat d'introducció a l'enunciat de la pràctica per a estalviar temps, ja que jo vaig perdre una bona estona amb això.

Val a dir també que a l'inici no sabía com funcionaba del tot la transmissió de les dades per '*paquets*', posteriorment a la classe de teoria ens ho van explicar, cosa que em va ajudar també a plantejar correctament la rutina de recepció de les dades.

Conclusions

Tot i que al principi va ser difícil d'entendre, gràcies a l'ajuda del professorat i al suport tècnic que es pot trobar al Campus Virtual de l'assignatura, puc afirmar que durant aquesta pràctica he après i he resolt tots els dubtes que tenia sobre com funciona aquest tipus de comunicació asíncrona entre motors i el micro. A partir d'aquí considero que tindré una bona base per a poder a dur a terme el projecte final de l'assignatura. Val a dir també que el codi es troba degudament comentat.
A continuació es troba els diagrames de flux de les funcionalitats del robot (equivalent a la taula que hem vist en aquest informe amb anterioritat). Les imatges dels diagrames de flux venen adjunts al .zip.

Marcos Plaza González
Niub: 20026915

Diagrama de flux del programa principal